Mobile Edge Cloud: Computation Scheduling and Caching

by

S M Shahrear Tanzil

MASc. in Electrical Engineering, The University of British Columbia - Okanagan, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Electrical & Computer Engineering)

The University of British Columbia
(Vancouver)

May 2018

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the dissertation entitled:

Mobile Edge Cloud: Computation Scheduling and Caching

submitted by S M Shahrear Tanzil in partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Electrical & Computer Engineering

Examining Committee:

Dr. Vikram Krishnamurthy, Electrical & Computer Engineering
Supervisor

Dr. Lutz Lampe, Electrical & Computer Engineering
Supervisory Committee Member

Dr. Z. Jane Wang, Electrical & Computer Engineering
Supervisory Committee Member

Dr. Rabab Ward, Biomedical Engineering
University Examiner

Dr. Steven Shechter, Business Administration
University Examiner

Abstract

Mobile edge cloud has been proposed to accommodate computational intensive application on mobile devices without augmenting their computational capacities and to combat with the growing traffic volume in the network. The main concept of mobile edge cloud is to bring computation and storage near to the end users, serving users' requests locally, and reducing wide area network latency. Although mobile edge cloud improves network performance and users' quality of experience, it brings several challenges due to possessing limited resources. The unified focus of the thesis is to design mechanisms for mobile edge cloud to maximally exploiting the computation and storage resources at the edge of the network to minimize network traffic and latency.

In the first part of the thesis, we design a distributed computational resource sharing mechanism where femtocell access points (FAPs) share their resources with neighbouring FAPs and form local clouds. The aim of forming such local *femto-clouds* is to serve computational requests locally, reducing the data transfer delay and improving users' quality of experience. The resource sharing problem is formulated as an optimization problem and a myopic procedure is presented that enables FAPs to collaboratively find its solution in a distributed fashion.

In the second and third part of the thesis, we focus on designing caching mechanisms for mobile edge network. It has been illustrated that almost 60% of the data traffic results from the asynchronous requests for some popular content. By caching those few popular content at the edge of the network, demand for the same content can be served locally, resulting in a reduction in the data traffic volume and

downloading delay in the network. In the second part of the thesis, we construct a caching scheme that accounts for content popularity prediction and properties of the network (e.g. cache size, bandwidth, and network topology). The caching scheme is further extended in the final part of the thesis that includes content popularity prediction errors and routing mechanism in the caching decision. For the caching schemes mixed-integer linear programming is used to compute where to cache the content in the network to minimize content downloading delay.

Lay Summary

Mobile edge cloud has been proposed that brings computation and storage resources near to the end users to serve users' requests locally. The unified focus of the thesis is to design mechanisms for mobile edge cloud, maximally exploiting the computation and storage resources at the edge of the network to minimize network traffic and latency.

At first, we design a distributed computational resource sharing mechanism where edge nodes share their resources with neighbouring nodes and form local clouds to reduce latency.

Then we design caching mechanisms for mobile edge network to minimize content downloading delay. The goal of these mechanisms is to cache popular content at the edge of the network, serving maximum requests locally to reduce downloading delay and traffic volume in the network. The presented caching schemes account for content popularity prediction, properties of the network, prediction errors, and routing mechanism in the caching decision.

Preface

The work presented in the thesis is based on the research works performed in Statistical Signal Processing Laboratory at the University of British Columbia-Vancouver. All the major contributions of the thesis including concept, literature review, problem formulation, analysis, and simulation were conducted by the author with assistance from Prof. Vikram Krishnamurthy. Dr. Omid Namvar Gharehshiran is the co-author of chapter 2: section 2.3 who helped the author to formulate the problem using game theory. The results presented in this thesis related to machine learning (Chapter 3: section 3.3 and section 3.4.2; and Chapter 4: section 4.4 and section 4.5.2) were performed by Dr. William Hoiles. Dr. William Hoiles also helped the author to write reviewers response letter. A detailed list of publications associated with different chapters of this thesis is provided below.

- The work of Chapter 2 has appeared in the following publications:
  - **Tanzil, S.**, Gharehshiran, O.N., and Krishnamurthy, V. (2016) A Distributed Coalition Game Approach to Femto-Cloud Formation. IEEE Transactions on Cloud Computing
  - **Tanzil, S.**, Gharehshiran, O.N., and Krishnamurthy, V. (2015) Femto-cloud formation: A coalitional game-theoretic approach. In Proceedings of the IEEE GLOBECOM
- The work of Chapter 3 has been appeared in the following publication and filed patent:

- **Tanzil, S.**, Hoiles, W., Krishnamurthy, V. (2017) Adaptive Scheme for Caching YouTube Content in a Cellular Network: A Machine Learning Approach. IEEE Access
- Hoiles, W., **Tanzil, S.**, Duan Y., Krishnamurthy, V., Ngoc Dung, and Zhang, H. (2016) Systems and methods for caching (US patent filed)

- The work of Chapter 4 has been submitted to a peer-reviewed journal
  - Hoiles, W., **Tanzil, S.**, Krishnamurthy, V. (2017) Risk-Averse Caching Policies for YouTube Content in Femtocell Networks using Density Forecasting.

Table of Contents

List of Tables

List of Figures

List of Glossary

BS	Base Station
CDN	Content Delivery Network
CM	Cache Manager
CVaR	Conditional Value at Risk
ELM	Extreme Learning Machine
FAP	Femtocell Access Point
FCM	Femto Cloud Manager
LRU	Least Recently Used
LTE	Long-Term Evolution
MCC	Mobile Cloud Computing
MEC	Mobile Edge Cloud
MILP	Mixed Integer Linear Programming
NS-3	Network Simulation-3
QoE	Quality of Experience
SAP	Smallcell Access Point
SGW	Smallcell Gateway
SLRU	Segmented Least Recently Used
UE	User Equipment
WAN	Wide Area Network

Acknowledgments

I would like to express my sincerest gratitude to my supervisor, Prof. Vikram Krishnamurthy for his guidance and persistent help throughout the research. This thesis would not have been possible without his support. I am greatly indebted to Prof. Vikram Krishnamurthy for his continuous encouragement and valuable suggestions.

I would like to extend my sincerest appreciation to Dr. William Hoiles for his valuable suggestions and always encouraging me to solve new challenges. I am also thankful to Dr. Omid Namvar Gharehshiran for his valuable insights in formulating research problem.

I would like to thank all the members of my research group for their encouragement. Last, but not least, I would like to express my gratefulness to my family.

Chapter 1

Introduction

1.1 Overview

The gaining popularity of mobile devices for ubiquitous use to enjoy a variety of applications pose new challenges to the wireless network operator. On the one hand, some of these applications are time critical and traditional centralized cloud-based solution that introduces high wide area network latency is not suitable. On the other hand, applications related to video streaming increase traffic volume in the network. Mobile edge cloud has recently been proposed to support computational and traffic intensive applications on mobile devices while satisfying latency constraint associated with the applications and reducing traffic volume in the network. The idea of mobile edge cloud is to bring cloud functionalities i.e., computation and storage near to the end users and working as an intermediate platform between users and centralized cloud. In mobile edge cloud architecture, a portion of the users' requests are served by the local cloud and forwards remaining portion of the requests to a cloud server. As a result, the amount of data traffic sent to the cloud server is reduced and users' quality of experience is improved by reducing wide area network latency. Deploying computational resources at the edge of the network is known as mobile edge computing while bringing storage resources at the edge of the network is referred to as mobile edge caching. In this

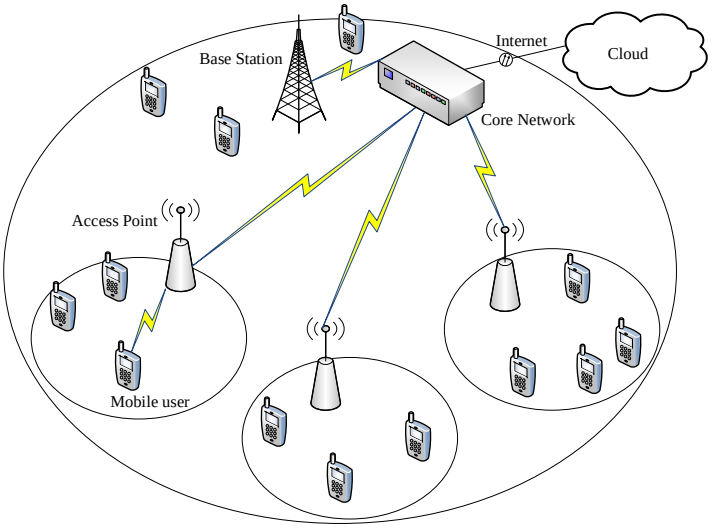

Figure 1.1: A typical mobile cloud computing architecture. Access points such as small cell, femtocell and base stations are connected to the core network via backhaul link. Core network communicates with a cloud server over the Internet.

thesis, we study both mobile edge computing and mobile edge caching. In the next section, we describe the state-of-the-art and available architectures on mobile edge computing and mobile edge caching; and their advantages over other established network architectures. Then we explain three research challenges in mobile edge cloud, their motivations and contributions of the thesis. The chapter concludes providing a brief description of the thesis organization.

1.1.1 Mobile Edge Computing

In this section, we describe the emergence of mobile edge computing and the benefits of mobile edge computing over other architectures.

Emergence of Mobile Edge Computing

Cloud computing has gained a lot of attentions from both industry and academia which generally incorporates infrastructure, platform, and software as services [2, 3]. Cloud service providers rent their resources as a utility–like electricity and users access to those resources "pay-as-you-go" basis through the Internet. In cloud computing, users enjoy powerful computing platform with software deployed by the cloud providers and store their data in the cloud storage instead of personal devices. On demand service, resource pooling, rapid elasticity, and lower capital investments are the main advantages of cloud computing [4].

On the other hand, mobile devices are becoming an integral part of human life and are used for a variety of applications e.g., entertainment, games, on-line banking, social networking, travel, and news. The resource constraints of the mobile device e.g., battery life, storage, and computations, however, hinders mobile users to enjoy various useful but computational intensive applications like health care and personal assistances [5]. One such solution to enjoy complicated applications via the mobile device, yet operating within available resources and without increasing mobile device cost; is mobile cloud computing (MCC) [6, 7]. The idea of MCC is to augment capacities of mobile devices by offloading computations to a remote cloud server over Internet. In MCC, mobile devices act as a sensory input, continuously send gathered data to the connected access point which forwards data to a cloud server via core network over the Internet. The cloud server performs all the computations and sends back the required information to the mobile device. A schematic of MCC is illustrated in Fig.1.1.

A major limitation in MCC is latency associated with data transferring to the cloud server for computations. Another bottleneck in MCC is that it requires high data rate backhaul connection between access point to the core network for faster data transfer. To alleviate such limitations in MCC, mobile edge computing architecture has recently been proposed [8].

What is mobile edge computing?

Mobile edge computing (MEC) is a new architecture that enables cloud computing functionalities at the edge of the network. The main idea of the mobile edge computing is to bring the cloud computing resources near to the end users and serving user requests locally. As a result, users' receive desired output with low latency and fewer requests are forwarded to the cloud server, resulting in a reduction in the network traffic. European Telecommunications Standards Institution (ETSI) proposed mobile edge computing architecture where they assumed that cloud functionality such as computation and storage will be integrated into the edge network devices such as macro base stations, eNodeB, small cell access points, and radio network controller [9].

Similar to the mobile edge computing, several other paradigms are available in the literature such as *fog computing*, *cloudlet*, and *femto-clouds*. These paradigms have some common ground but they differ in their usages. The term fog is used by Cisco as a metaphor instead of cloud wherein fog is closer to the end devices than the cloud [10]. Fog computing is employed between end devices especially internet of thing (IoT) devices and cloud computing platform. Fog nodes are deployed in different network tiers and perform computations associated with the most time-sensitive applications in a distributed fashion and migrate time-insensitive tasks to a cloud server. The main difference between *fog computing* and mobile edge computing is that *fog computing* aims to bring cloud functionality at the lower layer closer to the proximity of the IoT devices where mobile edge computing specifically designs to enhance cloud functionality at the edge of the wireless network such as small cell access points and base stations.

Satyanarayanan and his colleagues proposed the concept of *cloudlet* [11]: A trusted local cloud comprised of multi-core computers that are connected to over the Internet and is available for use within the proximity of mobile users. Mobile devices use Wi-Fi network to offload computational tasks to the cloudlet, which saves them a considerable amount of energy as compared to offloading over the 3G/Long Term Evolution (LTE) cellular network to a remote cloud [7, 12] in

mobile cloud computing. This prolongs the battery lifetime of mobile devices and, by reducing network latency, it also improves user's quality of experience (QoE) [13]. It is envisaged that *cloudlet* will be deployed like Wi-Fi hotspot and will act as an intermediate layer between mobile device and cloud server.

In *cloudlet*, Wi-Fi network is used for communication between mobile device and *cloudlet*. As a result, mobile devices require switching from the cellular network to Wi-Fi network whenever they are accessing to the *cloudlet*. Handover from the cellular network to the Wi-Fi network is one of the main limitations of the *cloudlet*. In addition, Wi-Fi does not provide any guarantee on the quality of service (QoS) and the range of the Wi-Fi coverage area is smaller than typical cellular base stations and access points. To avoid the limitations in the *cloudlet*, a European project named TROPIC, introduced the idea of small cell cloud namely *femto-clouds*. The main idea of *femto-clouds* is to enhance the small cell access points such as femtocell, picocell access points with computational and storage resources. The advantage of the *femto-clouds* over the *cloudlet* is that both small cell access points and mobile devices work under the same communication standard. Throughout the thesis, we use mobile edge computing and *femto-clouds* interchangeably as they both consider cloud resources at the wireless access points. The main difference between *femto-clouds* and mobile edge computing is that *femto-clouds* considers resource sharing among neighbour access points while current mobile edge computing architecture considers that there is no cooperation among access points for computation. A schematic of mobile edge computing architecture is illustrated in Fig 1.2.

A comparison among *fog computing*, *cloudlet*, and *femto-clouds* are provided in Table 1.1 [1].

Table 1.1: Comparison of *fog computing*, *cloudlet*, *femto-clouds* /MEC [1]

	fog computing	*femto-clouds* /MEC	*cloudlet*
Computing devices	IoT devices, routers, sensors	access points, base stations	dedicated computers
Access medium	Wi-Fi, bluetooth, cellular networks	cellular networks	Wi-Fi
Signalling execution	dedicated nodes are used	direct	direct

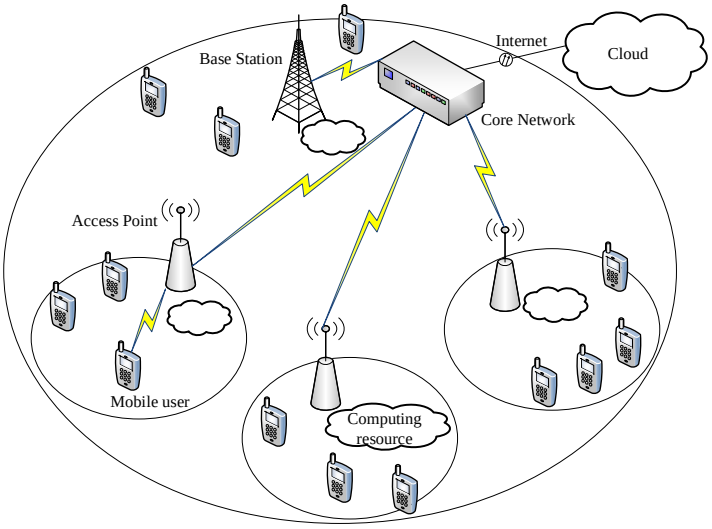

Figure 1.2: A typical mobile edge computing architecture. Access points such as small cell, femtocell and base stations are equipped with computation resources and connected to the core network via backhaul link. Core network connects to a cloud server over the Internet. As can be seen, computation requests can be served locally, resulting lower number of requests send to the core network, and reduces network traffic.

Advantages of the mobile edge computing

There are several benefits of mobile edge computing over mobile cloud computing such as low latency and reduced network traffic. In this section, we explain some of the advantages of the mobile edge computing which are as follows.

Latency: One of the major limitations of the mobile cloud computing is that it costs a significant amount of latency to transfer data associated with the tasks migration to a cloud server. The data transferring latency in MCC mainly originates from three sources that are: latency between mobile device to the connected access point, access point to the core network, and core network to cloud server. Data transferring latency between the mobile device and the connected access point

depends on several factors such as wireless channels' quality, path loss, number of users' sharing the bandwidth, and interference. Low data rate backhaul link capacity is the main reason for the latency to transfer data from access point to the core network. Finally, the wide area network latency between the core network and the cloud server depends on the distance and the number of hops between them. Once the data associated with the tasks are transferred to the cloud server, the cloud server performs all the required computations and sends back the final results to the mobile devices via core network and access points.

On the other hand, in mobile edge computing, a portion (or entire) of the tasks are processed in the edge cloud. As a result, a significant amount of latency arises from transferring data from an access point to a cloud server via core network can be reduced. A recent field trial operated by China Telecom showed that latency can be reduced to $60-90\%$ by deploying mobile edge computing [14]. [15] demonstrates that mobile edge computing reduces up to 88% latency for augmented reality application compared to mobile cloud computing.

Network traffic: Mobile edge computing enables network devices to perform computations on behalf of the mobile devices and serves computational requests locally. As a result, a fewer number of computational requests are migrated to the cloud server which leads to reduction in the network traffic.

High data rate backhaul link: High data rate backhaul link is one of the key requirements for mobile cloud computing to meet the stringent latency requirements for real-time applications such as augmented reality and video processing. However, in mobile edge computing, the network devices perform computation of the time-sensitive applications, thereby the requirement for high data rate backhaul link is lessened.

Energy: In mobile cloud computing architecture, mobile devices offload computational tasks to the could server via an access point, and a core network, incurring

a significant amount of latency. To satisfy stringent latency requirements in the real-time processing applications, mobile devices perform a portion of the tasks while offloading the remaining portion. As a result, real-time applications consume battery power of the mobile devices. On the other hand, in mobile edge computing, the latency is lower which enables the whole tasks or a higher portion of the tasks to be offloaded to the edge cloud. Thereby, energy consumption of the mobile devices is reduced in mobile edge computing, resulting in prolonging battery life-time. Gao *at el.* reveal that mobile edge computing saves 42% energy consumption compared to the mobile cloud computing [16].

1.1.2 Mobile Edge Caching

The advancement of the mobile devices has facilitated to access multimedia applications such as video on demand streaming and live chatting anytime and anywhere [17]. As a result, the wireless network is experiencing substantial growth of data traffic which will be approximate 30.6 exabytes (10^{18}) per month in 2020, an eightfold increase over 2015 [18]. The recent addition of high-density video formats such as 4K video amplifies the growth of data traffic even further [19]. It is challenging for the wireless operators such as mobile operators and Internet service providers to keep up with the massive growth of traffic while ensuring users' QoE requirements and keeping the data transmission cost low. Video content caching at the edge of the network such as caching at the base stations (BSs), at the small cell access points, at the edge routers, and at the wireless infostations is considered to be an attractive solution to address the demand for the data traffic. In this section, we explain what is mobile edge caching and the main advantages of the mobile edge caching.

What is mobile edge caching?

The usage of the mobile device to enjoy video on demand service increases the traffic growth over the wireless networks. In the traditional wireless network, video content request from the mobile users need to be fetched from a content

delivery network which is outside of the network and far away from the users. In particular, mobile users first send the video content request to the connected access points, which is then forwards it to the core network. The core network redirects the request to a content delivery network which has the requested video content. The requested content is then sent back to the users via core network and access point. As a result, a popular video that receives multiple requests asynchronously will be fetched from the content delivery network multiple times, thereby increases the network traffic. In addition, high data rate backhual links are required to meet the stringent latency constraints to avoid jerky video.

To mitigate the limitations in the current network architecture, mobile edge caching has recently been proposed [20–22]. The concept of mobile edge caching is to bring video content closer to the end users and serving video content requests locally. The cost of hardware such as solid-state devices has become cheaper in recent years which makes edge caching a cost effective solution instead of purchasing additional bandwidth for backhaul links [21, 22] for handling the immense pressure of data traffic in the network. In addition, research works on data traffic have revealed that the main source of data traffic is video traffic. In fact, 60% of the mobile data traffic are from video traffic [23] where few popular video content are requested several times asynchronously accounting for the majority of the traffic [24]. By caching those few popular video content at the edge of the network, repetitive requests for the same content can be served locally. This enables content delivery faster by shortening the communication distance between content and users. Caching at the edge of the network also alleviates the burden on the backhaul links and reduces inter-network traffic since the requested content can be served locally without the intervention of the core network. A schematic view of the mobile edge caching is illustrated in Fig.1.3.

Several paradigms are available for mobile edge caching such as femtocaching, cache enabled base stations and content-centric network. The main difference is where to put the caching resources in the network. Some of the research works advocate that caching in the macro base stations is considered to be a promising

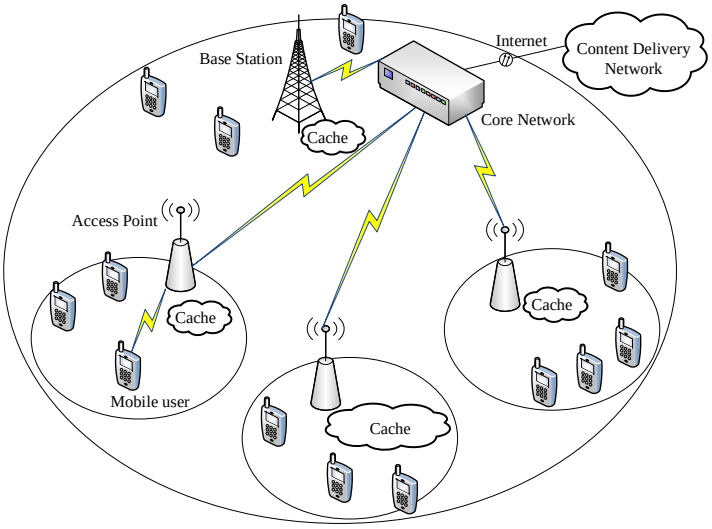

Figure 1.3: A typical mobile edge caching architecture. Access points such as small cell, femtocell and base stations are equipped with storage resources and connected to the core network via backhaul link. Core network is connected to a content delivery network over the Internet. As can be seen, content requests can be served locally, resulting lower number of requests forwarded to the core network, and reduces network traffic.

solution since coverage area of the macro base station is large and it can serve a lot of requests [25]. However, to increase the coverage and connectivity of the next generation heterogeneous network, small cell access points has become an integral part of the network. The main bottleneck in small cell network is the requirement for the high data rate backhaul links. Therefore, several research works propose that caching in the small cell network namely, femtocaching is beneficial for the network which reduces network traffic and the requirements for the high data rate backhual links [26, 27].

On the other hand, millions of routers and switches are deployed in the network. Equipping routers and switches with caching resources enable content requests to be served locally. The content-centric network is a network layer proto-

col specially designed to handle in-network caching such as caching in routers and switches [28, 29]. In this thesis, we consider edge caching in a cellular network such in base stations and small cell access points.

Benefits of mobile edge caching

The main benefits of mobile edge caching are as follows.

Latency: In mobile edge caching architecture, content are served locally resulting a shorter travelling distance for the content compared to content served from the content delivery network. As a result a significant amount of latency is reduced.

Network traffic: In mobile edge caching, cache enabled network devices serve a significant amount of content request locally and thereby a fewer requests are forwarded to the content delivery network. This leads to a reduction in the network traffic. Authors of [30] shows that a 22% of backhual link traffic can be reduced using edge caching which can be extended to a higher gain by deploying higher storage size.

High data rate backhaul link: As already mentioned, video on demand applications require high data rate backhaul link to avoid jerky video. One way to meet this requirement is to deploy high data rate backhaul link. Another solution is mobile edge caching wherein video content are served locally.

1.2 Main Contributions of the Thesis

In this section, a brief description of the major contributions of the thesis are provided. These contributions fall into two category: mobile edge computation and mobile edge caching. A schematic view of the contributions of the thesis is illustrated in Fig.1.4. A more detailed description of the contributions are provided in

separate chapters.

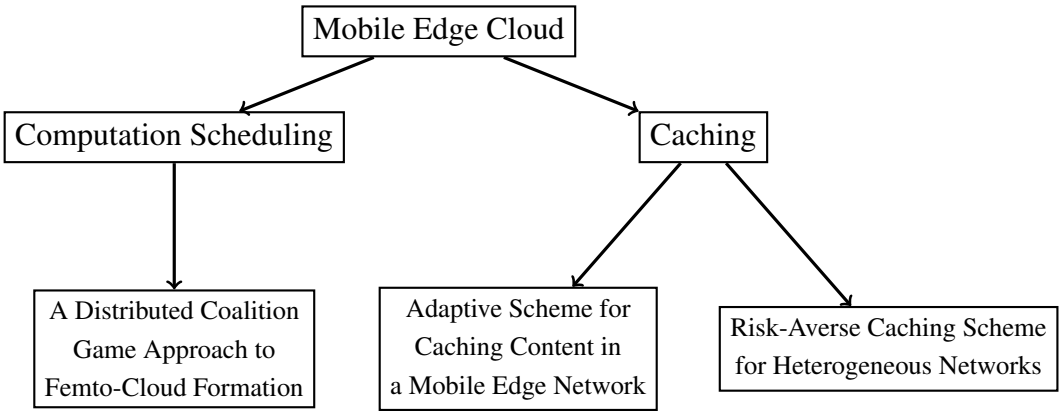

Figure 1.4: A schematic view of the contributions of the thesis.

1.2.1 A Distributed Coalition Game Approach to Femto-Cloud Formation

The aim of the work is to efficiently utilize computational resources in the edge cloud. A detailed description of the work is provided in Chapter 2. Motivation and contributions of this work are as follows.

Motivation

To increase the semantic richness of sensed data in personal assistant applications such as Apple Siri, Google Now, and Microsoft's Cortana, high data rate sensors such as vision-based sensors are required [31]. Analyzing real-time video and images captured by such sensors, however, requires intensive computational capacity, which makes it costly (in terms of energy consumption) to be processed in mobile devices. Therefore, offloading-based mechanisms have been developed to support vision-based functionalities [11, 12, 31].

One such solution is MCC [5] that augments the computational capacity of mobile devices by offloading computation and storage to a remote cloud. The interactive response essential for real-time video/image processing is, however,

limited by two major bottlenecks in MCC, namely, energy consumption and latency [11, 32–34]. Therefore, the concept of mobile edge computing such as *fog computing*, *cloudlet*, and *femto-clouds* have been proposed as explained in Section 1.1.1.

The main idea in this work is to allow edge network devices such as femtocell access points (FAPs) augmented with computational resources to cooperate with each other and form local computational pools, namely, *femto-clouds*. FAPs share the computational resources exceeding their demands in *femto-clouds*. Therefore, by maximally exploiting FAPs' local resources, such *femto-clouds* reduce latency[1] and, hence, improve end-user QoE. We assume that FAPs are deployed by different residential users. To motivate FAP owners to share their excess resources, it is natural to assume an incentive mechanism. The maximal use of FAP resources then translates into both lower handling latency and higher incentives to FAP owners. The question that this work focuses on is then: How should FAPs decide on formation of such *femto-clouds* in a distributed fashion?

The data transfer delay and limited computational capacity of FAPs impose stringent constraints that naturally prohibit formation of the grand coalition to which all FAPs join, namely, grand *femto-cloud*. Since offloading tasks to other FAPs within a *femto-cloud* incurs delay, it is not beneficial to collaborate with FAPs that are far away. On the other hand, the computation tasks exceeding the computational capacity of the *femto-clouds* have to be transported to the remote cloud. This incurs both data transfer delay and remote cloud costs. If such a cost exceeds the associated incentives, all FAPs within the *femto-cloud* will be responsible for the loss. Formation of the grand *femto-cloud* produces a huge pool of tasks, and increases the probability of such losses. Therefore, FAPs form *femto-clouds* in a way to minimize tasks that are needed to be transported to the remote cloud. The proposed *femto-cloud* formation scheme identifies such optimal localized *femto-clouds*, to which only a subset of FAPs subscribe, in a distributed fashion.

[1] Latency can be formulated as the sum of computational delay and data transfer delay.

Contributions

The main results in this work are:

- The resource sharing problem is formulated as an optimization problem with the objective to maximize the overall utility of all *femto-clouds* with constraints on the fair division of incentives among individual FAPs within a *femto-cloud*. The utility function of each *femto-cloud* takes into account the profile of request arrivals in individual femtocells, previous cooperative behaviour of FAPs, data transfer delay, and computational capacity of FAPs to determine the overall incentive available to each *femto-clouds*. Therefore, solving the formulated problem translates into finding the *femto-cloud* structure that maximizes utilization of FAPs' local resources (taking into account users' experience), yet provides incentives to FAPs for sharing their resources such that no FAP is willing to give up collaboration within its current *femto-cloud* to join another *femto-cloud*.

- The similarities between the formulated *femto-cloud* formation problem and coalition formation games enable us to employ the dynamic coalition formation algorithm in [35] to devise a procedure that prescribes individual FAPs how to revise their decisions as to which *femto-cloud* to join so as to reach the solution of formulated problem (i.e., core of the underlying coalition formation game) in a distributed fashion.

- Finally, numerical simulations using network simulator-3 (NS-3) illustrate superior performance of the proposed scheme in terms of both handling latency and incentives provided to FAP owners over alternative heuristic *femto-cloud* formation schemes. They further confirm that forming a grand *femto-cloud*, comprising of all FAPs in the network, is not always the optimal choice.

Related Work

Here, we provide a brief description of relevant works in the literature.

Collaboration among cloud providers: There is a large body of research works devoted to studying cooperation in cloud computing framework; see, e.g., [36],[37],[38]. Cooperation among mobile cloud service providers is studied in [36] for pooling computational resources with the goal to maximize revenue. The authors then use Shapley value to distribute the revenue among the collaborating cloud service providers. In [39], a cooperative outsourcing strategy is proposed which prescribes the providers whether to satisfy users' requests locally or to outsource to a certain provider. Dynamic cloud federation formation is also studied in [40].

Collaboration among femtocells: Coalition formation in femtocell network has been extensively studied in the literature; see, e.g., [41],[42],[43]. For instance, [44] studies coalition formation among femtocells in order to mitigate interference in the network. In [43], an interference management model is developed in a femtocell network wherein the cooperation problem is formulated as a coalition formation game with overlapping conditions. Rami *et al.* [45] also consider resource and power allocation in cooperative femtocell networks. All these works consider cooperation among femtocells with the aim to improve physical-layer throughput.

Incentives for cooperation in femtocell network: Femtocells are typically deployed by mobile network operators in an open/hybrid access mode, in which FAPs are willing to accommodate guest users; see, e.g., [46],[47],[48],[49]. To motivate FAP owners to adopt such an access mode, several incentive schemes have been studied in the literature, e.g., [46],[47],[48],[49],[50],[51]. Incentives can be categorized as reputation or remuneration [52]. Reputation reflects the willingness of wireless nodes' to cooperate with other nodes. Nodes receive services from other nodes based on their past behaviour—misbehaving nodes are deprived from receiving services. In contrast, remuneration-based mechanisms provide monetary incentives for cooperation, e.g., micropayment, virtual currency, E-cash, and credit transfer [53],[54],[55],[56].

Femto-clouds: Femto-clouds are relatively recent and only few studies can be found in the literature. For instance, [13] proposes a mechanism for joint optimization of communication and computational resources. In [33],[57], an offloading strategy is proposed for *femto-clouds*. All these works consider the cloud offloading mechanism while assuming that FAPs are already grouped into coalitions. Femto-clouds differ substantially from cloud radio access networks (CRAN) [58] in that FAPs are endowed with computational resources and the offloaded computations are preferred to be performed locally rather than in a centralized cloud (e.g. remote radio head in CRAN) to reduce handling latency.

Jessica *et al.* propose cluster formation strategies in [59] to handle a single user's requests in *femto-clouds*. These strategies are devised with different objectives, e.g., to minimize the experienced latency or to reduce power consumption in the cluster. This work is extended to a multi-user scenario in [60] where clusters are formed for each unserved request according to the strategies proposed in [59]. Their model, however, is suitable only for enterprise femtocell environments where all FAPs share their computational resources with each other. Moreover, cluster formation for each unserved request significantly increases the signalling overhead. To the best of our knowledge, the formulation and distributed scheme proposed in this work for formation of *femto-clouds* considering a remuneration incentive mechanism and taking into account the delay involved in migrating tasks between FAPs have not been studied before.

1.2.2 Adaptive Scheme for Caching Content in a Mobile Edge Network

The target of this work is to devise caching scheme in a edge network such as in cellular network to maximally utilize edge network storage resources. A detailed explanation of this work are provided in Chapter 3. Motivation and contributions of this work are as follows.

Motivation

The literature on content caching in edge networks has grown in recent years [61–63]. Caching methods roughly fall into two categories: i) designing new content caching methods with different objectives e.g., minimizing downloading delay, energy consumption, network congestions or maximizing users' QoE while assuming content popularity is known; and ii) developing new methods for predicting content popularity and caching the most popular content. For instance, [26] presents content caching methods for base stations (BSs) to minimize content downloading delay. The proposed coded content caching optimization problem in [26] is convex and involves the solution to a linear program. In [62], a multicast-aware caching method is developed that minimizes energy consumption in the network while the method described in [25] improves users' QoE. A cooperative content caching policy is proposed in [64] to improve network performance and users' QoE. The presented content caching problem is formulated as an integer linear programming and suboptimal solutions are devised using the hierarchical primal-dual decomposition method.

A limitation with the caching methods [26],[64],[62] is that they assume knowledge of the content popularity in advance, and assume that the content popularity follows a Zipf distribution. In reality, content popularity must be estimated [65, 66]. In [67],[68],[69], content popularity is estimated using the request statistics of the content[2]. The content is then cached based on the estimated content popularity. In [70],[71], collaborative filtering methods are used to cluster users with similar content preferences, and then cache the content based on the number of users in each cluster. For new users, their social network characteristics can be used to estimate their content preferences and cluster association. The combination of collaborative filtering with social network information is used in [72, 73] to mitigate the *cold start* and *data sparseness* issues associated with collaborative filtering. There are two main issues with employing these methods for caching content. The first is that these methods are highly intrusive

[2]In this work content refers to YouTube video files.

as the methods require knowing the user's content requests and social network information–for example, these methods can not be employed in countries such as Canada without user consent as it would violate privacy laws[3]. The second issue is that content popularity is estimated using the content request statistics. Therefore, these methods can not be used to cache new content which have no user request statistics.

Contributions

In this work, we construct an adaptive caching scheme that accounts for users' behaviour and properties of the cellular network (e.g. cache size, bandwidth, and network topology). The scheme does not require specific user's content requests and/or social network information. Instead, the popularity of the content is predicted using features of the content. This allows the caching of popular content without the need for directly observing user requests for the content. Additionally, since the popularity of the content is predicted, the content caching can be performed when the network load is minimal reducing in the overall energy usage of the network. A schematic of the adaptive caching scheme proposed in this work is displayed in Fig. 1.5. The main contributions of this work include:

- A machine learning algorithm, based on the extreme learning machine (ELM) [74], to estimate the popularity of content based on the features of the content.
- A mixed-integer linear program (MILP) is constructed to perform cache initialization that accounts for the predicted content popularity and properties of the cellular network.
- An adaptive caching scheme which uses the MILP for cache initialization, and the S3LRU (Segmented Least Recently Used with three segments) cache replacement scheme for dynamically adjusting the cache based on the requests from users. The combination of these schemes increases the overall performance of the cellular network.

[3] https://www.loc.gov/law/help/online-privacy-law/canada.php, 7 March, 2017

- The performance of the adaptive caching scheme is illustrated using real-world data from YouTube and NS-3 simulator. The results illustrated that the adaptive caching scheme improves network performance and users' QoE compared with industry standard caching schemes [26, 71, 75].

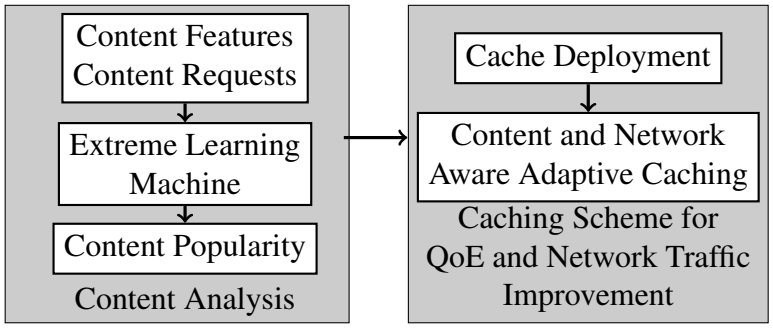

Figure 1.5: Schematic of the caching scheme. Improving the quality of experience (QoE) involves ensuring a higher cache hit ratio for requested content and reduced downloading delay.

1.2.3 Risk-Averse Caching Scheme for Heterogeneous Networks

This work is an extension of the previous work that accounts for error in the content popularity prediction. More details of this work are provided in Chapter 4. Motivation and contributions of the work are as explained below.

Motivation

As already mentioned, the design of efficient caching policies requires i) estimation of the future content requests, ii) a method to cache popular content based on the request estimates and routing protocol to deliver content to users in the network. In this work we design risk-averse caching schemes for heterogeneous networks that account for the uncertainty of predicting the future popularity of content, the content routing protocol, and balance the network traffic load.

Given the main source of traffic in the networks is video content, several methods have been proposed for estimating video content requests. These methods fall into two categories, namely, time-series or content feature based. Time-series methods use the historical content requests to predict the future content requests. Multivariate linear models [76], pure birth stochastic process [77], and ordinary differential equations [78] have all been used for constructing time-series methods for estimating content requests. A limitation with time-series methods is that they can only be used to predict the content requests of posted content. Content feature based methods use the uploader, textual, and image features of the content to predict the content requests. These methods include multivariate linear regression [79], Markov clustering [80], and extreme learning machines [81]. All these methods provide point forecasts (expected value) for the content requests. No estimate of the confidence interval, prediction interval, or measure of the uncertainty associated with the predicted content requests is provided. In this work we utilize conformal prediction algorithm for estimating the cumulative distribution function of the content requests. The key idea of the conformal prediction algorithm is to first group content, then for each group assume that the resulting error between the point forecasts and the actual content requests are generated from the same cumulative distribution function. The estimated cumulative distribution function provides a complete description of the uncertainty associated with the estimated content requests. Both the confidence interval and prediction interval of the content requests can be constructed from the estimated probability distribution. Note that density forecasting [82],[83] in the time-domain is typically referred to as prequential forecasting [84] in the economics literature.

Having estimated the future content requests, the aim is to optimally cache the content throughout the network to minimize the delay to transfer the requested content to the users. This requires that content is cached where it is likely to be requested, and to optimally transfer content throughout the network to serve user requests. The methods in [85],[86],[87] account for user specific downloading delay and wireless channel fading gain to determine where to cache content and

which user to connect to which access point. These caching methods can be considered as dynamic cache replacement methods as they adapt the cache based on the users ability to connect to different access points. However, a limitation with the caching methods [85],[86],[87] is that they do not account for the routing protocol used in the backhaul network to retrieve content that is not cached in the access points. In [88] cooperative caching is used to account for the backhaul link bandwidth in the network to cache content throughout the network. Once the content is cached in the hetergeneous network, then content and load aware routing methods are used to transfer the requested content to users [89],[90],[91]. The aim of these routing methods to ensure load balancing occurs throughout the network to minimize the delay of transferring content to users.

A common theme with the caching methods [85],[86],[87],[88] is that they contain three distinct steps. First, a point estimate of the content popularity is performed. Second, the content is cached in the network to minimize the delay of transferring content to users. Third, given the cached content, a routing method is used to deliver the content to the users. Notice that the routing method does not affect how the content is cached in the network. Additionally, the above methods are not risk-averse–that is, the point estimate of the content requests is equivalent to computing the expectation of the requests using the forecaster content request density. An issue with using the point estimate is that it does not provide a measure of the uncertainty associated with estimating the future content requests. As such, no probabilistic guarantees can be made on the network operating characteristics (e.g. downloading delay, bit-error-rate, energy consumption, cache miss ratio) for a selected caching decision.

Contributions

In this work, we include routing mechansism in the caching decision and instead of minimizing the expected downloading delay as explained in the previous work, we replace the additive expectation operator with a more general subadditive risk operator, to make caching decisions to optimize network performance. Specifi-

cally, we use the Conditional Value-at-Risk (CVaR) risk operator to construct the caching schemes in the heterogeneous network[4]. There are two important properties of CVaR that make it useful for performing caching decisions to reduce delay. First, CVaR accounts for the minimal probability of a substantial network delay for a caching decision where the total delay probability distribution is asymmetric. Second, CVaR is a coherent risk measure–that is, CVaR is monotonic, subadditive, positive homogeneous, and translation invariant. The monotonic property states that if the delay of a caching decision D_1 is always less than another caching decision D_2 almost surely, then the risk of selecting D_1 is always less than D_2. Additionally, the subadditive property guarantees that the risk of using two caching decisions D_1 and D_2 is always less than or equal to the risk associated with using D_1 and D_2 separately. Since CVaR is a coherent risk measure, the optimization of CVaR results in a convex optimization problem. As we show, the optimization of CVaR to confidently reduce the network delay while accounting for the routing protocol results in a mixed-integer linear program.

1.3 Thesis Organization

The remainder of the thesis is organized as follows. In chapter 2, we describe a mechanism to form *femto-clouds* using distributed coalition formation game. Chapter 3, explains a caching scheme in edge network. This work utilizes machine learning algorithms to predict content popularity from their features. Then an optimal caching scheme is devised using mixed integer linear programming. In Chapter 4, we describe caching schemes for heterogeneous networks. This work is an extension of the work described in Chapter 3. Finally, main research findings and future research directions are provided in Chapter 5. A schematic of the remainder of the thesis is illustrated in Fig.1.6.

[4]CVaR is one of the "big" developments for risk-averse decision making in mathematical finance; see [92–94].

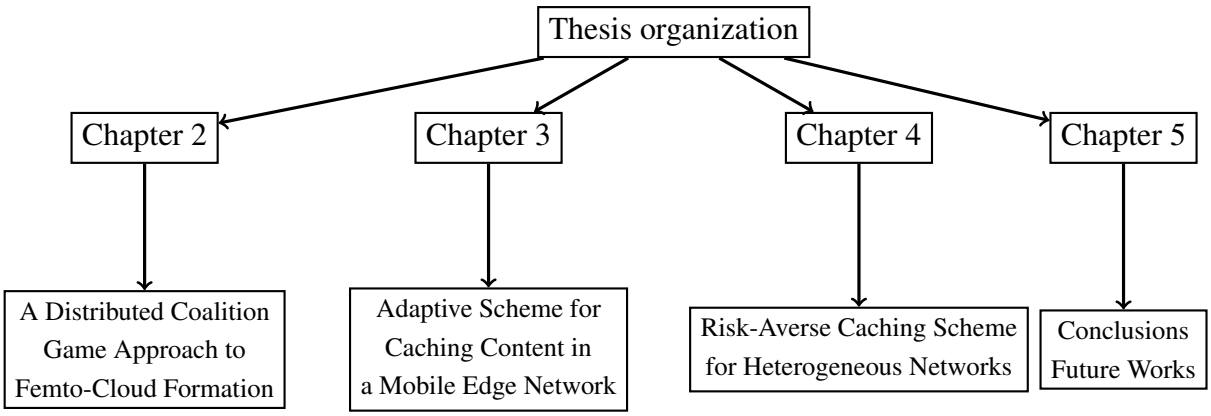

Figure 1.6: A schematic view of the remainder of the thesis.

Chapter 2

A Distributed Coalition Game Approach to Femto-Cloud Formation

In this chapter, we describe a distributed approach for *femto-clouds* formation. Recall from Chapter 1, the main target of *femto-clouds* formation is to maximally exploit computational resources at the edge of the network. In *femto-clouds*, edge nodes such as FAPs share their computational resources with other FAPs with the objective to maximize utilities that are lower overall network latency and higher incentives. FAP decides individually which *femto-clouds* to join in for performing computational tasks and receives a fair share of incentives following a distributed coalition formation game approach.

At the beginning of the chapter in Sec. 2.1, we introduce system architecture considered for the problem. The utility function for the *femto-clouds* is defined in Sec. 2.2. The distributed *femto-cloud* formation algorithm is presented in Sec. 2.3. Numerical studies are provided in Sec. 2.4. Finally, Sec. 2.5 concludes the chapter.

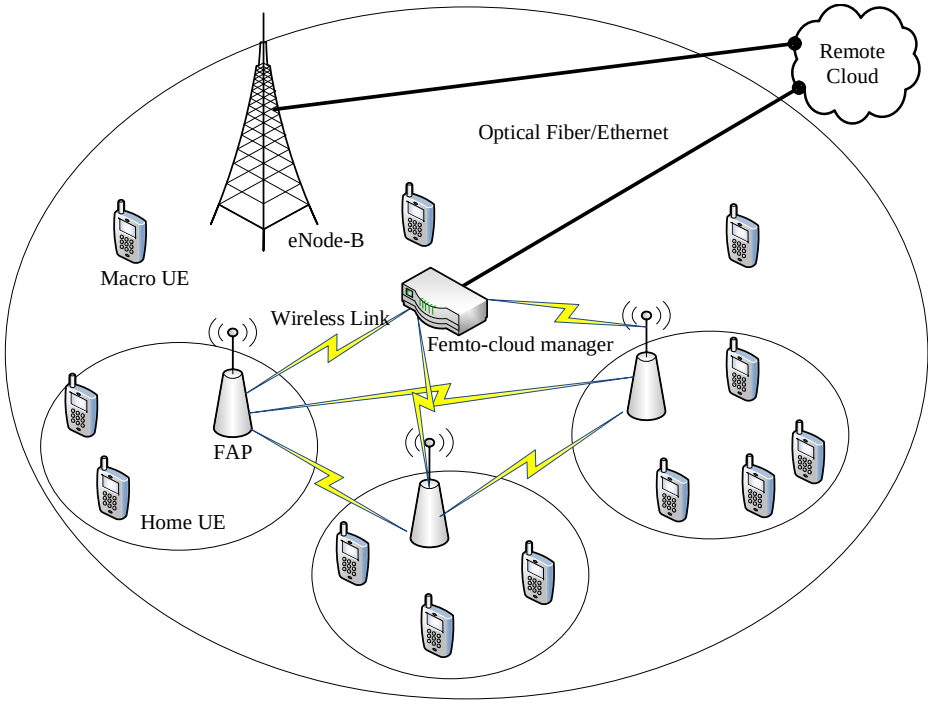

Figure 2.1: A typical *femto-cloud* architecture. The macrocell and femtocell base stations are referred to as eNode-B and femtocell access point (FAP), respectively, and the end users are referred to as user equipment (UE). FAPs are connected to their closest femtocell cloud manager (FCM) via the Z interface while FCM is linked with the remote cloud via optical fiber/Ethernet. The FAPs are also connected to the neighbouring FAPs via the Z interface.

2.1 System Architecture

We consider a UMTS LTE architecture with V FAPs/Home eNode-Bs (HeNBs) endowed with heterogeneous computational capacity. Each FAP is located in a separate room and possibly different floor of a multi-story building. The FAPs share bandwidth with a macro base station (BS) as shown in Fig. 2.1, and are deployed by different residential users.

We assume that there exist N_F femtocell cloud managers (FCMs) in the build-

ing, where $N_F < V$. The FAPs are connected to their closest FCMs via Z interface according to the proposed standalone FCM architecture in [34]. FCMs are responsible for:

(i) gathering task request information of the connected FAPs, and exchanging this information with neighbouring FCMs;

(ii) implementing the incentive mechanism by monitoring the tasks completed by each FAP;

(iii) performing computations for the *femto-cloud* formation mechanism proposed in this work.

FCMs are connected to the remote cloud via optical fiber links, hence, can offload the computational tasks of the connected FAPs to the remote cloud with no intervention of the core network. The FCMs substantially reduce the traffic generated by the MCC in the core network. It is therefore natural to assume that FCMs are installed and maintained by the mobile network operators.

It is assumed that FAPs are connected to the core network via wireless backhaul, and can be deployed by the residential users in a plug-and-play fashion. The FAPs use the 2.6 GHz licensed bandwidth to connect with FCMs, and communicate with other FAPs via the Z interface in a multicast fashion. Since FAPs and FCMs are located in different rooms/floors of the building, the FAP-FAP and FAP-FCM signal propagation undergo several losses. Here, we only consider external wall loss, shadowing loss, and the 2.6 GHz path loss models. As a result, the FAP-FAP communication delay depends on the location of the FAPs and channels' quality.

2.2 Formulation of the Femto-Cloud Formation Problem

This section formulates the *femto-cloud* formation problem. We first formulate the utility function that quantifies the performance of individual *femto-clouds* in

Sec. 2.2.1. The global *femto-cloud* formation problem with fair allocation of incentives to FAPs is then formalized in Sec. 2.2.2. We finally discuss the similarities between the formulated problem and coalition formation games. Table 2.1 summarizes the notations used in this section.

2.2.1 Local Femto-Clouds and Their Utility

Mobile devices make decisions on offloading their tasks to FAPs based on the handling latency[1] and energy [12]. If offloaded to FAPs, they will then decide whether to perform computations locally or send them to the remote cloud taking into account the users' QoE requirements, their computational capacity and workload. The main goal in this work is to motivate a cooperation protocol to maximally exploit FAPs' local resources. Neighbouring FAPs form collaborative coalitions to increase local computational capacity. Since FAPs are densely deployed, sending the data for the requested tasks to such local *femto-clouds* incurs less latency as compared to the remote cloud. This improves users' QoE while enabling FAP owners to earn incentive by sharing their excess resources.

Resource sharing problems can generally be formulated as constrained optimization problems with a utility function that trades-off the benefits and costs associated with collaboration by sharing resources. Consider a set of FAPs, indexed by the set $\mathscr{V} = \{1, 2, \ldots, V\}$, and let $\mathscr{C} \subseteq \mathscr{V}$ denote a coalition of FAPs formed for a fixed time interval over which the parameters described below remain constant. The case $|\mathscr{C}| > 1$ is referred to as a *femto-cloud*, whereas $|\mathscr{C}| = 1$ is referred to as an isolated FAP. Here, $|\cdot|$ denotes the cardinality operator. The performance of *femto-clouds* are then quantified by the function $U : 2^{\mathscr{V}} - \emptyset \rightarrow \mathbb{R}$, where $2^{\mathscr{V}}$ denotes the power set of the set of FAPs $\mathscr{V}$. This function quantifies the total incentive earned by a *femto-cloud* as the result of FAPs sharing their resources, which is then divided among the FAPs in the *femto-cloud*, and is formulated as

$$U(\mathscr{C}) = U^r(\mathscr{C}) - U^c(\mathscr{C}) + U^p(\mathscr{C}), \tag{2.1}$$

[1] Latency can be formulated as the sum of computational delay and data transfer delay.

Table 2.1: Notations and Terminology

System Parameters	Description
V	Number of FAPs
N_F	Number of FCMs
R_k	Trust/reputation value of FAP k
$d_k^{\max}$	Computational capacity of FAP k
$D_{\mathscr{C}}^{\max}$	Overall computational capacity of *femto-cloud* $\mathscr{C}$
$b_{k,l}$	Uplink data transmission rate from FAP k to FAP l
b_k	Uplink data transmission rate from FAP k to FCM
L	WAN latency for sending tasks to remote cloud

Task Request	Description
N_B	Data size
$\overline{d}_k$	Sample mean of task requests received by FAP k
$\overline{D}_{\mathscr{C}}$	Sample mean of task requests in *femto-cloud* $\mathscr{C}$
$\overline{H}_{\mathscr{C}}$	Entropy of total task requests in *femto-cloud* $\mathscr{C}$

Utility Function	Description
m_r	Revenue per unit task
m_p	Proportionality constant for trust
c_r	Remote cloud charges per unit task
c_o	Offloading delay cost
c_u	Penalty for demand uncertainty

where each term on the right hand side is described below:

The first term $U^r(\mathscr{C})$ models the revenue earned by the *femto-cloud* and is formulated as

$$U^r(\mathscr{C}) = m_r \cdot \overline{D}_{\mathscr{C}}, \tag{2.2}$$

where m_r is the revenue per unit task (\$/task). Further, $\overline{D}_{\mathscr{C}}$ denotes the sample mean of the task requests received by *femto-cloud* $\mathscr{C}$ over the past time slots since the *femto-cloud* has been modified/formed. If the requested tasks for a particular FAP exceed its computational capacity, the FAP offloads tasks to *femto-cloud* members and shares the incentive with them. Since *femto-clouds* are formed for several time slots, rather than dealing with instantaneous offloaded tasks, the incentive function relies on the previously observed statistics of requests.

The second term $U^c(\mathscr{C})$ in (2.1) represents the costs incurred by forming a *femto-cloud*, and is comprised of four terms

$$U^c(\mathscr{C}) = U^c_r(\mathscr{C}) + U^c_{o,r}(\mathscr{C}) + U^c_{o,m}(\mathscr{C}) + U^c_u(\mathscr{C}) \tag{2.3}$$

where each term is described below:

1) Remote cloud cost: When the accumulated task requests within a *femto-cloud* exceeds its computational capacity, the excess tasks have to be offloaded to the remote cloud to avoid processing delays. This incurs two types of costs:

a) Remote cloud processing cost: The term $U^c_r(\mathscr{C})$ in (2.3) models the remote cloud processing cost

$$U^c_r(\mathscr{C}) = c_r \cdot \left|\overline{D}_{\mathscr{C}} - D^{\max}_{\mathscr{C}}\right|^+, \tag{2.4}$$

where c_r is the remote cloud charges in \$/task. Further, $|x|^+ = \max\{0,x\}$, and $D^{\max}_{\mathscr{C}} = \sum_{k\in\mathscr{C}} d^{\max}_k$ is the overall computational capacity of *femto-cloud* $\mathscr{C}$, where $d^{\max}_k$ represents the computational capacity[2] of the k-th FAP. This term motivates

[2]One unit of computational capacity is equal to one unit of workload.

FAPs to form coalitions with FAPs with low workload to computational capacity ratio.

b) Remote cloud offloading delay cost: The second term $U^c_{o,r}(\mathscr{C})$ in (2.3) is the penalty associated with the data transfer delay in offloading excess *femto-cloud* workload to the remote cloud, and is formulated as

$$U^c_{o,r}(\mathscr{C}) = c_o \cdot \left(\left| \overline{D}_{\mathscr{C}} - D^{\max}_{\mathscr{C}} \right|^+ \cdot N_B \cdot \left(\frac{1}{\min_{k \in \mathscr{C}} b_k} + L \right) \right). \tag{2.5}$$

Here, N_B denotes the data size, in bytes, of a task, b_k is the uplink data transmission rate, in bytes/sec, from k-th FAP to FCM, L represents the wide area network (WAN) latency introduced by transporting the task to the remote cloud via the FCM (byte/second), and c_o ($/sec) is the dimension for proportionality constant.

2) Multicast offloading delay to FAPs: The term $U^c_{o,m}(\mathscr{C})$ in (2.3) represents the penalty for the delay in transmitting data, associated with the tasks exceeding FAPs' computational, to the *femto-cloud* into a monetary penalty. It provides incentive for FAPs to collaborate with neighbouring FAPs to decrease the handling delay and improve the QoE of users, and is formally given by

$$U^c_{o,m}(\mathscr{C}) = c_o \cdot \left(\sum_{k \in \mathscr{C}} \left| \overline{d}_k - d^{\max}_k \right|^+ \cdot \frac{N_B}{\min_{l \in \mathscr{C} - \{k\}} b_{k,l}} \right). \tag{2.6}$$

Here, $b_{k,l}$ denotes the uplink data transmission rate from the k-th FAP to the l-th FAP, $\overline{d}_k$ is the sample mean of the task request in the k-th FAP over the past time slots since the *femto-cloud* has been modified/formed. Finally, $\overline{d}_k - d^{\max}_k$ is the number of tasks that exceeds the computational capacity of the k-th FAP, and have to be sent to the cloud.

3) Demand uncertainty cost: Since *femto-clouds* are formed for multiple time slots and we use sample statistics rather than instantaneous task requests, it is important to account for deviation from the mean demand so as to avoid remote

cloud costs. The last component of the cost function captures such uncertainty in the overall *femto-cloud* demand, and is formulated as

$$U_u^c(\mathscr{C}) = c_u \cdot \overline{H}_{\mathscr{C}}, \tag{2.7}$$

where $\overline{H}_{\mathscr{C}}$ denotes the sample entropy of the accumulated task request time series. This term simply motivates FAPs to form *femto-clouds* with FAPs with less variability around their mean computational demand.

Finally, the last term $U^p(\mathscr{C})$ in (2.1) models the priority value of the coalition $\mathscr{C}$. With each FAP, there corresponds a trust value, denoted by R_k, that captures the quality of its previous cooperative behaviour [95]. By joining *femto-clouds* and successfully performing computations offloaded by other cloud members, FAPs earn trust. Femto-cloud comprising of FAPs with higher trust values are expected to perform tasks in a timely manner; therefore, the service provider is willing to provide them with higher monetary incentives as they improve the users QoE. This further eliminates free-rider FAPs that join coalitions to obtain incentives without performing tasks.

We formulate $U^p(\mathscr{C})$ as follows:

$$U^p(\mathscr{C}) = m_p \cdot \left(\sum_{k \in \mathscr{C}} R_k \cdot \frac{\min\{d_k^{\max}, f\}}{\overline{d}_k} \right), \tag{2.8}$$

where m_p (\$) is the proportionality constant that determines the relative weight of trust in formation of *femto-clouds*, and f is a system parameter[3] that depends on the overall task requests in the system. Note in the above formulation that higher priority is placed on FAPs with lower mean demand to computational capacity ratios and higher trust values. It is assumed that FCMs are responsible for updating the trust values for their neighbouring FAPs. The mechanism for updating these trust values is however out of the scope of this work, and merits a separate

[3]Taking the minimum in (2.8) is a technicality to avoid obtaining excess priority for computational capacity that exceeds the *femto-cloud* demands.

research work. We further assume that R_k remains constant for several time slots while the FCM monitors the k-th FAP cooperative behaviour, and is only updated when the *femto-clouds* structure is being modified.

2.2.2 Optimization of the Femto-clouds with FAP Incentives

As mentioned in Sec. 2.2.1, FAPs expect incentives for sharing their excess resources. Let $\mathbf{r} = (r_1, \ldots, r_V)$ denote the incentive allocation vector. Each element r_k represents the share of each FAP k from the total incentive obtained by the *femto-cloud* $\mathscr{C}$ that FAP k have joined. To make the problem mathematically tractable, the set of incentive values is confined to a finite set. Suppose Δ ($) is the smallest incentive unit. Each FAP's demand is then restricted to the set

$$\mathscr{P} = \left\{ \widehat{n}\Delta; \widehat{n} \in \mathbb{N}, 0 \leq \widehat{n}\Delta \leq \max_{\mathscr{C} \in 2^{\mathscr{V}} - \emptyset} U(\mathscr{C}) \right\}, \tag{2.9}$$

where $\mathbb{N}$ represents the set of all natural numbers, and the function $U(\cdot)$ is defined in Sec. 2.2.1. Let further $\mathscr{B}$ denote the set of all possible *femto-cloud* structures. Each *femto-cloud* structure $\mathscr{S}$ is a partition on the set $\mathscr{V}$, i.e., $\cup_{\mathscr{C} \in \mathscr{S}} \mathscr{C} = \mathscr{V}$. The *femto-cloud* formation problem is then formulated as

$$\begin{aligned} \max_{\mathscr{S} \in \mathscr{B}} \quad & \sum_{\mathscr{C} \in \mathscr{S}} \lfloor U(\mathscr{C}) \rfloor_{\Delta}, \\ \text{s.t.} \quad & r_k \in \mathscr{P}, \\ & \sum_{k \in \mathscr{C}} r_k = \lfloor U(\mathscr{C}) \rfloor_{\Delta}, \qquad \forall \mathscr{C} \in \mathscr{B}, \\ & \sum_{k \in \mathscr{C}'} r_k \geq \lfloor U(\mathscr{C}') \rfloor_{\Delta}, \quad \forall \mathscr{C}' \subseteq \mathscr{V}, \mathscr{C}' \neq \emptyset. \end{aligned} \tag{2.10}$$

where $\lfloor x \rfloor_{\Delta} = \lfloor \frac{x}{\Delta} \rfloor \cdot \Delta$ denotes the greatest integer multiple of the smallest divisible incentive unit Δ, and $\mathscr{P}$ is defined in (2.9).

Before proceeding to provide an intuitive interpretation of (2.10), a few definitions are in order. Let $\mathbf{r}$ and $\mathbf{r}'$ denote two $V \times 1$ incentive vectors. The product ordering $\mathbf{r} \leq \mathbf{r}'$ holds if and only if $r_k \leq r'_k$ for all $k \in \mathscr{V}$. An incentive allocation

$\mathbf{r}$ is then called efficient if the sum of incentives of all FAPs is equal to the maximum total incentive, achievable under the most desirable *femto-cloud* structure. In addition, if a group of FAPs can form a *femto-cloud* $\mathscr{C}'$ where the division of coalition's incentive guarantees $\mathbf{r}' \geq \mathbf{r}$, then $\mathscr{C}'$ will block the currently formed *femto-cloud* $\mathscr{C}$ and the associated incentive vector $\mathbf{r}$. An incentive vector $\mathbf{r}$ is called non-blocking if for all possible *femto-clouds* $\mathscr{C}'$, the associated incentive $\mathbf{r}'$ satisfies $\mathbf{r} \geq \mathbf{r}'$. The second constraint in (2.10) ensures that the incentives allocated to FAPs are efficient. The third constraint in (2.10) is the non-blocking condition, and can be interpreted as a fairness criterion on the division of incentives among FAPs in each *femto-cloud*. An incentive allocation vector is called fair if no FAP can gain higher incentive by sharing its resources with a different group of FAPs. The solution to (2.10) can thus be considered as the optimal *femto-cloud* structure in that: i) the computational capacity of all FAPs is maximally exploited, and ii) the FAP incentives are distributed in a fair fashion within each *femto-cloud*.

Coalition Formation Game Interpretation: The *femto-cloud* formation problem with FAP incentives outlined above fits well within the context of coalition formation games. The coalition formation games encompass cooperative games where the coalition structure plays a major role, and are defined by the pair $(\mathscr{G}, Q)$, where $\mathscr{G}$ denotes the set of players and $Q : 2^{\mathscr{G}} - \emptyset \rightarrow \mathbb{R}$ is the characteristic function[4]. This function associates with any non-empty coalition a number that quantifies the total payoff that can be gained by the coalition. A cooperative game is called superadditive if for any two disjoint coalitions $\mathscr{C}_1, \mathscr{C}_2 \subset \mathscr{G}$:

$$Q(\mathscr{C}_1 \cup \mathscr{C}_2) \geq Q(\mathscr{C}_1) + Q(\mathscr{C}_2).$$

In superadditive games, the grand coalition—the coalition consisting all players—forms the stable coalition structure. The coalition formation games encompass

[4]The term characteristic function is as used in cooperative games and is unrelated to characteristic functions in probability theory.

cooperative games where the coalition structure plays a major role. These games are generally non-superadditive; therefore, the optimal coalition structure may be comprised of several disjoint coalitions. Due to the data transfer delay and limited computational capacity of FAPs, it is intuitive that the optimal structure of *femto-clouds* has to incorporate several disjoint coalitions of FAPs. It is thus natural to formulate the *femto-cloud* formation problem as a coalition formation game with $\mathscr{G} = \mathscr{V}$ and $Q(\cdot) = U(\cdot)$. In particular, the solution of the *femto-cloud* formation problem (2.10) is identical to a solution notion in coalition formation games, namely, modified core [35]. Therefore, solving (2.10) is equivalent to finding the modified core of the underlying coalition formation game. The interested reader is referred to [96],[97],[98] for further details.

2.3 Distributed Femto-Cloud Formation and Convergence to the Core

This section presents a distributed *femto-cloud* formation algorithm that guarantees convergence to the solution of (2.10) almost surely, and elaborates on its implementation considerations.

2.3.1 Distributed Femto-Cloud Formation Algorithm

Define network state pair by $\boldsymbol{\omega} = (\mathscr{S}, \mathbf{r})$, which contains the *femto-clouds* structure $\mathscr{S}$ and the incentive vector of FAPs $\mathbf{r}$. The distributed *femto-cloud* formation procedure relies on the dynamic coalition formation algorithm proposed in [35] and is summarized below in Algorithm 1. The advantage of using the decentralized procedure in Algorithm 1 over centralized solutions is that it retains autonomy of FAP owners as whether to collaborate and better captures the dynamics of the negotiation process among them [35]. In a centralized solution, FAP owners have to be forced to follow the calculated optimal *femto-cloud* structure. In fact, if an FAP owner decides to not follow the prescription, the implemented *femto-cloud* structure is no longer the optimal solution. In contrast, the decentralized solution

implemented in Algorithm 1 mimics the natural procedure that FAP owners will follow to form collaborative groups—they explore their options and settle in the *femto-cloud* that provides the highest feasible incentive. The implementation considerations will be addressed in the next subsection.

Algorithm 1 Distributed Femto-Cloud Formation

Initialization. Set $0 < \varepsilon, \rho < 1$, where ρ is the probability of revising strategy and ε is the experimentation probability. Initialize $\omega^0 = (\mathscr{S}^0, \mathbf{r}^0)$, where

$$\mathscr{S}^0 = \big\{\{1\}, \ldots, \{V\}\big\}, \mathbf{r}^0 = \big(\widehat{r}_1, \ldots, \widehat{r}_V\big), \text{and } \widehat{r}_k = U(\{k\}).$$

Step 1. Find blocking coalitions by FCM:
Let $\mathscr{A}^n = \emptyset$. For all $\mathscr{C} \in 2^{\mathscr{V}} - \emptyset$,

$$\text{if } \sum_{k \in \mathscr{C}} r_k^n < \lfloor U(\mathscr{C}) \rfloor_\Delta, \text{ then } \mathscr{A}^n \leftarrow \mathscr{A}^n \cup \mathscr{C}.$$

Step 2. Each FAP $k \in \{1, \ldots, V\}$ independently performs:

Step 2.1. With probability ρ, continue with Step 2.2. With the remaining probability $1-\rho$, stay in the same coalition, set $r_k^{n+1} = r_k^n$, and go to Step 2.5.

Step 2.2. Compute

$$\widetilde{\mathscr{C}}_k^{n+1} = \underset{\mathscr{C} \in \mathscr{S}^n \cup \emptyset}{\operatorname{argmax}} \left(\lfloor U(\mathscr{C} \cup \{k\}) \rfloor_\Delta - \sum_{l \in \mathscr{C}, l \neq k} r_l^n \right) \tag{2.11}$$

$$\widetilde{r}_k^{n+1} = \big\lfloor U\big(\widetilde{\mathscr{C}}_k^{n+1} \cup \{k\}\big) \big\rfloor_\Delta - \sum_{l \in \widetilde{\mathscr{C}}_k^{n+1}, l \neq k} r_l^n \tag{2.12}$$

Step 2.3. If $k \in \mathscr{A}^n$, with probability ε, go to Step 2.4. With the remaining probability $1-\varepsilon$, sample uniformly from the set $\mathscr{S}^n \cup \emptyset$, denote it by $\widetilde{\mathscr{C}}_k^{n+1}$, and set $r_k^{n+1} = \widetilde{r}_k^{n+1}$, where $\widetilde{r}_k^{n+1}$ is computed according to (2.12). Go to Step 2.5.

Step 2.4. Set $r_k^{n+1} = \widetilde{r}_k^{n+1}$ and, if non-singleton, randomize among $\widetilde{\mathscr{C}}_k^{n+1}$ uniformly.

Step 2.5. If $k \neq V$, continue with the next FAP.

Step 3. Form $\omega^{n+1} = (\mathscr{S}^{n+1}, \mathbf{r}^{n+1})$.
Set $n \leftarrow n+1$ and go to Step 1.

The myopic best-reply strategy implemented in Step 2.1-2.3 of Algorithm 1 defines a finite-state Markov chain, namely, best-reply process [35]. Standard results on finite state Markov chains show that, no matter where the process starts, the probability that the best-reply process reaches a recurrent set of states after n iterations tends to one as n tends to infinity. The outcome that which of these ergodic states will eventually be reached is determined by the initial state. Under the best-reply process, absorbing states do not necessarily guarantee reaching the solution of (2.10). To address this issue, perturbation has to be introduced. That is, to allow FAPs deviate from optimal strategies and choose sub-optimal strategies with a small probability with the hope of achieving higher incentives. The interested reader is referred to [35] for details and further discussion.

Deviation from the best-reply process, namely, experimentation, is formally defined as follows: In any state, when there exists a potential *femto-cloud* $\mathscr{C}' \in 2^{\mathscr{V}}$ such that

$$\sum_{k \in \mathscr{C}'} r_k < \lfloor U(\mathscr{C}') \rfloor_{\Delta}, \tag{2.13}$$

each FAP $k \in \mathscr{C}'$ follows the best-reply process of Step 2.1-2.3 with probability $1-\varepsilon$. With the remaining probability ε, it randomly joins an existing *femto-cloud*, and demands the surplus incentive that the *femto-cloud* expects to achieve as the result of FAP k joining it. The blocking condition (2.13) is checked in Step 1 of Algorithm 1. This modified best-reply process defines a finite-state Markov chain, namely, best-reply process with experimentation [35], with the same state space as the best-reply process (without experimentation) and slightly modified transition probabilities.

The limiting distribution of the best-reply process with experimentation summarized in Algorithm 1 assigns probability one to the states $(\mathscr{S}^n, \mathbf{r}^n)$ that solve the *femto-cloud* formation problem (2.10). This result is summarized in the following theorem.

Theorem 2.3.1. *Let* $\omega^c = (\mathscr{S}^c, \mathbf{r}^c)$ *denote the states that solve the femto-cloud formation problem* (2.10)*. Then, the sample path of* $\omega^n = (\mathscr{S}^n, \mathbf{r}^n)$ *generated by*

Algorithm 1 converges almost surely to the core, i.e.,

$$P\big(\lim_{n\to\infty} \omega^n = \omega^c\big) = 1, \tag{2.14}$$

for all initializations ω^0 *if the solution set is non-empty.*

- *The proof relies on the results of [35] and the analogy between the femto-cloud formation problem* (2.10) *and the modified core of the underlying coalition formation game; see Sec. 2.2.2 for details. It is shown in [35] that the best-reply process with experimentation implemented by Algorithm 1 converges almost surely to the modified core of the coalition formation game; see [35] for the detailed proof. Comparing the definition of modified core in [35] with* (2.10) *then completes the proof.*

2.3.2 Implementation Considerations

a) Decentralized Implementation: The proposed algorithm, independently followed by each FAP, provides a decentralized solution to (2.10). This decentralized implementation relies on collaboration among the FCMs. It is assumed that FAPs monitor their users task request statistics over an interval comprising several time slots, and periodically transmit this information to their neighbouring FCM. The FCMs then exchange this information with each other so as to be able to evaluate the *femto-cloud* characteristic function (2.1) and detect for blocked FAPs (Step 1 in Algorithm 1) in their neighbourhood. Note that data size of user request information is negligible compared to the task data size. The FCMs are further responsible for providing FAPs that decided to revise their cooperation strategies with the feasible incentive (the term inside parentheses in (2.11)) in the associated *femto-cloud*, and to inform the blocked FAPs of their potential for obtaining higher incentives in other *femto-clouds*. Finally, having been enabled to communicate with each other, it is the task of FCMs to collaboratively update the network state parameter ω^n in Algorithm 1.

b) Complexity: Searching for the blocking coalition (Step 1 in Algorithm 1) is the main computational complex part of the algorithm. An exhaustive search for blocking coalition requires checking $(2^V - 1)$ different combination of coalitions. As the number of FAPs increase, the randomized searching algorithm presented in [99] can be utilized. According to the randomized searching algorithm, only a subset of the combination of coalitions is constructed for checking blocking coalition. The randomized searching algorithm still converges to the core of the coalition formation game with probability one, however, resulting in slower convergence rate.

c) Time-scales: We assume that the *femto-cloud* structure remains constant for several time slots, and FAPs update their user request statistics with the same frequency. During this period, FAPs run Algorithm 1 based on the most recent sample statistics of the user requests. Once convergence to the solution takes place, the *femto-cloud* structure and associated incentives will be followed in the next decision epoch that the *femto-cloud* structure is being revised. Note that, since FAPs and FCMs are both static, the FAP-FAP and FAP-FCM channel responses vary slowly. In the utility function, average data transmission rate is considered over which *femto-cloud* structures are assumed to remain constant.

d) Characteristic Function Parameters: The parameters m_r, c_r, c_o, c_u, and m_p in (2.1) could be mathematically be interpreted as weight factors that determine the relative importance of the different factors considered in formulation of the characteristic function such as the delay cost, the demand uncertainty cost, and remote cloud processing cost. The choice of these parameters is an interesting topic in utility theory. Clearly, the values of these parameters affect the optimal *femto-cloud* structure and incentive allocations. The particular choice of these values will depend on the specific application. For instance, in some applications users may be willing to incur longer delays to pay less for using the *femto-cloud*, in which case c_o should be smaller relative to m_r. In others, users may not tolerate

delay, where c_o should be set very large. We further emphasize that the utility function formulated in Sec. 2.2.1 is only an example that exhibits how to incorporate different factors into the implementation of *femto-clouds*. Depending on the application specifics, certain terms could be added or omitted.

e) Empty Core: Finally, imposing conditions on the utility function to ensure existence of a solution (modified core of the underlying game) could be inherently complex in some applications. To address this issue, the experimentation factor ε in Algorithm 1 can be made to diminish to zero with time, e.g., one can replace ε with $\varepsilon_n = 1/n^{\alpha}$ for $0 < \alpha < 1$. This ensures that Algorithm 1 converges to the absorbing states of the best-reply process (Steps 2.1-2.3 in Algorithm 1) if the core is empty. Extensive simulations in Sec. 2.4 numerically verify that the results still outperform alternative schemes.

2.4 Numerical Results

This section provides numerical examples to evaluate the performance of the proposed incentive-based *femto-cloud* formation scheme.

2.4.1 Object Recognition Tasks

We focus on the processing associated with the object recognition task from images and videos captured via vision-based sensors in mobile devices, which is required to support mobile augmented reality applications. The formulation, however, is general enough to be adapted to various computationally intensive applications such as face recognition, pattern recognition, and optical character recognition from images/videos[5]. In particular, applications with different computational requirements can be split into several equal-sized computational sub-tasks. The utility function only requires how many sub-tasks can be executed in the *femto-*

[5]Different object recognition applications may require different feature descriptors. The choice of the descriptor, however, is not crucial to the problem formulation and the proposed *femto-cloud* formation mechanism; it only affects the parameters of the utility function defined in Sec. 2.2.1.

cloud and the predicted demand of sub-tasks in the coalition.

Feature extraction is typically the most computationally intensive task in object recognition at the deployment stage [100]. We assume that FAPs are equipped with graphics processing units (GPUs), and are capable of performing parallel computations in their GPUs. Therefore, the feature extraction procedure can be performed either on the UE's local processor, or on the FAPs. When both UE and FAPs are busy or lack sufficient computation capacity, the task is outsourced to the remote cloud. Once extracted, the feature vectors are sent to the application server, which compares them with the training models, and sends the best matched result(s) to the UE. In the examples to follow, we consider feature extraction tasks on both images and videos. At each time, each UE can either offload an image or a video to the FAP for the feature extraction task. In the numerical examples, gPb[6] is used for feature extraction. We assume that the duration of a video is uniformly distributed between 1 to 10 seconds.

Here, one unit of workload/demand associated with feature extraction is considered to be 144 Giga floating point operations per second (GFLOPs), which is also used to define one unit of computational capacity[7]. We assume that a 3264×2448 pixels image is divided into 9 sub-images [102, 103] with each sub-image containing 1088×816 pixels and occupying 2.1 mega bit (Mb) memory. Similarly, videos are divided up into 1 second segments. Each 1 second video of 640×480 pixels and 30 frame rate occupies 2.2 Mb memory. In both cases, the feature extraction task requires approximately 144 GFLOPs which is equivalent to one unit of workload or computational capacity.

[6]The global probability algorithm (gPb) is a contour detection algorithm that achieves the best performance among all such schemes [101]. The computational requirement of gPb is 158,600 FLOPS per pixel [100].

[7]72 cores, each with 1000 MHz clock speed, are grouped together and considered as one unit of computational capacity.

2.4.2 Simulation Setup

Throughout this section, the NS-3 simulator is used for simulating LTE system architecture. We consider a city environment and use the LTE module developed by the LENA project [104, 105] as follows: We use LENA's *RandomRoomPositionAllocator* function to randomly locate 15 FAPs inside a 10-story building made of concrete and comprising 20 apartments, as depicted in Fig. 2.2. There exist 2 FCMs in the building located close to FAP-2 and FAP-15, respectively. The FCMs are connected to the remote cloud via $1Gbps$ optical fiber link. LENA's *HybridBuildingsPropagationLossModel* and *3kmphTraceFadingLossModel* functions (i.e., slowly varying Nakagami-*m* fading model) are used for propagation loss and channel fading between UEs and FAPs, respectively. We further use the *Kun2600MhzPropagationLossModel* and the *NakagamiPropagationLossModel* functions as the propagation loss model and channel fading for FAP-FAP and FAP-FCM communication. The handover is handled via the LENA's *AddX2Interface* function. UEs are further randomly located inside the building and connected to FAP using the *AttachToClosestEnb* function. At each time slot, sub-channels are allocated to users in each FAP according to the proportional fair (PF) scheduling policy with hybrid automatic repeat request (HARQ) re-transmission mechanism. Further, the UEs and FAPs are equipped with multiple input multiple output (MIMO) antennas, and support adaptive modulation and coding. UEs transmit UDP packets to the FAP. FAPs also transmit UDP packets for multi-cast communication. The data transfer rates are calculated from the *RLCTrace* files generated by the NS-3 simulator. Other NS-3 simulation parameters are listed in Table 2.2. Please note that in this section and throughout the thesis we assume that the network carries video traffic and NS-3 simulator takes care of this traffic using LTE EPS Video bearer. The service quality, strict transmission deadline, and priority of video traffic have already taken care in the NS-3 simulation setup.

Finally, the UE is considered to be an iPhone 5S and can perform 76.6 Giga floating point operations per second.

2.4.3 Numerical Examples

With the above simulation set-up, in the following examples, the effect of a single parameter is studied on the formation of *femto-clouds* while other parameters are kept constant. We set $\varepsilon = 0.3$, $\rho = 0.2$, $\Delta = 1$, and $\alpha = 0.5$ in Algorithm 1. Table 2.4 summarizes the parameters of all FAPs. These parameters are chosen so as to enable illustrating different scenarios. Each point on the graphs of Figs. 2.3-2.6 are averaged over 1000 i.i.d. realizations. The results are compared with two alternative heuristic schemes for *femto-cloud* formation. Scheme-1 is based on the relative distance of the FAPs. That is, κ FAPs with the least relative distances form a local *femto-cloud*. Scheme-2 relies on the computational capacity, the sample mean and sample entropy of demand at the FAPs. That is, FAPs are ranked based on the value of $d_k^{\max} - \overline{d}_k - \overline{H}_k$. Then, $\overline{\kappa}$ FAPs with the highest ranks are collected to form a local *femto-cloud* with $\underline{\kappa}$ lowest ranked FAPs. The procedure continues until all FAPs form/join a coalition. Coalition structures in heuristic schemes are listed in Table 2.3.

Example 1

The first example studies the effect of data transfer delay in the formation of *femto-clouds*. This scenario represents an enterprise environment where all FAPs are owned by the same authority. Therefore, we set $m_r = c_r = c_u = m_p = 0$, and $c_o = 1$ \$/sec. The utility function defined in equation(2.1) can be rewritten as follows.

$$U_{delay}(\mathscr{C}) = -U(\mathscr{C}) = U^c_{o,r}(\mathscr{C}) + U^c_{o,m}(\mathscr{C}), \tag{2.15}$$

where the first term represents remote cloud offloading delay cost which is defined in equation(2.5). The second term defined in equation(2.6) represents multicast offloading delay to FAPs. The goal will thus be to reduce the overall handling delay by forming local *femto-clouds*. In this case, $W = (w_1, \ldots, w_V)$ denote the handling latency cost allocation vector where $w_k = -r_k$, $k \in \mathscr{V}$. The femto-cloud

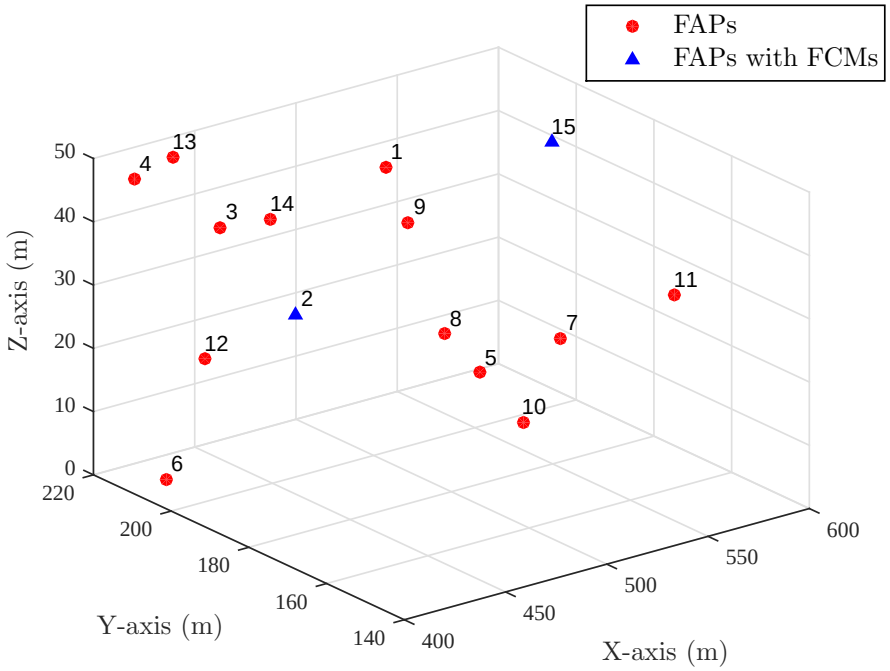

Figure 2.2: FAPs and FCMs locations inside the building. UE arrival at each FAP follows a Poisson distribution. The number of UEs in the simulation depends on the user arrival rate at each FAP (see Table 2.4).

formation problem is then reformulated as

$$\begin{aligned} \min_{\mathscr{S}\in\mathscr{B}} \quad & \sum_{\mathscr{C}\in\mathscr{S}} \lfloor U_{delay}(\mathscr{C})\rfloor_{\Delta}, \\ & \sum_{k\in\mathscr{C}} w_k = \lfloor U_{delay}(\mathscr{C})\rfloor_{\Delta}, \qquad \forall \mathscr{C}\in\mathscr{B}, \\ & \sum_{k\in\mathscr{C}'} w_k \leq \lfloor U_{delay}(\mathscr{C}')\rfloor_{\Delta}, \quad \forall \mathscr{C}'\subseteq\mathscr{V}, \mathscr{C}'\neq\emptyset. \end{aligned} \tag{2.16}$$

As can be seen from equation (2.16), the goal of the *femto-clouds* formation problem is to minimize overall handling latency. Inequality constraint ensures that no other coalition can provide lower handling latency. Algorithm 1 is redefined to

Table 2.2: Simulation setup: LTE system parameters in NS-3

Parameters	Value/Type
Adaptive Modulation & Coding	*PiroEW2010*
Bit Error Rate	0.0005
MIMO	2×2
FAP Antenna	*IsotropicAntennaModel*
External Wall Loss	10 dB
Shadowing Loss	5 dB
EPS Bearer	*GBR_CONV_VIDEO*
FAP Transmission Power	20 dbm
FAP Noise Figure	5 dbm
UE Transmission Power	10 dbm
UE Noise Figure	5 dbm
Macrocell Bandwidth	20 MHz
Mobility Model	*ConstantPosition*
Scheduler	*PfFfMacScheduler*

identify optimal coalition structures that minimize overall handing latency.

Fig. 2.3 shows the average data transfer delay in the *femto-clouds* versus the computational capacity of FAP-1. The 'Isolated FAPs' case refers to the scenario where no FAP is willing to cooperate and operates individually—that is, there exist no *femto-cloud*. In contrast, the 'Grand *femto-cloud*' refers to the case where all FAPs form one large collaborative *femto-cloud*. In the 'Isolated FAPs' case, as the computational capacity increases, FAP-1 can perform more tasks locally and offloads fewer tasks to the remote cloud. This leads to the reduction of WAN latency. Therefore, the data transfer delay of FAP-1 decreases and, hence, the overall data transfer delay in the *femto-clouds* decreases.

As can be seen in Fig. 2.3, the data transfer delay in the *femto-cloud* structures prescribed by Algorithm 1 is the lowest. This is in contrast to the grand *femto-cloud* which provides the highest delay. This is mainly because some FAPs are located far away in the building; hence, the multicast delay in the grand *femto-*

Table 2.3: Femto-cloud coalition structures in heuristic schemes

	Femto-Clouds Coalition Structure
Scheme-1	{1,2,5,8,9},{3,4,12,13,14},{6,7,10,11,15}
Scheme-2 (FAP-1 comp. capacity 0–4)	{1,2,6,10,15},{3,7,9,11,12},{4,5,8,13,14}
Scheme-2 (FAP-1 comp. capacity 6–8)	{2,6,7,10,15},{1,3,9,11,12},{4,5,8,13,14}
Scheme-2 (FAP-1 comp. capacity 10–14)	{1,2,7,10,15},{3,9,11,12,14},{4,5,6,8,13}
Scheme-2 (FAP-1 comp. capacity 16–20)	{2,7,10,12,15},{3,4,9,11,14},{1,5,6,8,13}
Scheme-2 (FAP-1 arrival rate 1)	{1,7,10,12,15},{2,3,4,9,14},{5,6,8,11,13}
Scheme-2 (FAP-1 arrival rate 2)	{1,2,7,10,15},{3,9,11,12,14},{4,5,6,8,13}
Scheme-2 (FAP-1 arrival rate 3–5)	{1,2,6,10,15},{3,7,9,11,12},{4,5,8,13,14}

Table 2.4: Simulation setup: FAP parameters in the numerical example

FAP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Trust Value	0.1	0.5	0.5	0.4	0.1	0	0.1	0.2	0.4	0.1	0.2	0.4	0.1	0	0.5
Comp. Capacity	10	10	30	10	10	5	5	20	20	15	15	5	10	10	30
User Arrival Rate	2	1	2	2	1	2	3	2	3	1	2	2	1	3	1
Mean Process. Requests	20.47	13.13	17.63	15.7	13.51	16.46	20	11.4	17.77	8.13	17.53	11.67	10.64	21.63	13.16
Entropy	3.55	2.96	3.38	3.23	3.02	3.23	3.29	2.97	3.28	2.75	3.29	2.97	2.8	3.38	2.99

cloud is high. The data transfer delay in alternative scheme-1 is higher than alternative scheme-2. This is due to the fact that some FAPs have more requests than their computational capacity, in which case tasks are transported to the remote cloud and, hence, the WAN latency increases. The 'Isolated FAPs' case ignores cooperation among FAPs, which naturally results in higher delay.

The *femto-cloud* structures are listed in Table 2.5 for various values of computational capacity for FAP-1. FAP-1 forms a *femto-cloud* with FAP-8 and FAP-15 when its individual computational capacity is low. In this case, FAP-1 offloads a portion of the requested tasks to the *femto-cloud* and reduces WAN latency as compared to transporting tasks to the remote cloud. However, as the computational capacity of FAP-1 goes beyond its demand, it joins in a different *femto-cloud* so as to be able to process tasks exceeding the capacity of the *femto-cloud* members. This reduces the overall handling delay in the *femto-cloud* and improves users' QoE.

Example 2

This example considers a scenario where FAPs are deployed by residential users. To motivate owners for sharing excess resources, monetary incentives are considered as described in Sec. 2.2.2. Therefore, FAPs are motivated to cooperate by forming *femto-clouds* not only to reduce the handling delay, but also to earn incentive. We assume $m_r = 4$ \$/task, $c_r = 5$ \$/task, $c_o = 3$ \$/sec, $c_u = 2$ \$/task, $m_p = 1$, and $f = 200$ in the characteristic function (2.1).

Figure 2.4 plots the total incentive earned by all FAPs versus computational capacity of FAP-1. As the capacity of FAP-1 increases, it can serve more tasks exceeding other FAPs' capacities within the *femto-cloud*; hence, it receives higher incentives, which in turn increases the total incentive. Note that, for lower computational capacity, the incentive obtained by FAP-1 is still higher than the 'isolated FAPs' case. This is because incentives depend not only on the revenue but also on costs associated with delay costs. By forming a *femto-cloud*, FAP-1 can save on its delay costs as explained in Example 1 and, thus, obtains higher incentives.

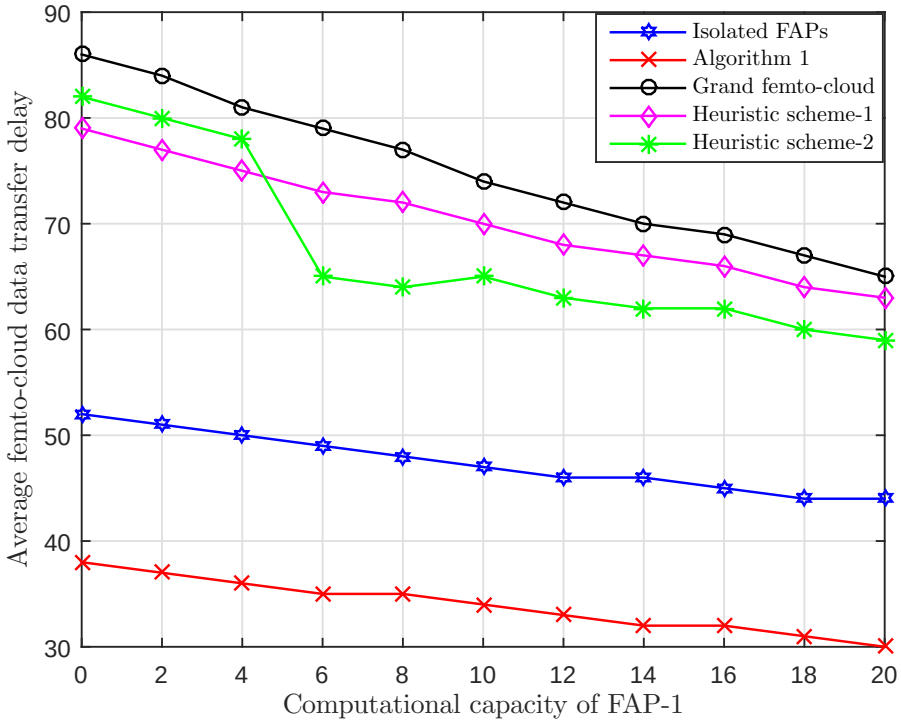

Figure 2.3: Computational capacity of FAP-1 vs. average data transfer delay per 2.2 Mb data in the *femto-clouds* ($c_0 = 1$). Here, lower data transfer delay improves users' QoE.

Figure 2.5 also displays the total incentive obtained by all FAPs versus the user arrival rate at FAP-1. As expected, as the user arrival rate at FAP-1 increases, the tasks requested at FAP-1 will increase and the incentives it receives will decrease. This is mainly because FAP-1 (in the isolated case) as well as other FAPs in the *femto-cloud* need to transport more tasks to the remote cloud, which increases the delay costs and remote cloud charges and, hence, reduces the incentives offered to FAPs. Note that this example considers the case where the charge per computation in the remote cloud is higher than the revenue obtained per computation in *femto-cloud*, i.e., $m_r \leq c_r$ in (2.1). Therefore, for fixed computational capacity, FAP-1's incentives decreases as the user arrival rate increases. The *femto-cloud* structures

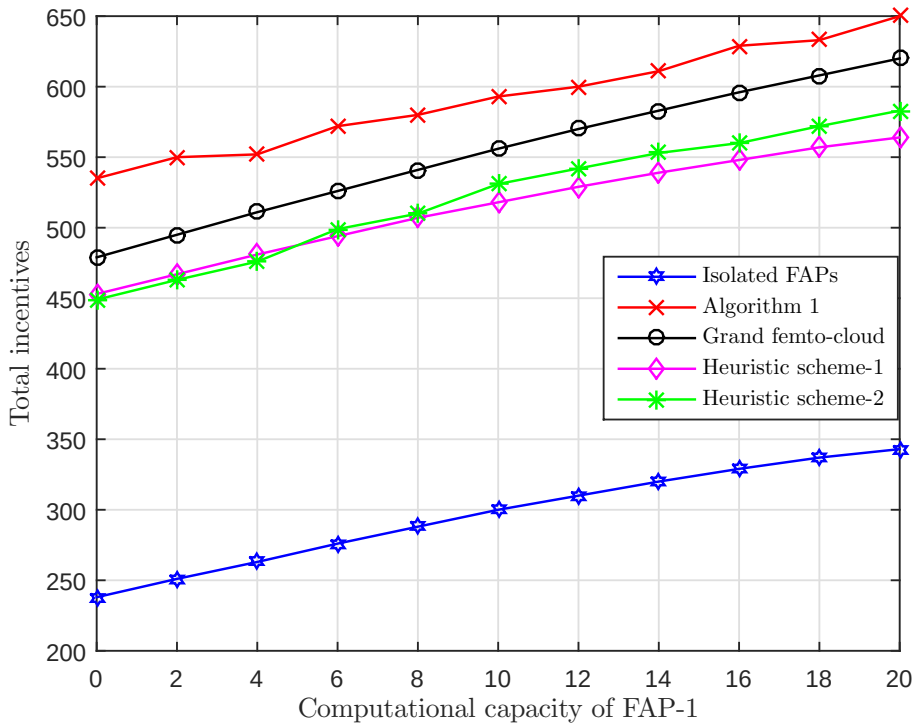

Figure 2.4: Computational capacity of FAP-1 vs. *femto-cloud* incentive. Higher incentives mean that FAP owners receive more incentives by joining in the *femto-clouds*.

are listed in Table 2.6.

Fig. 2.6 shows the delay-incentive trade-off for a range of computational capacity of FAP-1. As expected, the *femto-cloud* data transfer delay for the *femto-cloud* structures in Example 2 is higher than those obtained in Example 1. This is due to the fact that the main goal of *femto-cloud* formation in Example 2 is to maximize the incentives where delay cost c_0 is lower than the computational revenue m_r and remote cloud processing cost c_r, whereas the aim of *femto-cloud* formation in Example 1 was to reduce the data transfer delay.

Table 2.5: Femto-clouds coalition structures in Example 1

FAP-1 Computational Capacity	Femto-Clouds Coalition Structure
0	{1,8,15}, {2}, {3,7}, {4}, {5,10}, {6}, {9}, {11}, {12}, {13}, {14}
2–14	{1,6,8,15}, {2}, {3,4}, {5,10}, {7}, {9}, {11}, {12}, {13}, {14}
16–20	{1,3,4,6,8,9}, {5,10}, {11,12,15}, {2}, {7}, {13}, {14}

Table 2.6: Femto-clouds coalition structures in Example 2

FAP-1 Computational Capacity	Femto-Clouds Coalition Structure
0-10	{1,2,3,4,6,7,8,9}, {11,12,13,14,15}, {5,10}
12-20	{1,6,8,11,12,13,14,15}, {2,3,4,5,7,9,10}
FAP-1 User Arrival Rate	Femto-Clouds Coalition Structure
1-5	{1,2,3,4,6,7,8,9}, {11,12,13,14,15}, {5,10}

Example 3

In this example, we consider a hotspot scenario where all FAPs are located closely such that the multicast offloading delay among FAPs is negligible. More precisely, in such a case, the uplink data transmission rate from the k-th FAP to the l-th FAP, denoted by $b_{k,l}$ in (2.6), is much greater than N_B. This results in the $U^c_{o,m}(\mathscr{C})$ term in (2.3) being negligible compared to other terms.

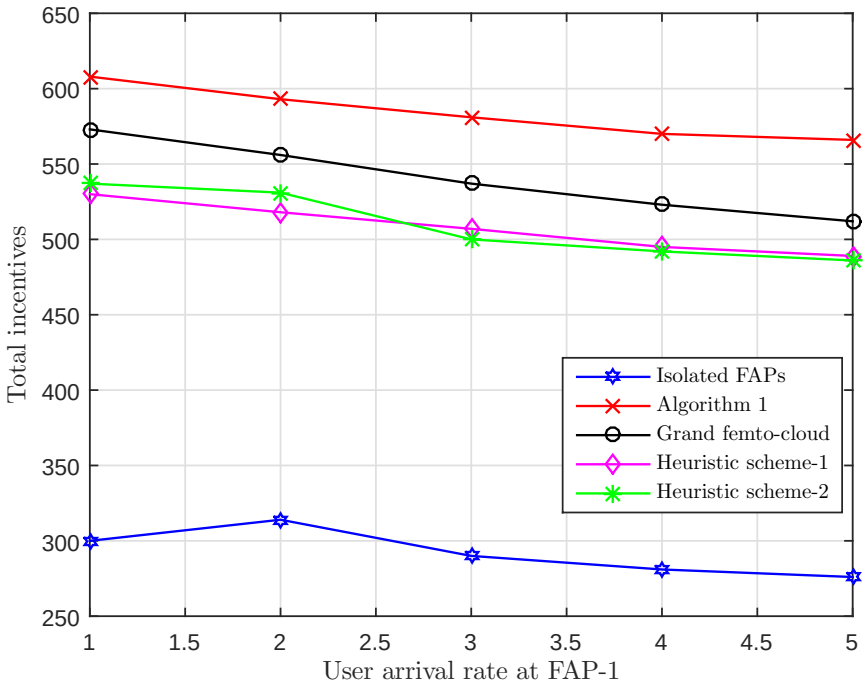

Figure 2.5: User arrival rate at FAP-1 vs. *femto-cloud* incentive. Higher incentives provide higher benefit to the FAP owners.

Figure 2.7 shows the total incentives earned by all FAPs versus computational capacity of FAP-1. Here, the grand *femto-cloud* is the optimal coalition structure and provides the highest incentives to the FAP owners compared to other heuristic schemes.

2.5 Chapter Summary

To reduce the handling latency and costs associated with offloading computationally intensive tasks to remote clouds, the local computational capacity of FAPs should be maximally exploited. To this end, this work proposed formation of *femto-clouds* comprising of several FAPs wherein their excess computational resources are shared. In exchange for sharing their excess resources, FAP own-

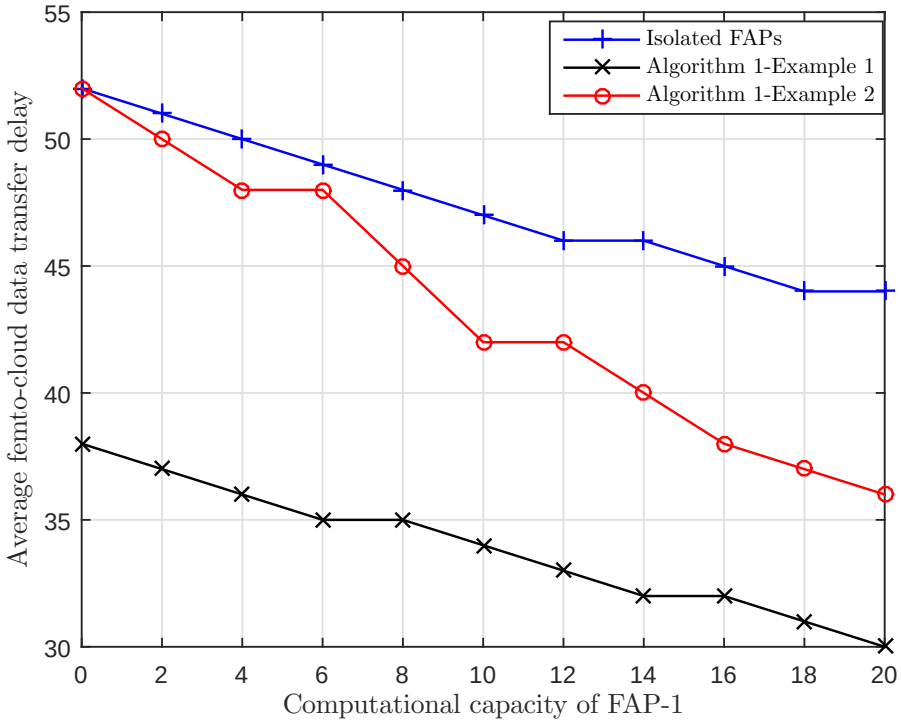

Figure 2.6: Computational capacity of FAP-1 vs. average data transfer delay 2.2 Mb in the *femto-clouds*. Lower data transfer delay improves users' QoE.

ers receive monetary incentives. We formulated the resource sharing problem as an optimization problem with the objective to maximize the overall utilities of all *femto-clouds* subject to the fair division of incentives among individual FAPs within a *femto-cloud*. We then presented a distributed *femto-cloud* formation algorithm that enabled FAPs to reach the optimal solution in a distributed fashion. We further commented on the similarities between the solution of the formulated problem and the modified core of a coalition formation game. Finally, simulation experiments using the LTE protocol stack in NS-3 showed superior performance of the proposed scheme in terms of both handling latency and incentives provided to FAP owners. They confirmed the interesting observation that a *femto-cloud*

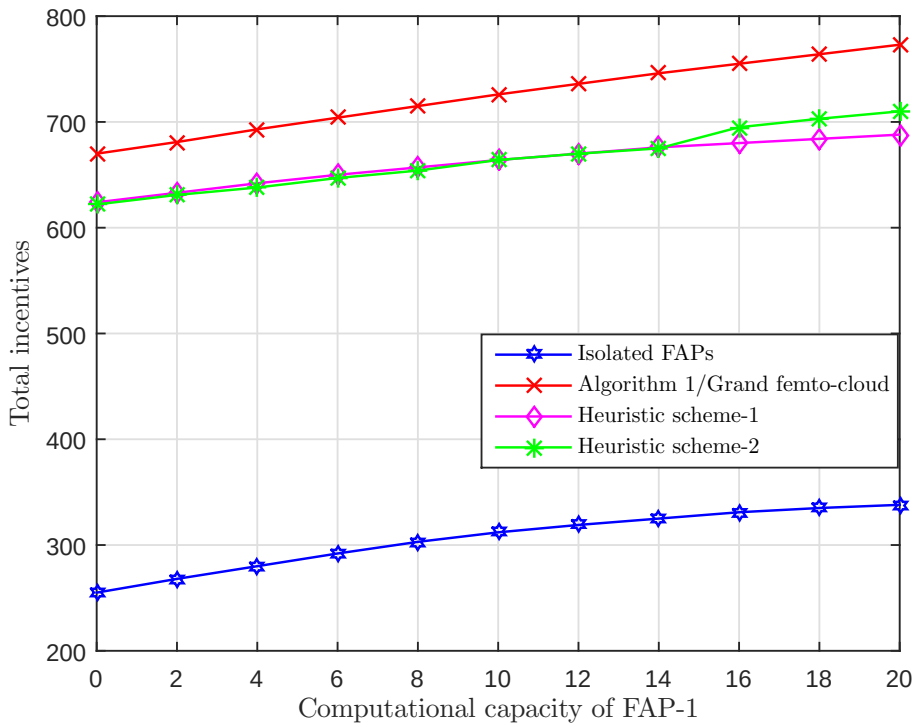

Figure 2.7: Computational capacity of FAP-1 vs. *femto-cloud* incentive. Higher incentives lead to higher benefit to the FAP owners.

comprised of all FAPs is not always optimal—in many cases, multiple disjoint *femto-clouds* resulted in reduced latency and higher incentives to the FAP owners. The numerical examples further verified the applicability of Algorithm 1 in a wide range of scenarios, e.g., hotspot area, residential, and enterprise femtocell environments.

Chapter 3

Adaptive Scheme for Caching Content in a Mobile Edge Network

In this chapter, we explain an adaptive caching scheme for a mobile edge network. The main goal of the caching scheme is to maximally utilize the caching storage at the edge network especially at the cellular network to reduce overall network traffic and content downloading delay. Recall from chapter 1, the caching scheme takes into account content popularity and network parameters for optimally decides cached content in the network. Precisely, the caching scheme estimates content popularity from their features using extreme learning machine algorithm. The caching scheme involves solving a mixed integer linear programming problem that considers predicted content popularity and network parameters. At the beginning of the chapter, we describe the system model and formulate the caching problem as a mixed integer linear programming problem. Then we describe the content popularity estimation methods using extreme learning machine. Finally, we demonstrate the efficacy of the presented scheme using YouTube dataset.

The organization of the chapter is as follows. The system model and problem formulation are presented in Sec.3.1. The content and network-aware adaptive caching scheme, which accounts for the parameters of the content popularity and technological network is presented in Sec.3.2. In Sec.3.3 we describe how ELMs

Table 3.1: Glossary of Parameters

Parameters	Description
$\mathscr{G} = (\mathscr{V}, \mathscr{E})$	Network graph
$\mathscr{V}$	Set of cache-enabled base stations
$V = \lvert\mathscr{V}\rvert$	Number of cache-enabled base stations
$\mathscr{E}$	Set of communication links
$\mathscr{F}$	Set of content
$F = \lvert\mathscr{F}\rvert$	Number of content
$\mathscr{C}$	Set of categories
$C = \lvert\mathscr{C}\rvert$	Number of categories
f_j	Size of content $j \in \mathscr{F}$
Popularity Estimation	
T	Total number of observations
t	Time index
$\mathscr{D} = \{x_j, v_j(t)\}$	Observation dataset
$h_k(x;\theta_k)$	Transfer function for hidden-layer neuron k
θ_k	Parameters of hidden-layer neuron k
β_k	Output weights
L	Total neurons of ELM
l_{th}	Video popularity threshold

can be used to efficiently estimate content popularity using both content features and the request statistics of users as they become available. The performance of ELM for caching, and content and network-aware adaptive caching scheme are illustrated in Sec.3.4 using real-world data from YouTube.

3.1 System Model and Problem Formulation

We consider a heterogeneous cellular network in a geographical region where base stations (BSs), such as eNodeB and home eNodeB, are deployed and equipped with a physical storage/cache capacity. The network shown in Fig. 3.1 can be represented by a graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. The set of vertices $\mathscr{V}$ is used to denote the set of cache enabled BSs which comprises of V BSs and indexed by $i \in \mathscr{V} = \{1, 2, \ldots, V\}$. The set of edges $\mathscr{E}$ denotes communication links among BSs. BSs can communicate with each other and with a cache manager (CM) via Xn interface [106, 107]. CM is connected to a content server such as a telco content

delivery network (CDN) via a high-speed dedicated link and is responsible for:
i) retrieving unavailable content from the content server;
ii) maintaining a lookup table that stores cached content location in the network;
iii) forwarding content request to the neighbouring BS which has the content;
iv) gathering information from BSs about the content are being requested;
v) making decision when to refresh entire cache of the BSs which can be done either specific intervals or when content popularity changes significantly;
vi) performing computations for adaptive caching.

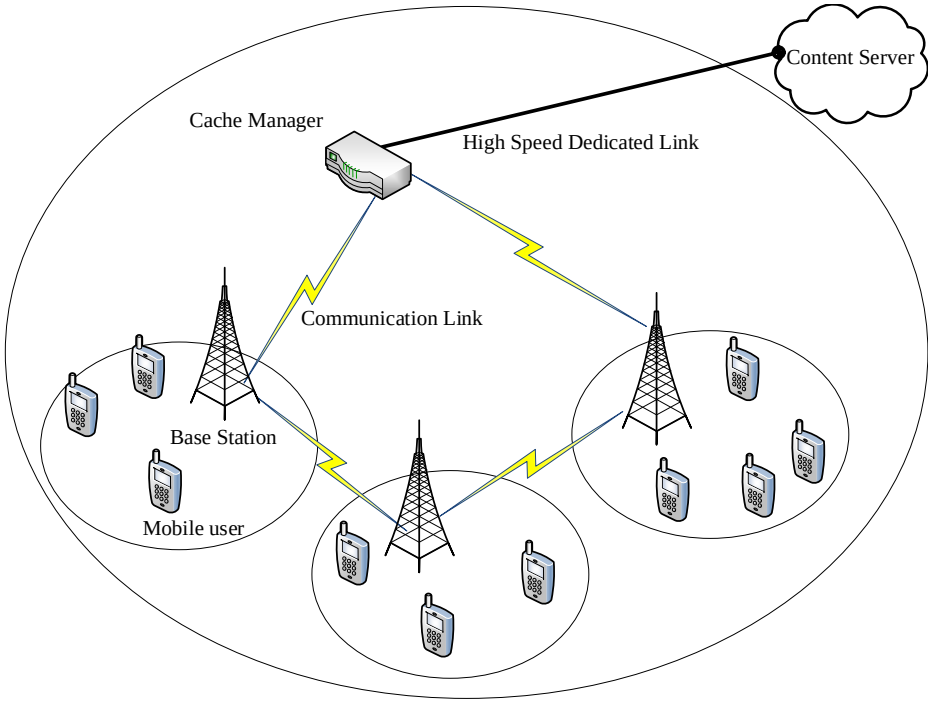

Figure 3.1: A typical network architecture. Base stations (BSs) are connected with each other and with the cache manager (CM) via heterogeneous communication links. Cache manager is connected to the content server via high speed dedicated link.

Mobile users are connected to the BSs according to a cellular network pro-

tocol. Connected BS is responsible for serving users' content requests. If a requested content is in the cache of the connected BS, the request is served instantly. In this case, the content downloading delay is lower, and hence, improves user's QoE. In addition, no additional load is put on the back-haul connection which reduces network traffic. On the other hand, when a requested content is not available at the connected BS, the request is forwarded to the CM. The CM checks the lookup table whether the requested content is available in the network. If the content is available in the network, CM performs all the required signaling to fetch the content from the neighbour BS. Content served by the neighbour BSs incur lower downloading delay and reduce network traffic. Finally, CM fetches content from the content server when requested content is unavailable in the network or when retrieving content from neighbour BSs incurs higher delay than the content server.

The content that can be cached is indexed by $\mathscr{F} = \{1, 2, \ldots, F\}$. Let f_j denote the size of the j-th content. The initial file transferring cost via the CM to a BS i is denoted by d^{gi} (second per byte), and the latency between BS i and BS l is denoted by d^{il} where $l \in \mathscr{V}$. d^{gi} and d^{il} both depends on the network topology, communication link, and routing strategy. The network topology may vary over time and routing strategy can be adjusted according to the network traffic. d^{gi} and d^{il} also depends on channel quality when BSs are communicating with one another via a wireless link.

3.2 Content and Network Aware Adaptive Caching Scheme for Cellular Base Stations

Our proposed content and network aware adaptive caching scheme proceeds as follows. Given the estimated content popularity, network topology, link capacity, routing strategy and cache deployment budget/energy usage budget in the network, the adaptive caching scheme prescribes individual BSs which content to cache and adapts its prescriptions as the preferences of users (content popularity) evolves over time. The caching scheme utilizes popularity estimation to account

for the users content request characteristics. The benefit of using popularity estimators in the caching decision is that it allows caching decisions to be made–that is when the network is not being heavily utilized, popular content can be transferred between BSs without hindering the quality of service of the network.

The rest of this section is organized as follows: Sec.3.2.1 formulates the caching scheme as a mixed-integer linear programming (MILP) while Sec.3.2.2 provides implementation considerations of the proposed caching scheme.

3.2.1 Mixed-Integer Linear Program Formulation

The adaptive caching scheme takes into account content popularity, link capacity, network topology, cache size, and network operating costs. The network operating costs include storage read/write costs and the cost of data transmission in the network. In this work, energy usage to read/write files from hardware units are considered as cache deployment cost. Hardware units draw energy when they are active due to read/write of the cached content. On the other hand, hardware units do not cache content when they are in sleep/idle mode and draw a negligible amount of energy. Therefore, higher active hardware units mean higher storage/cache size for caching at a higher energy cost to operate. Network operators allow BSs to activate a certain number of hardware unit(s) for caching. The flexible cache size facilitates network operators to provide physical storage at different BSs according to their content popularity distribution and network parameters such as link capacity and network topology while maintaining a target cache deployment cost in the network at a given time.

In the network, each BS can select a maximum of R possible hardware units (e.g. physical cache storage sizes) where each hardware unit has a storage size of s_0. Each BS can only use $r^i \in \{1,\ldots,R\}$ active hardware unit(s) due to the cache deployment cost constraint. Each hardware unit that is activated has an associated cost defined by z_0. The maximum physical storage size that can be used in the network at any given time to maintain target cache deployment cost is denoted by S. The parameter $\widehat{\mu}_j^i(t) \in [0,1]$ represents the estimated popularity of content j at

BS $i \in \mathscr{V}$ for time index $t \in \{1, \cdots, T\}$. The parameter $\widehat{\mu}_j^i(t)$ is computed by:

$$\widehat{\mu}_j^i(t) = \frac{\widehat{v}_j^i(t)}{\sum\limits_{j \in \mathscr{F}} \widehat{v}_j^i(t)}, \tag{3.1}$$

where $\widehat{v}_j^i(t)$ is total views of content j at BS i for time index t. In Sec.3.3 we provide a method to estimate the popularity of content based on the features of the content.

The MILP is formulated in (3.2) which minimizes the content downloading delay, taking into account initial file transferring cost, and cache deployment cost in the network while maintaining total cache deployment cost. There are three decision variables in the MILP:
i) $r^i \in \{1, \ldots, R\}$ denotes the number of memory units used at BS i. The total size of the physical cache used at BS i is equal to $r^i s_0$ where s_0 is the physical size of the memory units (e.g. one hardware unit may represent 200 GB of physical memory);
ii) $a_j^i \in \{0, 1\}$ which is equal to 1 if content j is cached by BS i;
iii) b_j^{il} which represents the fraction of content j served by BS i to BS l. Note that $b_j^{ll} = 1$ means that BS l caches the content j and serves the request itself. Note, the time index t is omitted for brevity in the content popularity and decision variables.

$$\min_{b_j^{il},a_j^i,r^i} \Big(w_1 \sum_{j\in\mathscr{F}} \sum_{\substack{i\in\mathscr{V} \\ l\in\mathscr{V}}} f_j \widehat{\mu}_j^l d^{il} b_j^{il} + w_2 \sum_{j\in\mathscr{F}} \sum_{i\in\mathscr{V}} f_j d^{gi} a_j^i + w_3 \sum_{i\in\mathscr{V}} r^i z_0 \Big) \tag{3.2}$$

subject to constraints

$$\sum_{i\in\mathscr{V}} b_j^{il} = 1 \quad \forall j \in \mathscr{F}, l \in \mathscr{V} \tag{3.3}$$

$$b_j^{il} \leq a_j^i \quad \forall j \in \mathscr{F}, i \in \mathscr{V}, l \in \mathscr{V} \tag{3.4}$$

$$\sum_{j\in\mathscr{F}} f_j a_j^i \leq r^i s_0 \quad \forall i \in \mathscr{V} \tag{3.5}$$

$$\sum_{i\in\mathscr{V}} r^i s_0 \leq S \tag{3.6}$$

$$b_j^{il} \geq 0 \quad \forall j \in \mathscr{F}, i \in \mathscr{V}, l \in \mathscr{V} \tag{3.7}$$

$$a_j^i \in \{0,1\} \quad \forall j \in \mathscr{F}, i \in \mathscr{V} \tag{3.8}$$

$$r^i \in \{1,2,\cdots,R\} \quad \forall i \in \mathscr{V}. \tag{3.9}$$

The first term of the objective function in the MILP (3.2) accounts for the content downloading delay in the network. The second term of equation (3.2) represents the initial content transferring cost in the network. The third term reflects cache deployment cost in the network. w_1 and w_2 are the weight of real-time latency/downloading delay cost and initial file transferring cost in the objective function, respectively. w_3 reflects the weight of cache deployment cost in the objective function. Constraint (3.3) ensures that total fraction of j-th content is equal to 1. Constraint (3.4) represents the fact that BS i can serve other BSs' request only when it caches the requested content. Constraint (3.5) ensures that each BS i fully uses the available cache where f_j is the size of the j-th content, s_0 is the size per unit of physical storage, and r^i is the number of units of storage. Constraint (3.6) maintains the cache deployment budget in the network.

3.2.2 Implementation Considerations

MILP Solution

The MILP (3.2) is NP-hard [108, 109]. Due to the size of the problem such as the number of available content and the number of network nodes, it is intractable to find optimal solutions in real-time. However, several numerical methods exist for estimating the solution to (3.2) which include: Branch-and-bound, cutting planes, branch-and-cut, branch-and-price are popular heuristic approaches to solve MILP via linear relaxation [67, 110],[111],[112],[113]. In this work, individual videos are grouped into clusters $c \in \mathscr{C}$ where $\mathscr{C} = \{1, \cdots, C\}$ represents the set of cluster/category of videos. Machine learning methods can be used to estimate the optimal clusters, however, in the YouTube network a suitable clustering method is to cluster the YouTube videos based on their associated category. Examples of categories of YouTube videos include "Entertainment", "Music", "News", "Sports", "Howto". The set of content in category $c \in \mathscr{C}$ is denoted by $F^c \subseteq \mathscr{F}$. The popularity associated with each category $c \in \mathscr{C}$ at base station $i \in \mathscr{V}$ is given by:

$$\widehat{\mu}^{ic}(t) = \sum_{j \in F^c} \widehat{\mu}^i_j(t) \tag{3.10}$$

where $\widehat{\mu}^i_j(t)$ is computed using (3.1). Given the categorical content popularity (3.10), the MILP can be used to optimally select where to cache these content.

Cache Update Frequency

In the adaptive caching scheme there are two content caching schemes used. The first is the MILP which performs the cache initialization, and the second caching scheme which uses users' request statistics to dynamically cache content. An important design parameter to consider when using the MILP (3.2) for cache initialization is when to replace the currently cached content. Given that the MILP may replace a significant portion of the cached content, typically the solution of the MILP will only be used when the network traffic flow is minimal. Fig. 3.2

provides a schematic of the adaptive caching scheme.

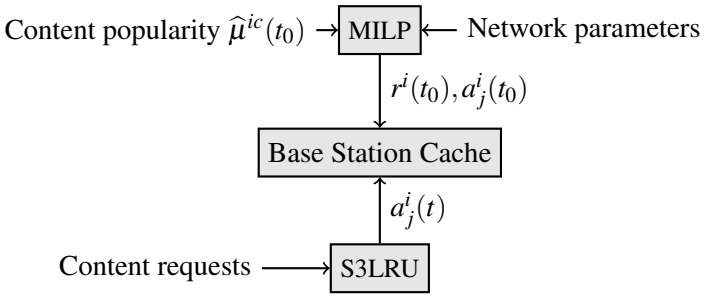

Figure 3.2: A schematic of the adaptive caching scheme. Initially the MILP uses the estimated content popularity $\widehat{\mu}^{ic}(t_0)$ (3.10) and network parameters to compute the physical cache size $r^i(t_0)s_0$ to be used at base station $i \in \mathscr{V}$, and $a^i_j(t_0) \in \{0,1\}$ indicating if content $j \in \mathscr{F}$ is to be cached at base station $i \in \mathscr{V}$ initially. t_0 indicates when the solution of the MILP is used to update the base station cache. Then, the S3LRU uses the content requests from users to compute $a^i_j(t) \in \{0,1\}$ at times $t > t_0$.

The BSs initialize their physical cache size and content to cache based on the results of the MILP. Then, the S3LRU [114] is used to select the content to cache based on the users' requests. In the S3LRU caching scheme, the physical cache of each BS is composed of three segments as illustrated in Fig. 3.3. The segments are defined by Level-3, Level-2, and Level-1 in which Level-3 has the content with the largest popularity, and Level-1 has the content with the lowest popularity. Each of the Levels is composed of a head segment and a tail segment where the head segment contains the most popular content, and the tail segment contains the least popular content in the associated level. The dynamics how the content is cached in the S3LRU is illustrated in Fig. 3.3. If the requested content is currently in the cache, then the content is moved to the head of the next level with all other content shifted down by one. Note that in Level-1, the content in the tail segment is evicted from the cache (that is, it no longer resides in the cached content). If the requested content is not currently in the cache, then the requested content from the cache manager is placed in the head segment of Level-1 of the

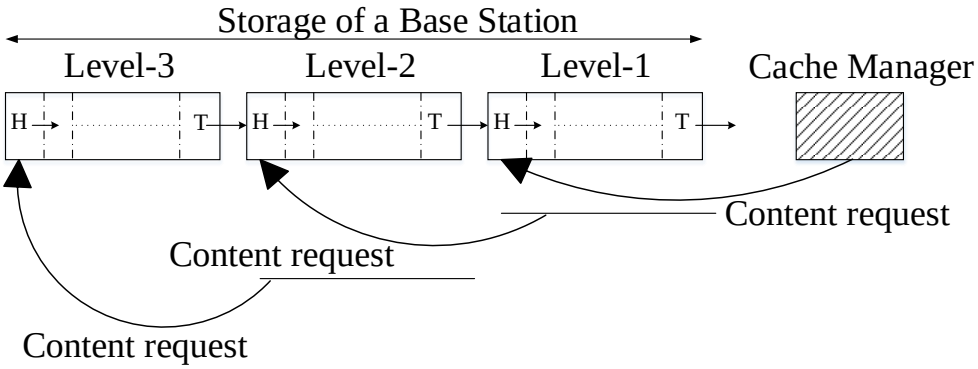

Figure 3.3: A schematic of the Segmented Least Recently Used (S3LRU) cache replacement scheme with three levels denoted by Level-3, Level-2, and Level-1. The S3LRU is used to control the cache content at each base station after the cache is initialized. Level-3 has the content with the highest popularity, and Level-1 has the content with the lowest popularity. In each Level, the most popular content is placed in the head H of the level, and the least popular content is placed in the tail T of the level. If the requested content is not currently in the cache, then the requested content is transferred from the cache manager (CM) to the head segment of the Level-1 cache. If the requested content is currently in the cache, then the content is moved to the head of the next level with all other content shifted down by one with the content in the tail of Level-1 removed from the cache.

cache. This replacement policy ensures that the popular content resides in Level-3 of the cache, and the least requested content resides in Level-1 of the cache.

Effect of Different Parameters on Caching Decision

The parameters w_1, w_2 and w_3 in equation (3.2) could be mathematically interpreted as relative importance of the different factors considered in formulation. Clearly, the choice of caching content and caching performance depend on the values of these parameters. The choice of these values depends on the network specifications. For instance, $w_1 < w_2$ when real-time latency has higher effect in the network than initial file transferring cost.

Cache Deployment Cost

Clearly, the choice of the parameter w_3 is crucial since the objective function accounts for two different types of costs i.e., energy cost and latency cost. One option is to move the cache deployment cost from the objective function and considers it as a constraint. In particular, the objective function defined in equation (3.2 is a Lagrangian relaxation of dual problem of the new formulation that considers cache deployment cost as a constraint. The optimal solutions of equation (3.2) provide a lower bound of the new formulation [115]. We used the objective function in equation (3.2) since the formulation can search optimal solutions faster than the new formulation [116].

3.3 Extreme Learning Machine (ELM) for Popularity Prediction

The adaptive caching scheme in Sec.3.2 requires the future popularity of the content to be known. In this section we use ELMs [117, 118] to estimate the popularity of content given the content features and previous request statistics of the content. Additionally, we provide methods to optimize the number of neurons of the extreme learning machine and select the optimal features to perform the popularity prediction. As an illustrative example, we focus on predicting the popularity of videos in YouTube. The prediction of popular content in YouTube is challenging as the features of YouTube video contains significant noise. Therefore the machine learning algorithms used must be able to address this challenging problem of mapping from these type of noisy features to the associated popularity of a video. Of the machine learning methods tested we found that the ELM [117, 118] provides sufficient performance to estimate the popularity of YouTube videos. Though the results presented in this section are focused on the use of ELM, the constructed features, neuron selection algorithm, and feature selection algorithm are general and can be used with other machine learning techniques.

3.3.1 Predicting Content Popularity with Extreme Learning Machines

Consider a dataset $\mathscr{D} = \big\{\{x_j, v_j(t)\} : j \in \mathscr{F} = \{1,\ldots,F\}, t \in \{1,\ldots,T\}\big\}$ of features $x \in \mathbb{R}^M$, and total views $v_j(t)$ on day t for content $j \in \mathscr{F}$. The aim is to construct a model that relates the features x to the total views v based on the dataset $\mathscr{D}$. For example, single hidden-layer feed-forward neural networks can be used for estimating the functional relationship $v_j(t)$ and the features x_j. However, in practice the selection of the model and training method is complex requiring consideration of the universal approximation ability of the model, sequential learning ability, efficiency, parallel implementation, and hardware implementation. Recently, based on the Rosenblatt perceptron [119], ELMs [74] have been introduced for learning the functional relationship between inputs x_j and output $v_j(t)$. The ELM which satisfies the universal approximation condition [120, 121], can be implemented in parallel [117], can be trained sequentially for large datasets or as new training data becomes available [122, 123], and can be efficiently implemented on field-programmable gate array devices as well as complex programmable logic devices [118]. The ELM is a single hidden-layer feed-forward neural network in which the parameters of the hidden-layer are randomly generated by a distribution, and the subsequent output weights are computed by minimizing the error between the computed output $v_j(t)$ and the measured output from the dataset $\mathscr{D}$. Each hidden-layer neuron can have a unique transfer function. Popular transfer functions include the sigmoid, hyperbolic tangent, and Gaussian however any non-linear piecewise continuous function can be utilized.

The classic extreme learning machine, presented in [124], is given by:

$$\widehat{v}_j(t) = \sum_{k=1}^{L} \beta_k h_k(x_j; \theta_k) \tag{3.11}$$

with $\beta_1, \beta_2, \ldots, \beta_L$ the weights of each neuron, $h_1(x_j), h_2(x_j), \ldots, h_L(x_j)$ the associated transfer function of each neuron, and $\widehat{v}_j(t)$ the estimated total views of the video content j at time t. Given $\mathscr{D}$, how can the ELM model parameters β_k, θ_k,

and L in (4.16) be selected? For fixed number of hidden neurons L, the ELM trains β_k and θ_k in two steps. First, the hidden layer parameters θ_k are randomly initialized. Any continuous probability distribution can be used to initialize the parameters θ_k. Second, the parameters β_k are selected to minimize the square error between the model output and the measured output from $\mathscr{D}$. Formally,

$$\beta^* \in \underset{\beta \in \mathbb{R}^L}{\operatorname{argmin}} \left\{ ||H\beta - Y||_2^2 \right\} \tag{3.12}$$

with H the hidden-layer output matrix with entries $H_{kj} = h_k(x_j; \theta_k)$ for $k \in \{1, 2, \ldots, L\}$ and $j \in \mathscr{F}$, and Y the target output with entries $Y = [y_1, y_2, \ldots, y_F]$. The solution of (4.18) is given by $\beta^* = H^+ Y$ where H^+ denotes the Moore-Penrose generalized inverse of H. Several efficient methods can be used to compute β^* (refer to Golub and Van Loan, 2012). The benefit of using the ELM, (4.16) and (4.18), is that the training only requires randomly generating parameters θ_k; the parameters β_k are computed as the solution of a linear algebraic system of equations.

3.3.2 Feature Construction for Popularity Prediction

Here we describe how the features of YouTube videos are constructed using the YouTube Application Programming Interface[1].

The meta-data of each YouTube video contains four primary components: Thumbnail, Title, Keywords (also known as tags), and Description. Additionally, each YouTube video is associated with a Channel that contains features such as the number of subscribers. The viewcount of a video is sensitive to the features of the Thumbnail, Title, Keywords, and Channel. However, features associated with the description appear not to significantly impact the viewcount of a video. This may result because when performing video searched on YouTube, only a subset of the description is provided to the users. In this work we focus on how features of the Thumbnail, Title, Keywords, and Channel can be used to estimate

[1] Specific details on how to interact with the YouTube API are provided at https://developers.google.com/youtube/v3/docs/, 7 March, 2017.

the the viewcount of a YouTube video. Note that our analysis does not include the video or audio quality, and the description of the YouTube video. These features will impact the dynamics of users subscribing to a channel, and rating the video, however, they do not directly impact the viewcount of a specific video.

For the Thumbnail, 19 features are computed which include: the blurriness (CannyEdge, Laplace Frequency), brightness, contrast, overexposure, and entropy of the thumbnail. All image analysis is performed using the OpenCV (Open Source Computer Vision) library[2]. To compute the brightness η_w of each video thumbnail, we first import the thumbnail in RGB color format. Let us denote X_{ue} as a pixel in the thumbnail image, and $R(X_{ue}) \in [0,255]$ as the red light, $G(X_{ue}) \in [0,255]$ as the green light, and $B(X_{ue}) \in [0,255]$ as the blue light associated with pixel X_{ue}. The total size of the thumbnail is given by $N_X N_Y$ with $u \in \{1,\ldots,N_X\}$ and $e \in \{1,\ldots,N_Y\}$. The brightness of the image is then computed using:

$$\begin{aligned}\eta_w(X_{ue}) &= 0.299R(X_{ue}) + 0.587G(X_{ue}) + 0.114B(X_{ue}) \\ \eta_w &= \frac{1}{765N_X N_Y}\sum_{u=1}^{N_X}\sum_{e=1}^{N_Y}\eta_w(X_{ue}).\end{aligned} \tag{3.13}$$

Typically humans' perceived brightness for color are most sensitive to variations in green light, less to red, and least to blue. The coefficients in (3.13) are associated with the perceived brightness for color, and the specific values are obtained from the OpenCV software. The contrast of each thumbnail ζ_w is computed using the RMS Contrast given by:

$$\zeta_w = \sqrt{\frac{1}{765N_X N_Y}\sum_{u=1}^{N_X}\sum_{e=1}^{N_Y}\big(\eta_w(X_{ue}) - \eta_w\big)^2}. \tag{3.14}$$

As we illustrate, the brightness η_w and contrast ζ_w of a videos Thumbnail provide important information that can be used to estimate the viewcount of a YouTube

[2]http://opencv.org/, 7 March, 2017

video.

For the Title, 23 features are computed which include: word count, punctuation count, character count, Google hits (e.g. if the title is entered into the Google search engine, how many results are found), and the Sentiment/Subjectivity of the title computed using Vader [125], and TextBlob [3]. For the Keywords, 7 features are computed which include: the number of keywords, and keyword length. In addition, to the above 49 features, we also include auxiliary video and channel features including: the number of subscribers, resolution of the thumbnail used, category of the video, the length of the video, and the first-day viewcount.

In total 54 features are computed for each video. The complete dataset used for the sensitivity analysis is given by $\mathscr{D} = \{(x_j, v_j)\}_{j \in \mathscr{F}}$, with $x_j \in \mathbb{R}^{54}$ the computed features for video $j \in \mathscr{F}$, v_j the viewcount $t = 14$ days after the video is published, and F the total number of videos used for the sensitivity analysis.

3.3.3 Optimizing the Number of Neurons in the Extreme Learning Machine

For online applications of caching where millions of videos may be cached, it is critical to consider the computational cost of evaluating the popularity of the content. In this section we consider how to select the number of neurons L in the ELM while still ensuring a sufficient predictive performance is maintained.

Several methods exist for selecting the number of neurons L (4.16) in the extreme learning machine [120, 126, 127]. In [127] a multiresponse sparse regression is utilized to order the neurons from best to worst. Then using the leave-one-out generalization metric the optimal number of neurons can be selected. Another method is to incrementally increase L until the desired accuracy or maximum number of neurons is reached [120]. In [126] neurons are added incrementally until the output of the ELM negligibly effected as measured using a non-parametric noise estimator known as the delta test. The main idea in [126] is that if the increase in accuracy of the ELM is above the estimated variance of the ELM then a

[3] http://textblob.readthedocs.io/en/dev/, 7 March, 2017

neuron is added.

The predictive performance (e.g. probability of Type-I and Type-II errors) of the ELM, as a function of the number of neurons L, is a random variable as a result of how each ELM is initialized. Given the desired predictive performance, instead of having to estimate the mean of the ELM for each L and then using a gradient decent method, one could instead employ the stochastic perturbation simultaneous approximation (SPSA) [128] method to compute the optimal number of neurons. The ELM parameter L is adapted to estimate:

$$\underset{L\in\{1,2,\dots\}}{\arg\min}\ A(L) = \mathbb{E}\left[\mathbb{P}(\text{Type-I error}) + \mathbb{P}(\text{Type-II error}) + g\tau\right] \tag{3.15}$$

where τ is the training time of the ELM, and g is a design parameter. Here $\mathbb{E}$ denotes the expectation with respect to the random variable θ defined in (4.16), and $\mathbb{P}$ denotes the probability. Since the probability of Type-I errors, Type-II errors, and the training time τ is not known explicitly, (3.15) is a simulation based stochastic optimization problem. To determine a local minimum value of $A(L)$, several types of stochastic optimization algorithms can be used [128]. In this work we use the following SPSA algorithm (Algorithm 2):

The SPSA is a gradient based stochastic optimization algorithm where the gradient is estimated numerically by random perturbation(3.17). The nice property of the SPSA algorithm is that estimating the gradient $\nabla_L A_n(L_n)$ in (3.17) requires only two measurements of the cost function (3.16) corrupted by noise per iteration. See [128] for a tutorial exposition of the SPSA algorithm. For decreasing step size $\psi = 1/n$, the SPSA algorithm converges with probability one to a local stationary point. For constant step size ψ, it converges weakly (in probability) to a local stationary point.

3.3.4 Stochastic Feature Selection

Feature selection algorithms are geared towards selecting the minimum number of features such that a sufficiently accurate prediction is possible. If the feature

Algorithm 2 SPSA Neuron Selection

Step 1: Choose initial ELM parameters L_0 by generating each from the distribution $\mathscr{N}(0,1)$, and define the video popularity threshold as l_{th}.

Step 2: For iterations $n = 1, 2, 3, \ldots$

· Estimate the Type-I and Type-II error probabilities of the ELM with L_n neurons using

$$\begin{aligned}
\text{FP} &= \sum_{j=1}^{|\mathscr{F}|} \mathbf{1}\{(\widehat{v}_j \geq l_{\text{th}}) \cap (v_j < l_{\text{th}})\}, \\
\text{TP} &= \sum_{j=1}^{|\mathscr{F}|} \mathbf{1}\{(\widehat{v}_j \geq l_{\text{th}}) \cap (v_j \geq l_{\text{th}})\}, \\
\text{FN} &= \sum_{j=1}^{|\mathscr{F}|} \mathbf{1}\{(\widehat{v}_j < l_{\text{th}}) \cap (v_j \geq l_{\text{th}})\}, \\
\text{TN} &= \sum_{j=1}^{|\mathscr{F}|} \mathbf{1}\{(\widehat{v}_j < l_{\text{th}}) \cap (v_j < l_{\text{th}})\}, \\
\mathbb{P}(\text{Type-I error}) &\approx FP/(TP+FN), \\
\mathbb{P}(\text{Type-II error}) &\approx FN/(TN+FP),
\end{aligned} \tag{3.16}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function and $\cap$ denotes the logical and operator. Given (3.16), compute the cost $\widehat{A}_n(L_n)$ by substituting (3.16) into (3.15).

· Compute the gradient estimate $\widehat{\nabla}_L \widehat{A}_n(L_n)$:

$$\widehat{\nabla}_L \widehat{A}_n(L_n) = \frac{\widehat{A}_n(L_n + \Delta_n \omega) - \widehat{A}_n(L_n - \Delta_n \omega)}{2\omega\Delta_n} \tag{3.17}$$

$$\Delta_n(j) = \begin{cases} -1 & \text{with probability } 0.5 \\ +1 & \text{with probability } 0.5 \end{cases}$$

with gradient step size $\omega > 0$.

· Update the number of neurons L_n of the ELM at step n with step size $\psi > 0$:

$$L_{n+1} = L_n - \psi \widehat{\nabla}_L \widehat{A}_n(L_n).$$

set is too large then the generalization error of the predictor will be large. Though several feature selection algorithms exist [129, 130], only the ELM feature selection method presented in [131] has utilized feature selection to improve the performance of the ELM. In this section we construct a feature selection algorithm, Algorithm 3, which relies on computing the optimal features based on the model sensitivity to variations in the features and an estimate of the generalization error of the model. Features are removed sequentially while ensuring the generalization error is sufficiently low.

The main idea of the sequential feature selection algorithm (Algorithm 3) is to sequentially remove the least useful features while ensuring that the performance of the ELM is sufficiently high. This is performed by computing the output of the ELM with all features, then computing the output with one of the features held constant at its mean (i.e. the null ELM model). If the output from the ELM and null ELM are similar under some metric then the feature held constant does not contribute significantly to the predictive performance of the ELM and should be removed. This process is repeated sequentially in Algorithm 3 until a performance threshold is reached.

3.4 Numerical Example of Content and Network Aware Adaptive Caching using Real-World YouTube Data

This section provides a numerical example to illustrate the performance of the adaptive caching using real-world YouTube dataset. Sec.3.4.1 describes simulation setup. Performance of extreme learning machine for caching is presented in Sec.3.4.2. We use results obtained from Sec.3.4.2, to evaluate the performance of the adaptive caching presented in Sec.3.4.3.

Algorithm 3 Sequential Wrapper Feature Selection

Step 0: Collect the dataset $\mathscr{D} = \big\{\{x_j, v_j\} : j \in \mathscr{F} = \{1,\ldots,F\}\big\}$ of features $x_j \in \mathbb{R}^M$ and video view count v_j for videos. Select the desired similarity metric $J(\cdot)$ (e.g. R^2 coefficient of determination).

Step 1: Train the ELM (4.16) using the dataset $\mathscr{D}$ and (4.18). Denote the predicted viewcount from the ELM by $\widehat{v}_{\mathscr{D}}$.

Step 2: For $m \in \{1,2,\ldots,M\}$, train the ELM using the dataset $\mathscr{D}^m$ where $\mathscr{D}^m$ is the dataset $\mathscr{D}$ with the feature $x_j(m)$ held at its mean for all $j \in \mathscr{F}$. Denote the predicted output from each of the $m \in \{1,2,\ldots,M\}$ ELMs by $\widehat{v}^m_{\mathscr{D}}$.

Step 3: Compute the feature index m with maximum similarity between $\widehat{v}_{\mathscr{D}}$ from Step 1 and $\widehat{v}^m_{\mathscr{D}}$ from Step 2:

$$m^* \in \underset{m\in\{1,\ldots,M\}}{\operatorname{argmax}} \{J(\widehat{v}_{\mathscr{D}}, \widehat{v}^m_{\mathscr{D}})\} \tag{3.18}$$

where $J(\cdot)$ denotes the selected similarity metric from Step 0.

Step 4: Compute the metrics of performance (Type-I and Type-II error probabilities) using the ELM trained using the dataset $\mathscr{D}^*$ where the feature m^* from Step 3 has been removed. If the metrics of performance are too high then stop. Otherwise return to Step 1 using the dataset $\mathscr{D} \leftarrow \mathscr{D}^*$.

3.4.1 Simulation Setup

The real-world YouTube data was collected using the YouTube API between the years 2013 to 2015 and consists of $F = 12,500$ YouTube videos. In the collected dataset, the viewcounts range from 10^2 to above 10^7. Therefore, to prevent the machine learning algorithms from biasing their prediction to only the videos with the highest viewcount, we scale the viewcount v_j to be on the log scale (i.e. if a video has 10^6 views then $v_j = 6$). All the content features are scaled to satisfy $x(m) \in [0,1]$ for $m \in \{1,\ldots,M\}$. Note that we also collect the category $c \in \mathscr{C}$ of each YouTube video (e.g. "Entertainment", "Music", etc.), however, this information is not included into the feature set used to train the ELMs. In total there

are 17 YouTube categories in the collected dataset. Each ELM is trained using an identical 10-fold cross validation method using the dataset $\mathscr{D}$, or in the case of the feature selection method $\mathscr{D}^*$. The trained ELMs are then used to compute both the video popularity $\widehat{\mu}_j^i(t)$ (3.1), and categorical popularity $\widehat{\mu}^{ic}(t)$ (3.10).

For evaluating the performance of the adaptive caching scheme we require the network parameters, a method to generate user requests, and the cache initialization time of the MILP. Initially, the content is cached via the solution of the MILP (3.2) at simulation time slot $p = 1$ with the parameters $w_1 = 1$, $w_2 = 0.005$, $w_3 = 1$, and $z_0 = 0.1$. The parameters w_3 and z_0 are chosen as presented in [132] in such a way so that all the cost terms in equation (3.2) receive priority according to the network specification. The topology of the network is provided in Fig. 3.4, and the parameters of the network are given by: $S = 9$ TB, $f_j = 500$ MB, $s_0 = 200$ GB, $r_i \in \{1,2\}$, and Table 3.2 provides the size of all content in each video category. To generate user requests based on the real-world YouTube data, we use the following stochastic simulation.

$$\begin{aligned} &\lambda^i \sim \mathscr{U}[1,10] \qquad i \in \mathscr{V} \\ &N_p^i \sim \text{Poisson}(\lambda^i) \qquad p \in \{1,\ldots,50,000\} \\ &J_q^i \sim \text{Cat}(\mu(t)) \qquad q \in \{1,\ldots,N_p^i\} \end{aligned} \tag{3.19}$$

where N_p^i is the total number of requests at BS i at simulation time slot p, J_q^i is the video content that is requested at BS i at simulation time slot p by the q-th request. The categorical distribution $\text{Cat}(\mu(t))$ is defined by the video popularity vector $\mu(t) = [\mu_1(t), \mu_2(t), \ldots, \mu_F(t)]$ where $\mu_j^i(t)$ is defined in (3.1). The parameter $\mu(t)$ is computed using the viewcount on day $t = 4$ ($\widehat{v}_j^i(4)$). The content popularity is assumed to be equal at each BS. With the parameters in (3.19), each BS $i \in \mathscr{V}$ will receive on average between $50,000$ to $500,000$ content requests per day. To compute the latency parameters, d^{il} and d^{gi} of equation (3.2) we use the ndnSIM2.0 (an NS-3 based simulator) software [133]. ndnSIM's *Best Route* strategy is used to transfer content between BSs.

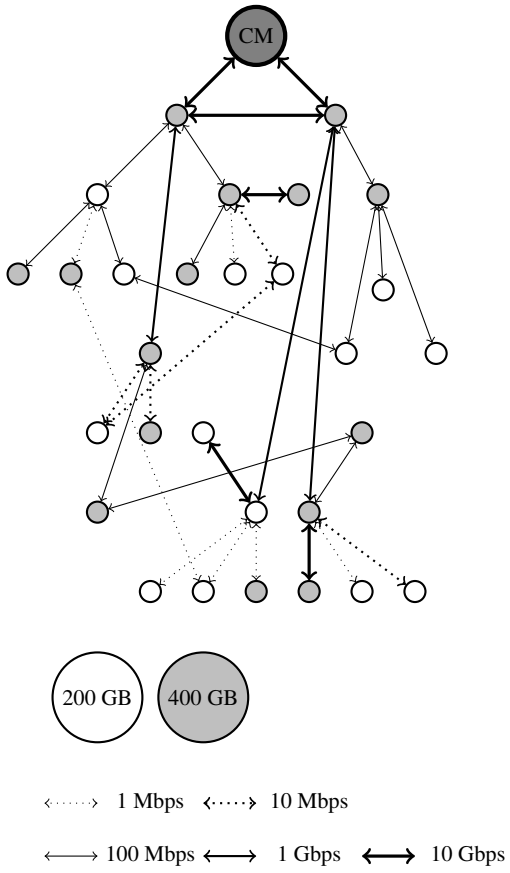

Figure 3.4: Schematic of the network. The circles with a solid black outline and light gray fill represent base stations having 400 GB storage size. Other base stations have 200 GB storage size. The associated communication links between the base stations are denoted by the connected arrows. CM is the content manager. CM represents the cache manager which has the access to all the video files, and the other nodes represent the base stations used to serve user requests.

3.4.2 Performance of Extreme Learning Machine for Caching

In this section, using the real-world data from YouTube, we illustrate how the number of neurons of the ELM (4.16) and features can be selected using Algorithm 2 and Algorithm 3. Additionally we will illustrate how the ELM can be used to both predict the popularity of new videos, and estimate the popularity of

Table 3.2: Number of files in each YouTube Category in the Collected Dataset

Category	1	2	3	4	5	6
Number of files	230	2	1475	76	151	47
Category	7	8	9	10	11	12
Number of files	3774	300	406	1839	220	913
Category	13	14	15	16	17	
Number of files	69	126	10	2855	7	

published videos.

Fig. 3.5 illustrates the mean and variance of the specificity, false negative rate, false positive rate, sensitivity, and Gmean computed using 600 independently trained ELMs for each number of neurons. Using the SPSA (Algorithm 2) we found that an ELM with $L = 300$ provides sufficient accuracy for performing the content popularity estimation.

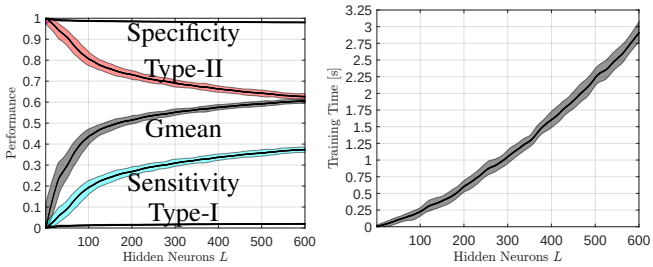

Figure 3.5: Performance of the ELM (4.16) for estimating YouTube video content popularity as a function of the number of neurons L in the ELM. The dataset used for this analysis is presented in Sec.3.3.2.

Having computed the optimal number of neurons, the next task is to select the video features which are most important for estimating the video viewcount. The performance of Algorithm 3 for selecting the YouTube features of the ELM is illustrated in Fig. 3.6. When using R^2, only 3 features are required to maintain a high level of performance for the ELM. These 3 features are the number of subscribers, contrast of the video thumbnail, and the overexposure of the video

thumbnail. This illustrates that the title and keywords contribute negligibly to the popularity of YouTube videos in the dataset analysed when using the ELM.

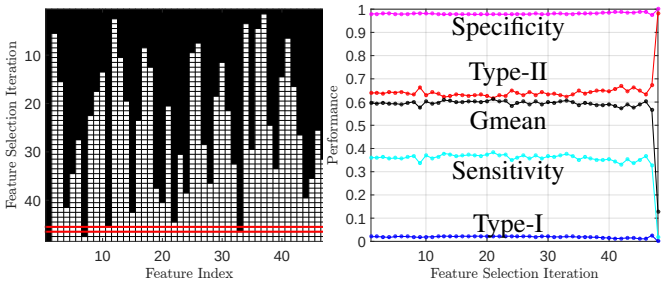

Figure 3.6: Performance of the feature selection Algorithm 3 when the R^2 coefficient of determination are used as the similarity metric.

Having optimized both the number of neurons of the ELM and the important features required for performing the popularity estimation, we now illustrate the performance of the ELM compared to several other machine learning methods. Fig. 3.7 provides a schematic of the ELM that can perform both prediction of new and published videos. The meta-data for a video is presented to the feature selection algorithm which constructs the video features $x_j \in \mathbb{R}^4$ which is composed of subscribers, contrast, overexposure, and previous day viewcount. Notice that $x_j(t)$ evolves per day t after the video is posted as new request statistics become available. The predicted viewcount on day t from the ELM is given by $\widehat{v}_j(t)$.

As expected, with no request statistics available, the predicted viewcount from the ELM has a large variance as illustrated in Fig. 3.8 for $v_j(1)$. However in typical caching applications we are only interested in the top 10% of content in which case we can construct a binary popularity estimator using the output from the ELM by thresholding. For a video popularity threshold of $l_{\text{th}} = 10^{4.5}$ views there are 1379 popular videos and 11,121 unpopular videos. Table 3.3 provides the performance of the Binary ELM classifier and several other machine learning classifiers. Note that all classifiers were trained using the same dataset and on a standard desktop computer. As seen, the ELM has comparable performance to several popular classifiers and can be evaluated efficiently. As the request statistics arrive the ELM can be used to make an accurate prediction of the viewcount dynamics as

illustrated in Fig. 3.9 for the viewcount on day 4 (i.e. $v_j(4)$). Therefore a course estimate of the popularity of videos can be made using the ELM initially, then as request statistics arrive the ELM can be used to provide a high accuracy estimate of the next day popularity of videos.

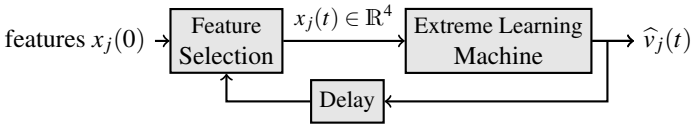

Figure 3.7: Schematic of the Extreme Learning Machine for estimating the viewcount $v_j(t)$ of video $j \in \mathscr{F}$. $x_j(0) \in \mathbb{R}^M$ is the initial set of features of video $j \in \mathscr{F}$, $x_j(t) \in \mathbb{R}^4$ are the features used by the ELM to estimate the video viewcount $\widehat{v}_j(t)$ on day t.

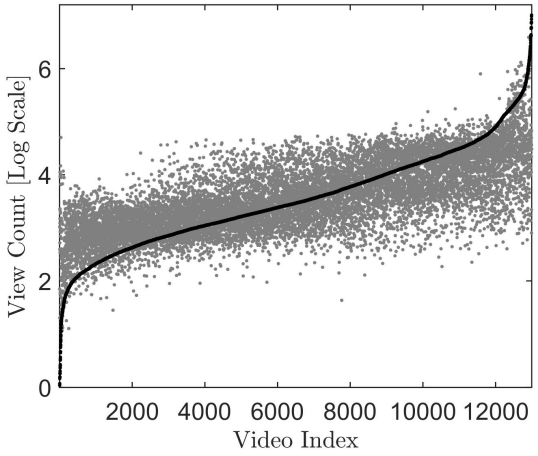

Figure 3.8: Viewcount on day 1. Real-World viewcount $v_j(t)$ (black dots) and numerically predicted viewcounts $\widehat{v}_j(t)$ (grey dots) computed using the ELM (4.16). The ELM is trained using the YouTube dataset $\mathscr{D}$ as described in Sec.3.4.

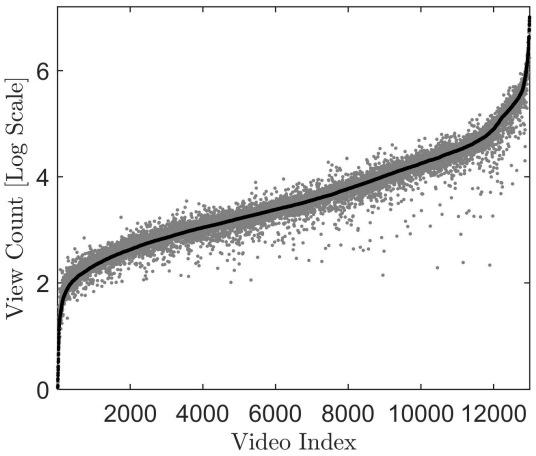

Figure 3.9: Viewcount on day 4. Real-World viewcount $v_j(t)$ (black dots) and numerically predicted viewcounts $\widehat{v}_j(t)$ (grey dots) computed using the ELM (4.16). The ELM is trained using the YouTube dataset $\mathscr{D}$ as described in Sec.3.4.

Table 3.3: ELM Performance Comparison: TP (true positive), TN (true negative), and training times

Method	**TP**	**TN**	**Times (s)**
Stochastic Gradient Boosting [134]	432	11509	5.41
Independent Component Regression [135]	769	11424	0.75
Generalized Linear Model [136]	769	11423	0.59
k-Nearest Neighbours [136]	1000	11524	24.55
Stacked AutoEncoder Deep Neural Network [137, 138]	959	11374	24.27
Boosted Tree [139]	1029	11419	16.32
Extreme Learning Machine	1067	11399	0.54

3.4.3 Performance of the Content and Network Aware Caching Scheme

This section illustrates the performance of the adaptive caching scheme presented in Sec.3.2. Specifically, the content downloading delay and cache hit ratio from the adaptive caching scheme are compared with the *most popular* and *random cache deployment* schemes.

In the *most popular* caching scheme, each BS caches the most popular estimated categories (computed using request statistics) of the content until each base station's cache is full [26, 71]. The *random cache deployment* scheme accounts for content popularity (computed using content request statistics) and network parameters using an MILP which does not account for changes in the physical cache sizes at the BSs [75]. Additionally, the method in [75] incorporates a cache replacement scheme using the LRU (Least-Recently-Used) scheme. In this section, we compare the performance of the adaptive caching scheme with the schemes in [26, 71, 75] with the predicted popularity from the ELM used in place of the request statistics, and the S3LRU used in place of the LRU cache replacement scheme.

To compute the optimal size of caches for the BSs, we solve the MILP problem (3.2). The results of the MILP are provided in Fig. 3.4 where the circles with a solid black outline and light gray fill represent BSs allocated with a 400 GB cache storage size. Other BSs are allocated with a 200 GB cache storage size. As seen in Fig. 3.4, the physical cache sizes in the network are heterogeneous. The MILP, based on the network topology, link capacity, routing strategy, and content popularity, optimally selects the physical cache sizes to use to reduce network energy consumption.

Fig. 3.10 shows the cumulative content downloading delay in the network vs. the simulation time. From Fig. 3.10, the adaptive caching scheme has the smallest cumulative content downloading delay compared with the *most popular* and *random cache deployment* schemes. This allows the adaptive caching scheme to increase the users' QoE in the network compared to these other caching schemes. The main reason the adaptive caching scheme outperforms the *most popular* and *random cache deployment* schemes is that the adaptive caching scheme considers adjacent BSs physical cache sizes and cached content to improve network performance. Comparing the performance of the *most popular* caching scheme and *random cache deployment* scheme, it is clear that methods which account for network parameters while making caching decisions will improve the users' QoE.

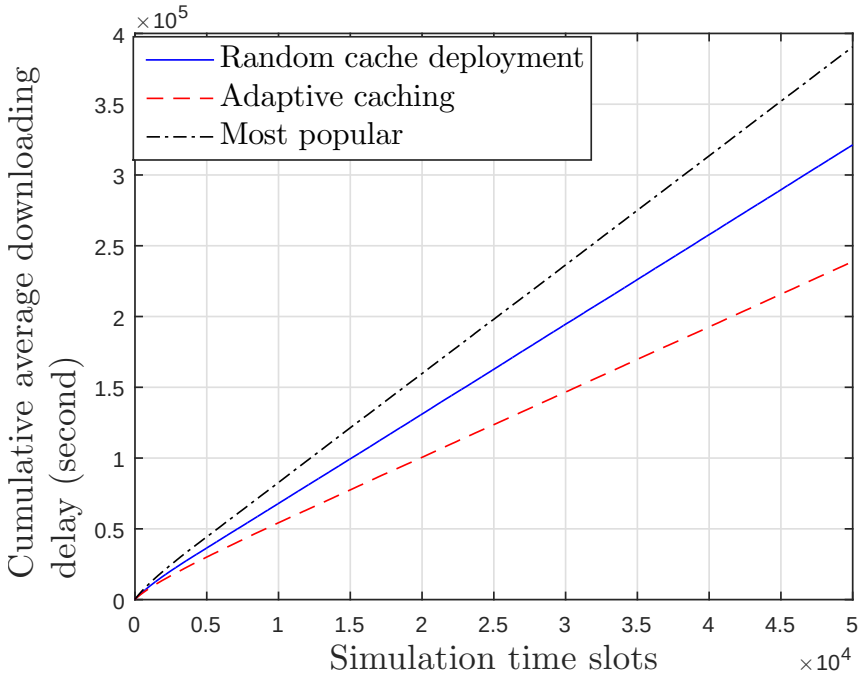

Figure 3.10: Cumulative average content downloading delay vs. simulation time. The figure illustrates that lower content downloading delay improves users' QoE.

Fig. 3.11 plots the cumulative average cache hit ratio in the network vs. the simulation time. As seen in Fig. 3.11, the adaptive caching scheme performs better than the other two caching schemes. This performance improvement is due to the fact that the adaptive caching scheme takes into account network topology, content popularity and cache deployment in the formulation. Here, higher cache hit ratio in the network means, a higher number of requests are being served by the connected BSs or by the neighbour BSs. As can be seen from Fig. 3.11, cache hit ratio for adaptive caching scheme is 0.9. That means only 10% of the content requests are served by the content server while *random cache deployment* and *most popular* caching schemes account for 15% and 25%, respectively for the given simulation setup. Therefore, the adaptive caching scheme reduces network traffic since fewer requests are served from the content server.

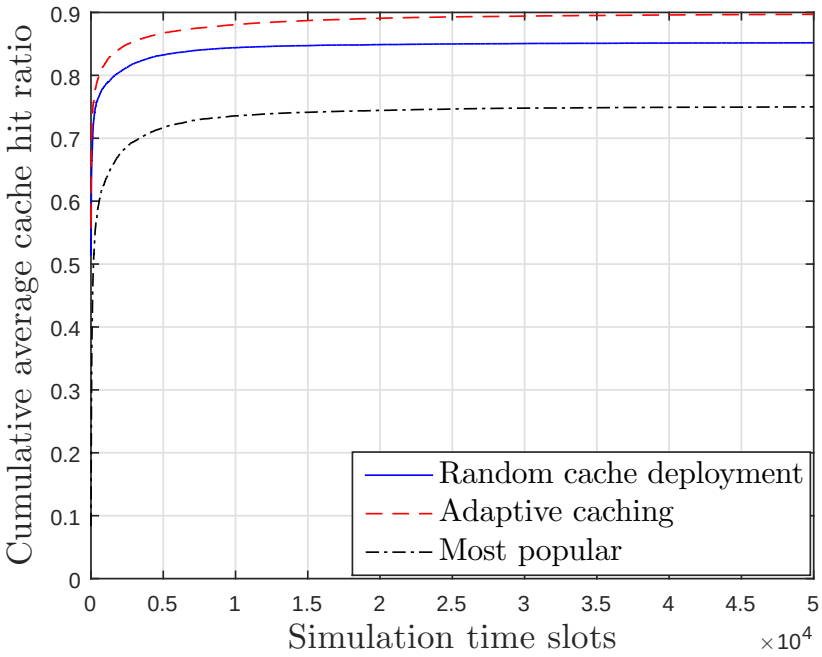

Figure 3.11: Cumulative average cache hit ratio in the network vs. simulation time. The figure illustrates that a higher cache hit ratio reduces the overall network traffic since fewer requests are served by transferring the content from the cache manager to the BS where the request originated.

3.5 Chapter Summary

In this work an adaptive caching scheme is presented that takes into account users' behaviour and operating characteristics of the cellular network. The caching scheme uses an optimized extreme learning machine to estimate the popularity of content based on users' behaviour, features of the content, and request statistics from users as they become available. The features of the content are computed using a combination of human perception models and network parameters. The estimates are used in a mixed-integer linear program which takes into account the cellular network parameters (e.g., network topology, communication link, and routing strategy) to select where to cache content and also to provide storage rec-

ommendations to the network operator. The scheme is validated using real-world data from YouTube and the NS-3 simulator.

Chapter 4

Risk-Averse Caching Scheme for Heterogeneous Networks

In this chapter, we describe risk-averse caching schemes for the heterogeneous wireless network. Recall from chapter 1, the caching schemes presented in this chapter not only consider content popularity and network parameters but also account for the routing protocol in the caching decision to ensure balancing the load throughout the network. In addition, the caching schemes take into account uncertainty associated with the predicted content requests. As a result, the caching schemes are formulated as risk-averse schemes using the CVaR measure instead of minimizing expected downloading delay as presented in chapter 3.

In this chapter, we propose four caching schemes for heterogeneous wireless network. The chapter is organized as follows. The system model of the heterogeneous wireless network is provided in Sec.4.1 where Table 4.1 provides a summary of the parameters used throughout the chapter. In Sec.4.2 dynamic cache replacement schemes are discussed. In Sec.4.3 we discuss four static caching schemes, two which are risk-neutral and two which are risk-averse. Specifically, in Sec.4.3.1 and 4.3.2 we construct risk-neutral static caching schemes for the heterogeneous network. The term risk-neutral is used to indicate that these methods use point estimates of the content requests–that is, they do not account for the

uncertainty associated with estimating the future content requests. An useful outcome of Sec.4.3.2 is that the static caching scheme, which accounts for content requests, cache size, bandwidth, load, and content routing, only requires the solution to a unimodular linear program. In Sec.4.3.4 and 4.3.5, risk-averse caching schemes are constructed based on the risk-neutral caching schemes in Sec.4.3.1 and 4.3.2. The main idea is that the uncertainty associated with estimating content requests is accounted for using the CVaR measure. In Sec.4.4 a novel content request conformal prediction algorithm is constructed based on the extreme learning machine and CVaR optimization. The performance of the risk-neutral and risk-averse caching schemes are evaluated using real-world data from the YouTube social network in Sec.4.5. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 25% reduction in average delay is achieved if both the uncertainty and routing protocol are accounted for compared to the risk-neutral caching scheme that neglects the routing protocol. Therefore, it is essential to account for both the uncertainty of predicting the content requests and routing when performing caching decisions.

4.1 System Model

In this section we introduce the system model of the heterogeneous wireless network and introduce the mathematical notation that will be used throughout the chapter to formulate the risk-neutral and risk-averse caching schemes.

We consider a heterogenous LTE wireless network that contains wireless nodes (e.g. base stations, smallcell access points, smallcell gateway) and a core network as illustrated in Fig. 4.1. The nodes in the network are defined by the set $\mathscr{V} = \{1, \ldots, V\}$. Each wireless node $v \in \mathscr{V}$ contains a physical cache of size S_v which stores the cached content. Additionally, the bandwidth between nodes is given by the parameter $b_{ij} \in \mathbb{R}_+$ for $i, j \in \mathscr{V}$. If the nodes i and j have no direct communication link then their bandwidth $b_{ij} = 0$. The bandwidth between nodes in the LTE network are typically heterogeneous as they are composed of

Table 4.1: Notation for Risk-Averse Caching

Parameters	Definition
$F(\cdot)$	cumulative distribution function
$\widehat{F}(\cdot)$	empirical cumulative distribution function
$p(\cdot)$	probability mass function
α	confidence level
t	time
Network Parameters	
$\mathscr{V}$	set of nodes $\{1,\ldots,V\}$
$\mathscr{V}_d$	destination nodes with $\mathscr{V}_d \subseteq \mathscr{V}$
S_v	cache size of node $v \in \mathscr{V}$
$C(t)$	cached content indicator matrix
$c_v(t)$	cached content indicator vector for node $v \in \mathscr{V}$
$n_v(t)$	load at node $v \in \mathscr{V}$
q_v	the request-queue-time at node $v \in \mathscr{V}$
A_{ijf}	weight between nodes $i,j \in \mathscr{V}$ for $f \in \mathscr{F}$
b_{ij}	bandwidth between nodes $i,j \in \mathscr{V}$
l_{ij}	latency between nodes $i,j \in \mathscr{V}$
δ_{ijdf}	shortest-path indicator for $i,j,d \in \mathscr{V}$ and $f \in \mathscr{F}$
$d_{vf}(t)$	content retrieval delay at $v \in \mathscr{V}_d$ for $f \in \mathscr{F}$
Content Parameters	
$\mathscr{F}$	set of content $\{1,\ldots,F\}$
$\mathscr{D}(t)$	dataset of content features and requests at time t
$\mathscr{G}$	content groups $\{1,\ldots,G\}$
f	content index $f \in \mathscr{F}$
s_f	size of content $f \in \mathscr{F}$
$g_{vf}(t)$	group association of content $f \in \mathscr{F}$ at $v \in \mathscr{V}_d$
$y_{vf}(t)$	request count for content $f \in \mathscr{F}$ at $v \in \mathscr{V}_d$
x_f	feature vector of content $f \in \mathscr{F}$
$\widehat{g}_{vf}(t)$	estimated group association of content $f \in \mathscr{F}$ at $v \in \mathscr{V}_d$
$\widehat{y}_{vf}(t)$	estimated request count for content $f \in \mathscr{F}$ at $v \in \mathscr{V}_d$
μ_g	mean vector of group $g \in \mathscr{G}$
Σ_g	covariance matrix of group $g \in \mathscr{G}$
β	neuron weights
θ	neuron transfer function parameters
L	number of neurons

wired, fiber-optic, and wireless links [140]. For example, the bandwidth between base stations and smallcell gateway nodes to the core network are on the order of several GB/s (fiber-optic). The link capacity of base stations and smallcell access points to mobile users are typically on the order of 1-100 MB/s (wireless). And the link capacity of smallcell access points to the smallcell gateways are on the order of 100 MB/s (wired connection). The content server stores all the content that can be requested by users. The core network communicates with the content server over the wide area network. Note that the content server is typically comprised of a commercial content distribution network (CDN) such as Akamai, Amazon CloudFront, Azure CDN or dedicated telco CDN that is maintained by a wireless network operator.

When a mobile user connects to the network, the network protocol established the communication link between the mobile user and either a smallcell access point or base station based on the minimal signal-to-interference-plus-noise ratio of the wireless channel. The set of content that can be requested by mobile users is denoted by $\mathscr{F} = \{1, 2, \cdots, F\}$. The size of each content is denoted by $s_f \in \mathbb{R}_+$ for $f \in \mathscr{F}$. When a smallcell access point or base station receives a user request for content $f \in \mathscr{F}$, the node that receives the request will retrieves the content from the network. The set of nodes (smallcell access points and base stations) that receive users' requests is denoted by $\mathscr{V}_d$ where $\mathscr{V}_d \subset \mathscr{V}$. The delay between when the user request was received and when the content is delivered to the user is known as the content retrieval delay. The content retrieval delay for content $f \in \mathscr{F}$ requested at node $v \in \mathscr{V}_d$ at time t is denoted by $d_{vf}(t)$. The content retrieval delay $d_{vf}(t)$ depends on where the content is cached in the network, the load of each node, bandwidth between nodes, network-layer protocol, and link-layer protocol of the LTE network. The load $n_v(t)$ of each wireless node $v \in \mathscr{V}$ is the total number of content requests the wireless node is currently processing. If a user requests content $f \in \mathscr{F}$ from the wireless node $v \in \mathscr{V}_d$ and node v has the content cached, then the content retrieval delay $d_{vf}(t) = s_f q_v n_v(t)$ where q_v is the request-queue-time of node v and s_f is size of the content f. The request-queue-

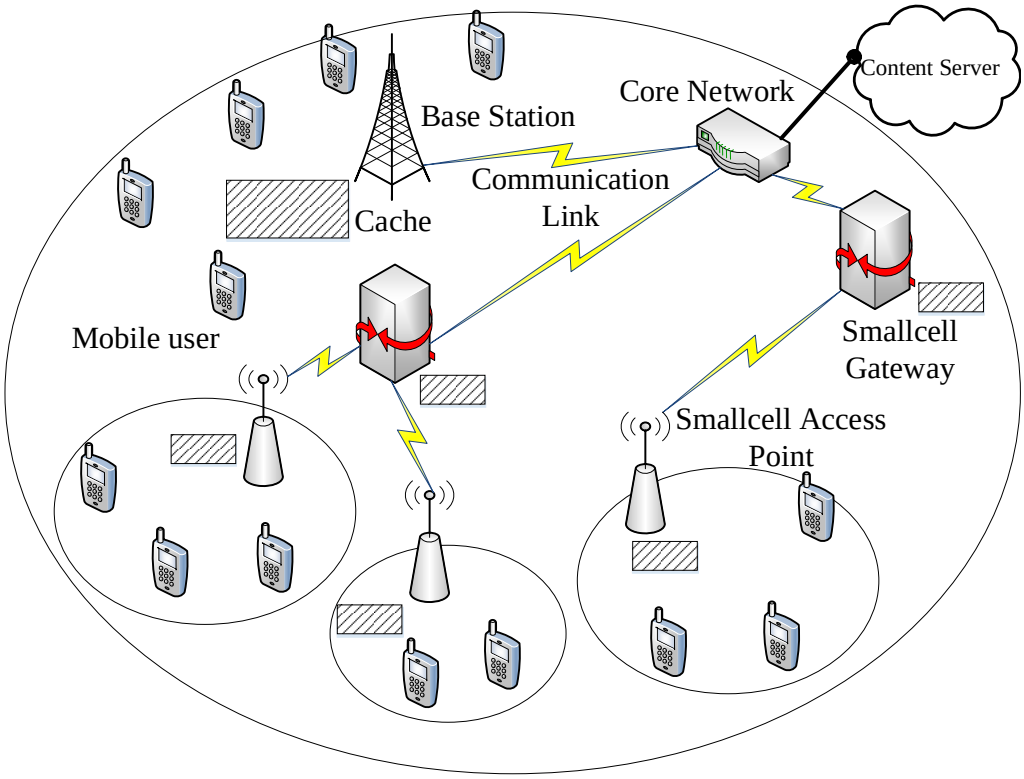

Figure 4.1: Schematic of an LTE wireless network. The wireless network is composed of smallcell access points, base stations, smallcell gateways, and a core network. The smallcell access point, smallcell gateway, and base stations all contain physical caches that can store content. Mobile users are connected to either the base station or the smallcell access point through a low-bandwidth connection. Smallcell access points are connected to the smallcell gateway which is then connected to the core network. Note that the base stations and smallcell gateway nodes do not communicate directly with each other. The core network is connected to the content server over the wide area network.

time q_v of node $v \in \mathcal{V}$ provides the average time to process a single packet/byte request. Note that if node v does not contain the requested content, then it must be retrieved by another node in the network or from the content server.

The goal of the network is to minimize the total content retrieval delay to serve

all user requests in the network. To achieve this objective, each node in Fig.4.1 contains a physical cache and a cache manager. The cache manager of each node controls the content that is cached at the node [66]. The currently cached content at node $v \in \mathscr{V}$ is given by the cached content indicator vector $c_v(t) \in [0,1]^F$ where F is the total number of contents that users can request in the network. The cache of each node is composed of a static segment and a dynamic segment. The content in the static cache does not change for a time interval ΔT and is associated with the slow-time scale caching decisions. The content in the dynamic cache changes as a function of the user requests and is associated with the fast-time scale. The cache manager controls the cache replacement scheme for the static and dynamic caches to minimize content retrieval delay $d_{vf}(t)$. Specifically, the cache manager at each node:

- runs the cache replacement scheme to minimize request delays.
- forwards content requests to neighbouring nodes or the content server if the content is not cached locally.
- records the number of requests $y_{vf}(t)$ and feature vector x_f for content $f \in \mathscr{F}$ at $v \in \mathscr{V}$.

It is assumed that each node in the network has computational resources equivalent to a standard desktop computer to allow the operation of the cache manager. Given that the cache size S_v, only a small fraction of all the content $\mathscr{F}$ can be cached locally (except for the content server which contains all contents). If the content is not cached locally, the content is retrieved from another node that minimizes the content retrieval delay $d_{vf}(t)$.

To perform cache replacement of the static cache, the cache manager records the request statistics $y_{vf}(t)$ and feature vector x_f for content $f \in \mathscr{F}$ at node $v \in \mathscr{V}$. For video content, the feature vector comprises information related to the thumbnail and title of the video, as well as information regarding the user that uploaded the video such as number of subscribers. The complete set of content

features and requests at each node is contained in the dataset

$$\mathscr{D}(T) = \Big\{ \{x_f, y_{vf}(t), g_{vf}(t)\} : v \in \mathscr{V}, f \in \mathscr{F}, t \in [0,T] \Big\} \tag{4.1}$$

where T is the total time the content $\mathscr{F}$ has been available to users. The parameter $g_{vf}(t)$ in (4.1) is the group association of content $f \in \mathscr{F}$ at node $v \in \mathscr{V}$. The possible groups that content can be associated with is denoted by $\mathscr{G} = \{1, 2, \ldots, G\}$. The cache manager uses the information in $\mathscr{D}(T)$ to estimate the future number of requests of content for both new content and previously cached content that users have requested.

Given the network parameters (cache size, load, and bandwidth between nodes) and the content parameters (feature vector, request count, group association), the aim is to design caching schemes to minimize the cumulative content retrieval delay

$$d(T) = \sum_{k=1}^{K_t} \sum_{v=1}^{V} \sum_{f=1}^{F} d_{vf}(t_k) \tag{4.2}$$

where K_t is the total number of content requests in the time interval $[0,T]$, and $t_k \in [0,T]$ denotes the time of each of the $k \in \{0,1,\ldots,K_t\}$ content requests. To minimize the delay requires a method to estimate the future content requests, and a method to cache popular content based on the request estimates and routing protocol to deliver content to users in the network. In this work we construct a content request density forecasting method, and both risk-neutral and risk-averse caching schemes to minimize the network delay.

4.2 Dynamic Caching Schemes

The cache of each node in the network, illustrated in Fig. 4.1, is composed of a dynamic segment and a static segment. This section discusses dynamic caching schemes, while Sec.4.3 discusses static caching schemes. To give more perspective, recall from Sec.4.1 that the content in the dynamic segment changes as a function of the user requests, while the content in the static segment changes on

a time interval ΔT that is significantly larger than the time-scale of individual content requests. The content in the dynamic and static segments of the cache are controlled by the dynamic caching scheme and static caching scheme respectively. Here we briefly discuss dynamic caching schemes that can be used in the network. In Sec.4.3 we present static caching schemes that are used in combination with the dynamic caching schemes presented in this section.

A schematic of the interaction of the dynamic and static cache is illustrated in Fig. 4.2. There are three possible scenarios that can occur depending on where the requested content is cached in the network. In the first scenario Fig. 4.2, the content is transferred from the static cache to the user and no change in the dynamic cache occurs. In the second scenario in Fig. 4.2, the content is transferred from the dynamic cache to the user. Additionally, the content in the dynamic cache will be adjusted according to the dynamic cache replacement scheme. In the third scenario in Fig. 4.2, the content must first be transferred to the dynamic cache from another node in the network, and then transferred to the user. Additionally, the content in the dynamic cache will be adjusted and evicted according to the dynamic cache replacement scheme. In all of the three scenarios, the content in the static cache remains unchanged, and identical content is not simultaneously available in both the static and dynamic caching segments. The content in the static cache is only updated at a time interval ΔT after it was first initialized. The aim of the static caching segment is to store content that is expected to have a large number of user requests in the duration ΔT, while the dynamic cache stores other content requested by users.

Several dynamic caching schemes exist which are based on users' real-time content requests including: Least-Recently-Used (LRU), Segmented Least-Recently-Used (SLRU) [114], Least-Frequently-Used, and Least-Frequently-Used with Dynamic Ageing and Adaptive Replacement Cache. The LRU cache replacement scheme operates by maintaining an ordered cache where recently requested content will reside at the beginning of the cache (known as most recently used position). As user requests are processed by the node, the content in the dynamic

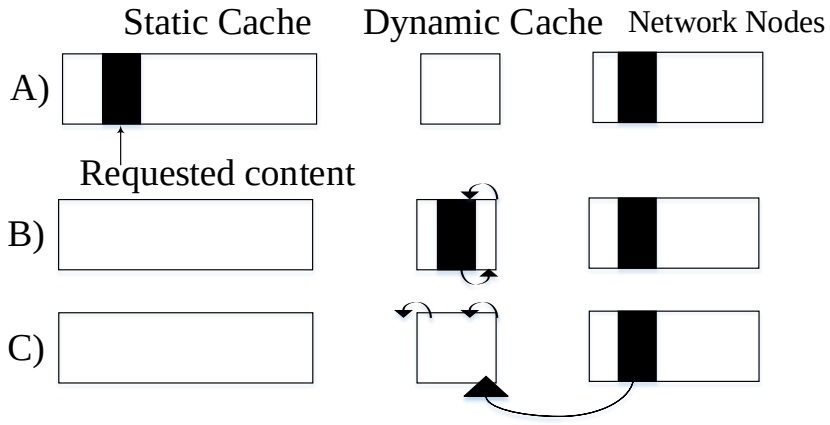

Figure 4.2: Schematic of the interaction between static and dynamic caching schemes. Three possible scenarios can be resulted from a content request. Scenario (A) refers to the situation when the requested content is available in the static cache. In this case, content in the static and dynamic cache remain unchanged. B) refers to the case when the requested content is available in the dynamic cache. In this case, content in the dynamic cache are adjusted according to the dynamic cache replacement scheme and content in the static cache remain unchanged. C) represents the situation when the requested content is unavailable in the cache. In this case, the content will be retrieved from the network nodes and stored in the dynamic cache. Other content in the dynamic cache will be adjusted and one content in the dynamic cache will be evicted. Content in the static cache remain unchanged.

cache that are not requested by users are shifted towards the end of the cache (least recently used position). If the requested content is not currently available in the cache, it will be retrieved from other nodes in the network and stored in the most recently used position. All other content in the cache will be shifted towards the end of the cache with the content in the least recently used position being evicted from the cache. Variants of the LRU scheme commonly used include the SLRU. In the SLRU cache replacement scheme, the entire cache is divided into different segments with each segment associated with a priority or popularity level. When requested content is unavailable in the cache, the content will be retrieved and

stored in the most recently used position of the lowest priority segment of the dynamic cache. Simultaneously, the content that is in the least recently used position of the lowest priority segment will be evicted from the cache. The main advantage of the LRU and SLRU dynamic cache replacement policies is that they do not require knowledge of the network architecture or where the content is stored throughout the network. As such, these caching schemes are straightforward to implement and are widely used in commercial distribution networks such as Facebook [114].

4.3 Risk-Neutral and Risk-Averse Static Caching Schemes

This section presents four static caching schemes and constitutes the main contribution of the chapter. Both risk-neutral and risk-averse caching schemes are discussed.

The static cache replacement scheme controls the content stored in the static cache of each node in the network illustrated in Fig.4.1. The content in the static cache remains the same for a time interval ΔT that is significantly longer than the characteristic time-scale of individual content requests from users. The goal of the static caching scheme is to cache content that is predicted to have a large number of requests in order to minimize the total content retrieval delay in the time interval ΔT. Given the parameters of the network and the content dataset $\mathscr{D}$, this section presents two risk-neutral and two risk-averse static caching schemes as illustrated in Fig. 4.3. The term risk-neutral is used for any scheme that uses point estimates of the content requests and do not account for the uncertainty associated with predicting the content requests. These include the risk-neutral (RN) and risk-neutral and network-aware (RNNA) caching schemes. Risk-averse caching schemes in contrast to the risk-neutral schemes, account for the uncertainty associated with estimating the content requests y_{vf} for content f at node v. These include the risk-averse (RA) and risk-averse and network-aware (RANA) caching schemes. The RA and RANA schemes can be viewed as generalizations of the RN and RNNA

schemes. Here, the uncertainty associated with the content requests is accounted for using the coherent CVaR risk measure with a confidence level $\alpha \in [0,1]$. Note that if we do not consider risk (e.g. risk-neutral) then the confidence level $\alpha = 0$.

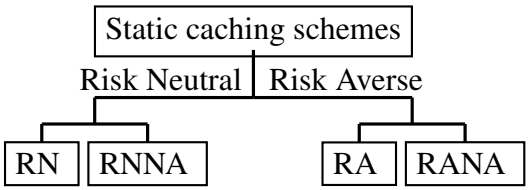

Figure 4.3: A schematic illustration of the risk-neutral (RN), risk-neutral and network-aware (RNNA), risk-averse (RA) and the risk-averse and network-aware (RANA) caching schemes discussed in Sec.4.3. RN and RNNA use point estimates of the content requests and do not account for the uncertainty associated with predicting the content requests. RA and RANA account for the uncertainty associated with estimating the content requests y_{vf} for content f at node v. The two network-aware caching schemes RNNA and RANA account for the LTE network parameters such as bandwidth, load at the nodes, request-queue-time, network-layer protocol, link-layer protocol and routing protocol in contrast to the network oblivious RN and RA caching schemes.

4.3.1 Risk-Neutral (RN) Static Caching Scheme

Let us assume that we have the predicted number of requests for each content $f \in \mathscr{F}$ at node $v \in \mathscr{V}$, which is denoted by $\widehat{y}_{vf}$. Then, the content to be cached at each node can be selected by solving the following binary integer program

$$C^* \in arg\,min_{c_{vf}} \left\{ \sum_{f=1}^{F} \sum_{v=1}^{V} \widehat{y}_{vf}(1 - c_{vf}) \right\}$$
$$\text{s.t.} \sum_{f=1}^{F} s_f c_{vf} \leq S_v \quad \text{for } v \in \mathscr{V}, \tag{4.3}$$

where $C^* \in [0,1]^{V \times F}$ and $c_{vf} \in [0,1]$ indicates if the content $f \in \mathscr{F}$ is cached at node $v \in \mathscr{V}$. The inequality constraint in (4.3) ensures that each node does not cache more files than can be stored in each nodes associated cache. Although (4.3) is a binary integer program which has complexity NP-complete, (4.3) can be solved with complexity $O(F\log(F))$ as each node merely caches the maximum number of content that are predicted to have the highest number of requests.

The RN caching scheme (4.3) is used extensively in the literature [26, 30, 66]. The key feature of the risk-neutral caching scheme is that it require an accurate estimation of y_{vf}, namely, the number of requests for the content. The RN scheme does not account for any aspects of the network other than the cache size of each node. Additionally, (4.3) does not account for the uncertainty associated with estimating the number of content requests y_{vf}. Therefore, although (4.3) can be evaluated with low complexity $O(F\log(F))$, the total content retrieval delay is expected to be higher compared with static caching schemes that account for the network parameters, routing protocol, and the uncertainty associated with predicted content requests.

4.3.2 Risk-Neutral and Network-Aware (RNNA) Static Caching Scheme

Here we construct RNNA static caching scheme to optimally cache content given the predicted content requests $\widehat{y}_{vf}$, and the network parameters (bandwidth, load at the nodes, request-queue-time, and cache size of each node). The RNNA caching scheme accounts for both the network parameters and routing protocol, however neglects the uncertainty associated with predicted content requests $\widehat{y}_{vf}$.

The (RNNA) static caching scheme is given by the following binary integer

program

$$
\begin{aligned}
&C^* \in arg\,min_{c,k,\delta,r} \left\{ \sum_{f=1}^{F} \sum_{d\in\mathscr{V}_d} \sum_{i,j\in\mathscr{V}} \widehat{y}_{df} A_{ijf} \delta_{ijdf} \right\} \\
&\text{s.t. } c_{sf} \in [0,1], \quad k_{sdf} \in [0,1], \quad \delta_{ijdf} \in [0,1], \\
&\quad r_{sdf} \in [0,1], \quad T_{ij} \in \mathbb{Z}_+ \\
&\quad \sum_{i\in\mathscr{V}} \delta_{sidf} - \delta_{isdf} = k_{sdf}, \quad \sum_{i\in\mathscr{V}} \delta_{didf} - \delta_{iddf} = -1, \\
&\quad \mathbf{1}\{b_{ij}=0\} + \delta_{ijdf} \leq 1 \qquad (4.4a) \\
&\quad \sum_{f=1}^{F} s_f c_{sf} \leq S_s, \quad \sum_{s=1}^{V} c_{sf} \geq 1 \qquad (4.4b) \\
&\quad \sum_{s=1}^{V} k_{sdf} = 1, \quad \sum_{s=1}^{V} r_{sdf} = 1, \\
&\quad \sum_{f=1}^{F} \sum_{d\in\mathscr{V}_d} \delta_{ijdf} \leq T_{ij} \qquad (4.4c) \\
&\quad r_{sdf} \leq k_{sdf}, \quad r_{sdf} \leq c_{sf}, \quad r_{sdf} \geq k_{sdf} + c_{sf} - 1, \qquad (4.4d) \\
&\quad \forall s \in \mathscr{V}, \quad \forall d \in \mathscr{V}_d, \quad \forall f \in \mathscr{F}.
\end{aligned}
$$

In (4.4), $C^* \in [0,1]^{V\times F}$ indicates the content cached in the static cache of all nodes, $\mathscr{V}_d \subset \mathscr{V}$ are the destination nodes in the network, and T_{ij} is a positive integer that indicates the maximum number of content transfer paths allowed between nodes i and j (e.g. congestion threshold). The destination nodes $\mathscr{V}_d$ communicate directly with the users and are comprised of the base stations and smallcell access points illustrated in Fig. 4.1. Given the predicted content requests $\widehat{y}_{df}$, the objective function in (4.4) represents the total content retrieval delay over the time-interval $[t, t+\Delta T]$. Note that we have dropped the time-dependence from the parameters in (4.4) to improve readability. The parameter A_{ijf} in (4.4) is the edge weight of

the network for content $f \in \mathscr{F}$ and is equal to

$$A_{ijf} = \begin{cases} s_f(l_{ij} + q_j n_j) \text{ if } i \neq j \\ s_f q_j n_j \quad \text{otherwise} \end{cases} \tag{4.5}$$

where s_f is the size of content f, l_{ij} is the latency between nodes $i \in \mathscr{V}$ and $j \in \mathscr{V}$, q_j is the request-queue-time of node $j \in \mathscr{V}$. The latency l_{ij} (second per byte) between nodes is a function of the bandwidth b_{ij}, the network topology, the network-layer protocol, and the link-layer protocol used in the network. The parameter δ_{ijdf} indicates if nodes $i, j \in \mathscr{V}$ are used to transfer the content $f \in \mathscr{F}$ to the destination node $d \in \mathscr{V}_d$. k_{sdf} indicates if source node $s \in \mathscr{V}$ is used to retrieve content $f \in \mathscr{F}$ for the destination node $d \in \mathscr{V}_d$. The parameter $r_{sdf} = k_{sdf} c_{sf}$ indicates if the source node $s \in \mathscr{V}$, used by destination node $d \in \mathscr{V}_d$ to retrieve content f, currently has the requested content cached.

The RNNA caching scheme (4.4) optimally selects the cache C^* to minimize the total content retrieval delay while accounting for the network parameters and predicted content requests. Additionally, RNNA accounts for the shortest-path routing used to transfer content throughout the network to serve user requests. The path constraints (4.4a) ensures that the δ_{ijdf} defines the shortest-path from source node $s \in \mathscr{V}$ to destination node $d \in \mathscr{V}_d$ for transferring content $f \in \mathscr{F}$. The caching constraints (4.4b) ensure that the files cached at each node do not exceed the nodes cache size, and that atleast one instance of each content $f \in \mathscr{F}$ is cached in the network. The link congestion constraint (4.4c) ensures that the number of content transferred over the link between node $i \in \mathscr{V}$ and node $j \in \mathscr{V}$ satisfies the congestion threshold T_{ij}. The source constraints (4.4d) ensures that only one source node $s \in \mathscr{V}$ is used to transfer content f optimally to the destination node $d \in \mathscr{V}_d$. Additionally, the source constraints ensure that the source node s has the content f to be transferred to the destination node d.

The binary integer program (4.4) to be solved for the RNNA caching scheme contains a total of $(V + 2V^2 + V^3)F$ binary variables, $(3V + V^2)F$ equality constraints, and $\Big((1 + V + 2V^2 + V^3)F + V^2 + V\Big)$ inequality constraints. If each content has

approximately equal size, then the constraint matrix in (4.4) is a network matrix. As such, (4.4) is a unimodular linear program and can be solved with polynomial time complexity using interior-point numerical methods [141]. Although YouTube content is not of equal size, it can be broken into equal sized blocks. The reason is that typically YouTube content is of duration between 30 seconds to 5 minutes, with a user attention span of approximately 90 seconds [142]. If each YouTube video is broken into 30 second intervals, then the decision of where to cache these video blocks throughout the network can be solved in polynomial time using the optimization problem (4.4). The main challenge, in this case, is to estimate the popularity of individual 30 second intervals of a content. One simple way to handle this challenge is to estimate the popularity of a content and then assume that all the individual 30 second intervals of the content have same popularity. However, in practice, the first 30 second of a content may receive more attention from users that the last 30 second and vice versa. In such case, we need to design a sophisticated prediction method and out of the scope of this thesis. Please note that the optimization problem (4.4) of solving four content with 30 second intervals is identical to the problem of solving two content with 60 second intervals where 60 second is divided into two 30 second intervals.

Though the RNNA caching scheme (4.4) uses the network parameters, popularity of content, and content transfer protocols, it does not account for the uncertainty associated with estimating the request count y_{vf} for content f at node v. Therefore, we can not control how to account for the risk associated with any given caching decision. Here the risk can be viewed as a measure of the worst case content retrieval delay that results from a given caching decision.

4.3.3 Conditional Value-at-Risk (CVaR) and Content Retrieval Delay Minimization

The two risk-averse caching schemes RA and RANA both use the coherent CVaR risk measure to account for the uncertainty associated with predicting the content requests. Here, we precisely define the CVaR risk measure and how it accounts

for the uncertainty associated with predicting the content requests.

Let D be a random variable that denotes the total content retrieval delay in a time-interval $[t, t+\Delta T]$. Realizations of the total content retrieval delay are defined by d (4.2). The ability to compare random outcomes of D based on the confidence $\alpha \in [0,1]$ is crucial for accounting for the risk associated with the uncertainty of estimating y_{vf} when performing caching decisions.

How can we estimate the total content retrieval delay D for a given confidence level $\alpha \in [0,1]$ (where $\alpha = 1$ is completely risk-averse, and $\alpha = 0$ is risk-neutral). One possibility is to use the Value-at-Risk (VaR) risk measure

$$\text{VaR}_\alpha(D) = \min\{d \in \mathbb{R} : F_D(d) \geq \alpha\} \tag{4.6}$$

where $F_D(d)$ is the cumulative distribution function of D. Classically, $\text{VaR}_\alpha(D)$ was a popular method to estimate risk, however, it has several limitations [92, 93]. First, $\text{VaR}_\alpha(D)$ is difficult to optimize as it is non-convex and is a non-coherent measure of risk as it fails the sub-additive condition. Second, $\text{VaR}_\alpha(D)$ does not account for the properties of the distribution $F_D(d)$ beyond the threshold $\text{VaR}_\alpha(D)$.

To account for the uncertainty of estimating y_{vf} for computing D, we use the CVaR measure [92, 93] which is a coherent risk measure. That is, CVaR satisfies the following properties: it is positive homogeneous, sub-additive, monotonic, translation invariant with respect to first order stochastic dominance, and monotonic with respect to second order stochastic dominance. CVaR is the expected total delay given that we are in the $\alpha \in [0,1]$ confidence interval of the cumulative distribution $F_D(d)$. Formally, CVaR is given by

$$\begin{aligned} \text{CVaR}_\alpha(D) &= E_D[D | D \geq \text{VaR}_\alpha(D)] \\ &= \frac{1}{1-\alpha} \int_\alpha^1 \text{VaR}_\beta(D)\, d\beta. \end{aligned} \tag{4.7}$$

where E_D denotes the expectation with respect to the random variable D that repre-

sents the content retrieval delay. As seen from (4.7), $\text{CVaR}_\alpha(D)$ can be interpreted as the conditional expectation of D where the expectation is evaluated on the α-confidence portion of the tail distribution $F_D(d)$. If $\alpha = 0$, then $\text{CVaR}_\alpha(D) = E[D]$ is the expected value of D, and if $\alpha \to 1$, then $\text{CVaR}_\alpha(D) = \max\{D\}$ gives the maximum value of D.

Given the CVaR risk measure (4.7), to minimize the total content retrieval delay in a time-interval $[t, t+\Delta T]$ for a confidence level α the following optimization problem needs to be solved

$$C^* \in arg\,min_{z \in \mathscr{Z}} \{\text{CVaR}_\alpha(D)\}, \tag{4.8}$$

where z are the decision variables that affect the total content retrieval delay D, and $\mathscr{Z}$ are the associated constraints on the decision variables. C^* in (4.8) is the associated caching decision that minimizes the total content retrieval delay with a confidence of α. If $\alpha = 0$ in (4.8), then (4.8) is equivalent to the risk-neutral caching schemes discussed in Sec.4.3.1 and Sec.4.3.2. Here, we are interested in risk-averse caching schemes for evaluating (4.8) where $\alpha \in (0, 1]$.

The evaluation of (4.8) for general $\alpha \in (0, 1]$ is non-trivial as it requires an analytical expression for the cumulative distribution function $F_D(d)$ which is unknown. Recall that the random variable D representing the total content retrieval delay in the time-interval $[t, t+\Delta T]$ which depends on the network parameters, popularity of content, content transfer protocols, and the uncertainty associated with estimating the request count y_{vf} for content f at node v. Using Theorem 2 in [92], the optimization problem (4.8) can be represented by

$$C^* \in arg\,min_{z \in \mathscr{Z}, c \in \mathbb{R}} \{c + \frac{1}{1-\alpha} E_D[\max\{0, D-c\}]\} \tag{4.9}$$

where the expectation is taken with respect to the random variable D. Though the distribution $F_D(d)$ is unknown, if the distribution of the content requests $F_{Y_{vf}}(y_{vf})$ at node $v \in \mathscr{V}$ is known, then the objective function (4.9) can be approximated using Monte-Carlo integration techniques. That is, the expectation $E_D[\cdot]$ in (4.9)

is estimated using K independent and identically distributed samples of the content requests y_{vf} generated from the distribution $F_{Y_{vf}}(y_{vf})$. Additionally, the non-smooth operator $\max\{\cdot\}$ in the objective function (4.9) can be removed by introducing auxiliary parameters ξ_k. The approximate solution to (4.9) can be computed by solving:

$$
\begin{aligned}
C^* \in arg\,min_{z\in\mathscr{Z},c\in\mathbb{R}}\{c + \frac{1}{K(1-\alpha)}\sum_{\xi=1}^{K}\xi_k\} \\
\text{s.t.} \quad & \xi_k \geq D(z,\widehat{y}_{vfk}) - c \\
& \widehat{y}_{vfk} \sim F_{Y_{vf}}(y_{vf}) \text{ for } v \in \mathscr{V}, f \in \mathscr{F} \\
& \xi_k \geq 0, \quad \text{for } k \in \{1,\ldots,K\}.
\end{aligned} \tag{4.10}
$$

where $\widehat{y}_{vfk}$ represents the generated sample of the content requests for content f at node v for sample k.

The optimization problem (4.10) provides the basis for constructing the risk-averse static caching schemes in this chapter.

4.3.4 Risk-Averse (RA) Static Caching Scheme

Here we construct a risk-averse (RA) static caching scheme that accounts for the uncertainty associated with predicting the content requests y_{vf} for content f at node v. The caching method RA is a generalization of the caching method RN in Sec.4.3.1 that does not account for the uncertainty associated with estimating the content requests.

Given the conditional density function $F_{Y_{vf}}(y_{vf})$ of content requests, the CVaR optimization problem (4.10), and using the constraints in the popularity based

caching method (4.3), the RA caching scheme is defined by

$$C^* \in arg\,min_{c_{vf},c} \left\{ c + \frac{1}{K(1-\alpha)} \sum_{\xi=1}^{K} \xi_k \right\}$$

$$\text{s.t.} \quad c_{vf} \in [0,1], c \in \mathbb{R},$$

$$\xi_k \geq \sum_{f=1}^{F} \sum_{v=1}^{V} \widehat{y}_{vfk}(1-c_{vf}) - c, \tag{4.11a}$$

$$\widehat{y}_{vfk} \sim F_{Y_{vf}}(y_{vf}),$$

$$\sum_{f=1}^{F} s_f c_{vf} \leq S_v, \tag{4.11b}$$

$$\xi_k \geq 0, v \in \mathscr{V}, f \in \mathscr{F}, k \in \{1,\ldots,K\},$$

where $C^* \in [0,1]^{V \times F}$. The parameter K in (4.11) is the total number of the content requests generated from the distributions $F_{Y_{vf}}(y_{vf})$ for $v \in \mathscr{V}$ and $f \in \mathscr{F}$. Additionally, the parameter c in (4.11) represents the estimated Value-at-Risk (VaR) of the total content delay for a confidence α, and $\alpha \in (0,1]$ is the confidence level.

The RA caching scheme (4.11) is a mixed integer linear program that contains $(2K+V)$ inequality constraints, (FV) binary variables, $(K+1)$ real variables, and requires the generation of (VFK) samples from the distributions $F_{Y_{vf}}(y_{vf})$. The complexity of (4.11) is NP-hard, however several numerical methods exist which can be used to evaluate (4.11) including: Branch-and-bound, cutting planes, branch-and-cut, and branch-and-price [143]. The selection of the number of samples K to use is still an open research problem. However, we found that a reasonable performance is achieved using $K = 10,000$ samples.

Though RA caching scheme (4.11) accounts for the uncertainty associated with predicting the number of requests, it neglects the network parameters (network bandwidth, load at the nodes, request-queue-time, content cached at other nodes), and the protocol of the network.

4.3.5 Risk-Averse and Network-Aware (RANA) Caching Scheme

Here we construct RANA caching scheme that accounts for the uncertainty associated with predicting the number of requests, network parameters (network bandwidth, load at the nodes, request-queue-time, content cached at other nodes), and protocol of the LTE network.

Given the conditional density function $F_{Y_{vf}}(y_{vf})$ of content requests, the CVaR optimization problem (4.10), and using the constraints in (4.4), the RANA caching scheme is given by

$$
\begin{aligned}
& C^* \in arg\,min_{C,k,\delta,r,c} \left\{ c + \frac{1}{K(1-\alpha)} \sum_{\xi=1}^{K} \xi_k \right\} \\
& \text{s.t.} \quad c_{sf} \in [0,1], \quad k_{sdf} \in [0,1], \quad \delta_{ijdf} \in [0,1], \\
& \qquad r_{sdf} \in [0,1], \quad T_{ij} \in \mathbb{Z}_+, \quad c \in \mathbb{R}, \\
& \qquad \xi_k \geq \sum_{f=1}^{F} \sum_{d \in \mathscr{V}_d} \sum_{i,j \in \mathscr{V}} \widehat{y}_{df} A_{ijf} \delta_{ijdf} - c, \\
& \sum_{i \in \mathscr{V}} \delta_{sidf} - \delta_{isdf} = k_{sdf}, \quad \sum_{i \in \mathscr{V}} \delta_{didf} - \delta_{iddf} = -1, \\
& \mathbf{1}\{b_{ij} = 0\} + \delta_{ijdf} \leq 1 \\
& \sum_{f=1}^{F} s_f c_{sf} \leq S_s, \quad \sum_{s=1}^{V} c_{sf} \geq 1 \\
& \sum_{s=1}^{V} k_{sdf} = 1, \quad \sum_{s=1}^{V} r_{sdf} = 1, \\
& \sum_{f=1}^{F} \sum_{d \in \mathscr{V}_d} \delta_{ijdf} \leq T_{ij} \qquad (4.12a) \\
& r_{sdf} \leq k_{sdf}, \quad r_{sdf} \leq c_{sf}, \quad r_{sdf} \geq k_{sdf} + c_{sf} - 1, \\
& \forall s \in \mathscr{V}, \quad \forall d \in \mathscr{V}_d, \quad \forall f \in \mathscr{F}, \quad \xi_k \geq 0, \quad k \in \{1, \ldots, K\}.
\end{aligned}
$$

A description of each of the constraints in (4.12) is provided below (4.4). Note that

the RANA caching scheme (4.12) is equivalent to the RNNA caching scheme (4.4) if we do not account for the uncertainty associated with predicting the number of requests for content $f \in \mathscr{F}$ at node $v \in \mathscr{V}$ – that is, we set $\alpha = 0$ in (4.12).

The mixed integer linear program (4.12) contains a total of $(V + 2V^2 + V^3)F$ binary variables, $(2K + 1)$ real variables, $(3V + V^2)F$ equality constraints, and $\Big((1 + V + 2V^2 + V^3)F + V^2 + V + 2K + 1\Big)$ inequality constraints. As discussed in Sec.4.3.4, several numerical methods can be used to evaluate C^* in (4.12) for typical small networks that contain up to $V = 10$ wireless nodes and $F = 1000$ content.

Summary: In this section we constructed four static caching schemes, namely, RN in (4.3), RNNA in (4.4), RA in (4.11), and RANA in (4.12). The RNNA caching scheme accounts for content requests, cache size, bandwidth, load, and content routing, only requires the solution to a unimodular linear program. The unique feature of the risk-averse caching schemes RA and RANA compared to RN and RNNA is that they use a coherent risk measure to account for the uncertainty associated with predicting the content requests. The confidence level $\alpha \in [0, 1]$ in the RA and RANA methods can be used to network operator to select the level of risk when performing a caching decision. For example, setting $\alpha = 1$ result in the RA and RANA minimizing the maximum possible content retrieval delay in the network. To evaluate the caching decision using RA or RANA requires a conformal prediction of the content requests–that is, an estimate of the cumulative distribution function of the content requests. In Sec.4.4 we provide a conformal prediction algorithm for constructing the cumulative distribution function of the requests.

4.4 Content Request Cumulative Distribution Function Forecasting

The risk-neutral and risk-averse static caching schemes constructed in Sec.4.3 require a point estimate $\widehat{y}_{vf}$ or cumulative distribution function estimate $F_{Y_{vf}}(y_{vf})$ of the request count y_{vf} for content $f \in \mathscr{F}$ at node $v \in \mathscr{V}$. In this section we construct

a conformal prediction algorithm to estimate $F_{Y_{vf}}(y_{vf})$ given the dataset $\mathscr{D}(T)$ in (4.1). The key idea of the conformal prediction algorithm is to use discriminant analysis to perform coarse-grained prediction of the content requests. Then use a feed-forward neural network to perform fine-grained request estimation. For both the coarse-grained and fine-grained request estimates, the prediction interval of each is denoted by $P(g|x_f)$ and $F_{Y_f}(y_f|g,x_f)$ respectively where $g \in \mathscr{G}$ represents the group association, and Y_f is the random variable representing the number of requests for content $f \in \mathscr{F}$. Using the total-law of probability the cumulative distribution function of the content requests is

$$F_{Y_f}(y_f|x_f) = \sum_{g=1}^{G} F_{Y_f}(y_f|g,x_f)P(g|x_f), \tag{4.13}$$

for content $f \in \mathscr{F}$. Below we present the coarse-grained and fine-grained request prediction methods, and the density forecasting algorithm to evaluate (4.13).

4.4.1 Content Group Association Classifier

Given the dataset $\mathscr{D}(T)$ in (4.1), the goal is to construct a classifier that can assign content $f \in \mathscr{F}$ to a particular group $g \in \mathscr{G}$ and provide a confidence estimate of the group association. That is, we desire a classifier which learns the conditional probability mass function $P(g|x)$ of group association.

The elements of the content features x_f are commonly constrained to intervals on the real line. For example, for YouTube videos, the number of subscribers to the user that uploaded the video must be a positive number and the minimum length of a video is 1 second (15 seconds if ads are present). Let us assume that the feature vector x is a random variable which has a conditional probability density function given by a doubly truncated multivariate normal distribution

$$\begin{aligned} p(x|g) &= \mathscr{N}(\mu_g, \mu_g^-, \mu_g^+, \Sigma_g) \\ &\propto \exp\left(-\frac{1}{2}(x-\mu_g)'\Sigma_g^{-1}(x-\mu_g)\right) 1\{\mu_g^- \leq x \leq \mu_g^+\}. \end{aligned} \tag{4.14}$$

In (4.14), μ_g is the mean vector of features in group $g \in \mathscr{G}$, μ_g^- is the minimum value of the content features in group g, μ_g^+ is the maximum value of the content features in group g, and Σ_g is the covariance matrix of the features in group g. If the mean μ_g, lower and upper limits (μ_g^- and μ_g^+) on the feature vector, and covariance matrix Σ_g are known for each group $g \in \mathscr{G}$, and the prior probability of group association is $P(g)$, then the probability of content $f \in \mathscr{F}$ being associated with group $g \in \mathscr{G}$ is

$$P(g|x_f) = \frac{P(x_f|g)P(g)}{\sum_{r=1}^{G} P(x_f|r)P(r)}. \tag{4.15}$$

The prior probability $P(g)$ of group association can either be set to an uninformative prior (i.e. $P(g) = 1/G$), or to the population average with $P(g) = n_g/F$ where n_g is the number of content in $\mathscr{F}$ that are associated with group $g \in \mathscr{G}$.

Given the dataset $\mathscr{D}(T)$ in (4.1), how can the parameters $\mu_g, \mu_g^+, \mu_g^-, \Sigma_g$ in (4.14) be estimated? If $\mu_g^- = -\infty$ and $\mu_g^+ = +\infty$ (the feature vector x is unconstrained), then classical discriminant analysis methods can be used to estimate μ_g and Σ_g. These include Linear Discriminant Analysis, Factor-Based Linear Discriminant Analysis, Maximum Uncertainty Linear Discriminant Analysis, and Regularized Discriminant Analysis [144]. Typically, the content features x are contained on an interval of the real line such that $-\infty < \mu_g^-$ and $\mu_g^+ < +\infty$. Given μ_g^- and μ_g^+, the estimate of the mean μ_g and covariance Σ_g can be computed using Gibbs sampling, maximum likelihood estimation, and generalized method of moments [145]. Here use the maximum likelihood estimation method to estimate μ_g and Σ_g using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with box constraints to account for the limits of the feature vector x.

4.4.2 Risk-Averse Feed-foward Neural Network for Predicting Content Requests

In this section we construct the risk sensitive extreme learning machine that combines the benefits of the extreme learning machine with a risk-aware training method to predict content requests. The presented method is comprised of a

single-layer feed-foward neural network trained using a stochastic optimization algorithm that dynamically adjusts the number of neurons and weights to minimize the risk of over fitting with a confidence level $\alpha \in [0,1]$.

Classic Extreme Learning Machine

Given the dataset $\mathscr{D}(T)$ in (4.1), the goal is to construct a method to estimate the functional relationship between the content features x_f and the associated request count $y_f(t)$. A single-layer feed-foward neural network can be used to relate the content features x_f to the requests $y_f(t)$. Denoting $\widehat{y}_f(t)$ as the estimated request count given x_f, the feed-forward neural network is given by

$$y_f(t) = \sum_{i=1}^{L} \beta_i h_i(x_f;\theta_i) = \beta' h(x_f) \tag{4.16}$$

with $\beta = [\beta_1, \beta_2, \ldots, \beta_L]'$ are the weights of each neuron, and
$h(x_f) = [h_1(x_f;\theta_1), \ldots, h_L(x_f;\theta_L)]'$ the associated transfer function of each neuron. Popular transfer functions include the sigmoid, hyperbolic tangent, and Gaussian, however any non-linear piecewise continuous function can be utilized. The neuron weights β_i and θ_i in (4.16) are computed from the solution to

$$\theta^*, \beta^* \in \arg\min \left\{ \sum_{f=1}^{F} (y_f(t) - \widehat{y}_f(t))^2 \right\}. \tag{4.17}$$

For general transfer functions $h(x_f)$, (4.17) is a non-convex and non-linear optimization problem that is commonly solved using stochastic gradient decent, ant-colony optimization, and simulated anealling methods [146, 147]. A draw-back with these numerical methods for solving (4.17) is that they require a large number of computations to converge to the global optimal solution of (4.17). However, using random matrix theory, it has been shown that the parameter values at the local minima of the objective function (4.17) yield similar mean-square error compared to the global optimal solution of (4.17) [148]. Therefore, the mean-square er-

ror of the feed-forward neural network is not sensitive to the set of local minima $\{\theta^*, \beta^*\}$ is used.

Could selecting the neuron weights θ randomly, and then fitting the weights β via least-squares minimization provide a reasonable approximation to the solution of (4.17)? Since the mean-squared error of the feed-forward neural network is not sensitive to which local minima of (4.17) are used, it is reasonable to postulate that randomly selecting θ and then fitting β will produce a reasonable solution. This is the main idea behind the extreme learning machine (ELM) proposed in [124]. For fixed number of neurons L, the ELM selects the parameters θ and β in two steps. First, the hidden layer parameters θ are randomly initialized. Any continuous probability distribution can be used to initialize the parameters θ. Second, β is selected to minimize the mean-square error between the model output and the measured output from the dataset $\mathscr{D}(T)$ in (4.1). Formally,

$$\begin{aligned} \theta^* &\sim \mathscr{N}(0,1) \\ \beta^* &= H(X;\theta^*)^+ Y \end{aligned} \tag{4.18}$$

where $\mathscr{N}(0,1)$ is the multivariate normal distribution with unit variance, $H(X;\theta^*)$ is the hidden-layer output matrix with entries $H_{if}(X;\theta) = h_i(x_f;\theta_i)$ for $i = 1,\ldots,L$ and $f \in \mathscr{F}$, H^+ denotes the Moore-Penrose generalized inverse of H, and $Y = [y_1,\ldots,y_F]'$. Several efficient methods can be used to compute β^*, for example Gaussian elimination. The major benefit of using the ELM, (4.16) and (4.18), is that the training only requires the random generation of the parameters θ^*, and β^* is computed as the solution of a set of linear equations.

The ELM satisfies the universal approximation condition [120], can be implemented in parallel [117], can be trained sequentially for large datasets or as new training data becomes available [122, 123], and can be efficiently implemented on field-programmable gate array devices as well as complex programmable logic devices [118]. A limitation with the ELM is that it can not be used to select the number of neurons L while minimizing the risk of overfitting the dataset $\mathscr{D}(T)$ in

(4.1).

Regularized Extreme Learning Machine

Here we construct the regularized extreme learning machine (RELM) which is a generalization of the ELM that minimizes the risk of overfitting with a confidence level $\alpha \in (0,1]$. For example, given the dataset $\mathscr{D}(T)$ in (4.1) and setting $\alpha = 1$, the RELM selects the parameters θ, β, L in (4.16) to minimize the mean-square generalization error between the actual and predicted content requests. The RELM is equivalent to the ELM if we do not account for risk–that is, we set $\alpha = 0$.

The RELM is trained by solving the following optimization problem (we discuss the motivation below):

$$
\begin{aligned}
L^* \in & \, arg\, min_{L\in\mathbb{Z}_+, c\in\mathbb{R}} \{c + \frac{1}{K(1-\alpha)} \sum_{k=1}^{K} z_k\} \\
\text{s.t.} \quad & z_k \geq \frac{1}{(1-\gamma)F} ||\eta(L) H(\overline{X}_k; \theta_k)\beta_k - \overline{Y}_k||_2^2 - c && \text{(4.19a)} \\
& X_k, \overline{X}_k, \overline{Y}_k, Y_k \sim \mathrm{FY}(\mathscr{D}(T), \gamma) && \text{(4.19b)} \\
& \theta_k \sim \mathscr{N}(0,1), && \text{(4.19c)} \\
& \beta_k = [\eta(L) H(X_k; \theta_k)]^+ Y_k && \text{(4.19d)} \\
& z_k \geq 0, \quad L \leq L_{\max}, \quad \text{for } k \in \{1, \ldots, K\}. && \text{(4.19e)}
\end{aligned}
$$

In (4.19), $\mathscr{N}(0,1)$ is the multivariate normal distribution, $\mathrm{FY}(\mathscr{D}, \gamma)$ is the Fisher-Yates random permutation which is used to partition the data $\mathscr{D}(T)$ into complementary subsets of training data (X_k, Y_k) and validation data $(\overline{X}_k, \overline{Y}_k)$ where $\gamma \in (0,1)$ denotes the percentage of data used for training, and $\eta(L)$ is a diagonal matrix with elements

$$
\eta_{ii}(L) = \begin{cases} 1 \text{ if } i \leq L \\ 0 \text{ otherwise.} \end{cases} \tag{4.20}
$$

The parameter $L_{\max}$ is the maximum number of neurons in the RELM, and K is the number of samples used to estimate the cumulative distribution function of the

mean-square error of the feed-forward neural network.

The objective function in (4.19) represents the CVaR of the mean-square generalization error of the ELMs with L neurons for a confidence level α. The mean-square generalization error of each ELM is evaluated using

$$\frac{1}{(1-\gamma)F}||\eta(L)H(\overline{X}_k;\theta_k)\beta_k - \overline{Y}_k||_2^2$$

defined in constraint (4.19a). The constraint (4.19b) is used to generate the training (X_k, Y_k) and testing $(\overline{X}_k, \overline{Y}_k)$ data for each of the $k \in \{1,\ldots,K\}$ ELMs. The constraint (4.19c) is used to generate the neuron transfer function weights θ_k. Given the transfer function weights θ_k and the training data (X_k, Y_k), constraint (4.19d) is used to construct the neuron weights β_k of each ELM. The final constraint (4.19e) defines the maximum number of possible neurons $L_{\max}$ and the number of ELMs K generated to evaluate the CVaR risk measure. The selection of $L_{\max}$ and K are important to restrict the computational resources used to train and evaluate the RELM. The final result of the RELM (4.19) is the optimal number of neurons L^* of use for predicting the content requests while minimizing the risk of overfitting the training data with a confidence level α. Given L^*, the ELM weights θ, β can be constructed using equation (4.18).

Discussion of (4.19): Solving the mixed integer nonlinear program (4.19) is equivalent to optimally selecting the number of neurons L to minimize the risk of overfitting the dataset $\mathscr{D}(T)$. The training method in (4.19) dynamically adjusts the number of neurons L based on the available training data in $\mathscr{D}(T)$. Typically, as the number of observations in $\mathscr{D}(T)$ increases, the number of neurons in the RELM will also increase. The solution to (4.19) can be computed in polynomial time by solving (4.19) for $L = 1,\ldots,L_{\max}$ independently, and then selecting the solution that minimizes the objective function in (4.19).

4.4.3 Conformal Prediction Algorithm for Content Requests

In this section we construct the conformal prediction algorithm to estimate the cumulative distribution function $F_{Y_f}(y_f|x_f)$ of the number of requests for content $f \in \mathscr{F}$ given the content features x_f.

A schematic of the conformal prediction algorithm is provided in Fig.4.4. The algorithm is comprised of an offline training stage, and an online stage to evaluate the requests for new content. In the offline stage the dataset $\mathscr{D}(T)$ in (4.1) is used to train the group association classifier defined in Sec.4.4.1, and the RELM defined in Sec.4.4.2. Then, using the dataset

$$\widehat{D}(T) = \{\{x_f, g_f(t), y_f(t), \widehat{y}_f(t)\} : v \in \mathscr{V}, f \in \mathscr{F}, t \in [0,T]\}, \tag{4.21}$$

the set of prediction errors $\{\varepsilon_f(g)\}$, where $\varepsilon_f(g) = \widehat{y}_f(g,x_f) - y_f$ for each content $f \in \mathscr{F}$ associated with group $g \in \mathscr{G}$ is constructed. The trained parameters from the offline stage are then used in the online stage to estimate the request count of the content. In the online stage, when a new content with features x_o is received, the group association classifier is used to compute the group association probabilities $P(g|x_o)$. Then the RELM is used to estimate the number of requests $\widehat{y}_f(x_o, g)$ for the content f. The group association probabilities and predicted number of requests from the RELM are then sent to the conformal prediction block which outputs the conformal prediction $F_Y(y_o|x_o)$.

Insight into the design of the conformal prediction algorithm in Fig. 4.4 is gained by considering the content requests y_f for content f as a random variable. We assume that the random variable y_f satisfies

$$y_f = \widehat{y}_f(g,x_f) + \varepsilon_f(g) \tag{4.22}$$

where $\widehat{y}_f(g,x_f)$ is the estimated number of requests from the RELM trained using data from group $g \in \mathscr{G}$, and $\varepsilon_f(g)$ is the random variable that accounts for the error in the predicted and actual number of requests. The error $\varepsilon_f(g)$ accounts for the errors associated with the parametric structure of the RELM, the parameters of the

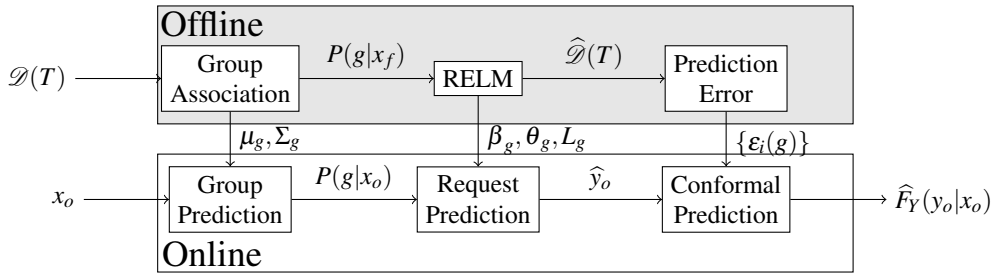

Figure 4.4: Schematic of the conformal prediction algorithm. $\mathscr{D}(T)$ in (4.1) is the training dataset, $P(g|x_f)$ is the probability of group association, $\widehat{D}(T)$ is the training dataset with the predicted content requests from the RELM, $\{\varepsilon_f(g)\}$ are the errors between the predicted and actual requests for content in group $g \in \mathscr{G}$, x_o is a new content feature, and $\widehat{F}_Y(y_o|x_o)$ is the empirical cdf function of the content given the content features x_o.

RELM, and the errors that may be contained in the feature vector x_f. The random variables $\varepsilon_f(g)$ are assumed independent and identically distributed. Then, given the dataset $\widehat{\mathscr{D}}$, the empirical cdf of the random variable y_f is

$$\widehat{F}_Y(y_f|g,x_f) = \frac{1}{n_g}\sum_{i=1}^{n_g} 1\left\{\varepsilon_i(g) \leq y_f - \widehat{y}_f(g,x_f)\right\} \tag{4.23}$$

where n_g is the total number of contents in group $g \in \mathscr{G}$ and $1\{\cdot\}$ is the indicator function.

Substituting (4.23) into (4.24) gives the conformal prediction of the content $f \in \mathscr{F}$. Formally, the empirical conditional distribution function of the number of requests for content $f \in \mathscr{F}$ is

$$\widehat{F}_Y(y_f|x_f) = \sum_{g=1}^{G} \widehat{F}_Y(y_f|g,x_f)P(g|x_f) \tag{4.24}$$

where $P(g|x_f)$ is the conditional probability of content f belonging to group $g \in \mathscr{G}$ from the discriminant analysis classifier.

Summary: In this section we constructed a conformal prediction algorithm,

illustrated in Fig. 4.4, for content requests. The conformal prediction algorithm is comprised of an offline learning stage in which content request and features are used to select the doubly-truncated multivariate normal distribution parameters for group association estimation, and train the RELM for constructing point estimates of the content requests. For new content, the output of the trained group classifier and RELM are used to construct the conformal prediction of the content requests. This conformal prediction is used with the risk-averse caching methods in Sec.4.3.4 and 4.3.5 to optimally cache content in the network.

4.5 Numerical Evaluation of the Conformal Prediction Algorithm and Coherent Risk Minimization Caching Schemes for YouTube Content

In this section we evaluate the performance of the conformal prediction algorithm and four static caching schemes (RN, RNNA, RA, RANA) presented in Sec.4.3 using real-world YouTube datasets. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 25% reduction in average delay is achieved if both the uncertainty and smallcell routing protocol are accounted for compared to the risk-neutral caching scheme that neglects the routing protocol. These results illustrate the importance of both accounting for the risk associated with estimating content requests and accounting for the routing protocol used to transfer content throughout the network.

4.5.1 Network Parameters and YouTube Dataset

To evaluate the performance of the static caching schemes (RN, RNNA, RA, RANA), and the conformal prediction method illustrated in Fig.4.4, we construct an LTE heterogeneous network and generate user content requests based on real-world YouTube datasets.

Network Parameters: The LTE network topology used for evaluation is illustrated in Fig.4.5. The LTE network is composed of a core network that is connected to base station nodes, smallcell gateway nodes, and a server which contains all the content $\mathscr{F}$ that can be requested by users. The content server and the core network communicate via the wide area network. The latency in the wide area network is typically in the range of 10 ms to 100 ms depending on the distance and the number of hops between the server and the core network [149, 150]. The core network, base station, and gateway nodes communicate with one another via an intra-network communication link with link capacity values in the range of 500 Mbps to 1 Gbps. Finally, the smallcell access points are connected to the gateway nodes via heterogeneous backhaul links with link capacity values in the range of 100 Mbps to 500 Mbps.

In the network illustrated in Fig.4.5, user requests are only received by the smallcell access points and the base station nodes. Each requested video content has a size of $s_f = 200$ MB. The smallcell access points, base station, and gateway have a cache size of 500 GB (approximately 10% of the entire content library), and the core network node has a cache size of 1 TB (approximately 20% of the entire content library). For each node, 90% of the cache storage is used for static caching, and the remaining is used for dynamic caching. The dynamic cache segment of each node is controlled using the Least-Recently-Used (LRU) replacement method discussed in [75]. Given the cache size, link capacity between the nodes, processing delay, propagation delay, number of hops between nodes, packet size, and the topology in the network in Fig.4.5, the edge weight parameter A_{ijf} (4.5) between node i and node j is evaluated using the ndnSIM 2.0 NS-3 base simulator [151]. ndnSIM allows a video content f to be addressable and routable inside the network. For the network-layer protocol, ndnSIM's NDN stack is used while *point-to-point* communication is considered as the link layer protocol. In ndnSIM, the smallcell access points and base station are defined as the *Consumer* node. We implement a new consumer application method for the *Consumer* node to generate content requests according to the YouTube datasets.

All the nodes having cache storage are considered as the *Producer* node which can satisfy content requests generated by the *Consumer* nodes. The ndnSIM *Best Route* is used to forward content requests to neighbouring nodes until the content is retrieved–this is equivalent to the shortest-path routing algorithm. Finally, the ndnSIM *Application-level trace helper* is used to compute the edge weight A_{ijf} (4.5) between node i and node j for content f. We set $T_{ij} = 16$ in equation (4.4c) and in equation (4.12a).

YouTube Dataset: The real-world YouTube dataset was collected using the YouTube API between the years 2013 to 2015 and consists of $25,000$ YouTube videos. The dataset is comprised of videos from 17 YouTube categories including "Gaming" (44% of videos), "Entertainment" (40% of videos), "Music" (5% of videos), and "Education" (2% of videos). Note that gaming and entertainment are among the most popular video categories on YouTube. The video requests range from 10^2 to above 10^7. Therefore, to prevent the ELM algorithm from biasing it's prediction to only the videos with the highest requests, we apply a log transform to the requests such that the range is in 2 to 7. All the content features are scaled to satisfy $x(m) \in [0,1]$ for $m \in \{1,\ldots,M\}$. We use $11,747$ video content to construct the training dataset $\mathscr{D} = \{\{x_f, g_f, y_f\} : f \in \mathscr{F}\}$. The remaining videos are used to test the performance of the conformal prediction algorithm. In total there are $G = 3$ groups in the dataset, where group $g = 1$ is associated with videos with less than 100 requests, $g = 2$ with videos with requests in the range of 100 to $30,000$ requests, and $g = 3$ videos that have more then $30,000$ requests. For testing the performance of the discriminant analysis classifier, extreme learning machines, and coherent risk-minimization algorithms, we use the dataset $\widehat{\mathscr{D}}$ which is comprised of $13,253$ videos. The number of requests for each video content is identical at each of the smallcell access points and base station.

As a preliminary step, the YouTube videos are clustered based on their associated category. The content requests of cluster $c \in \mathscr{C}$ at node v is:

$$y_c^v = \sum_{f \in F^c} y_{vf}, \tag{4.25}$$

where $F^c \subseteq \mathscr{F}$. Given the content requests per cluster, the caching methods presented in this chapter optimally cache each category of content in the network. We use the top 10 most popular YouTube categories to compute the performance metrics of the caching schemes. To evaluate the delay of the risk-averse caching schemes a method is required to generate user request that is consistent with the YouTube dataset $\overline{\mathscr{D}}$.

Content Request vector: We discretize time into a total of K_t time slots, where for each time-slot a content request is received. We construct content popularity distribution using dataset $\overline{\mathscr{D}}$ where P_f^v denotes the probability of content f being requested at node v and $P_f^v \sim y_{vf}$. Let r^v denote the content request vector with elements $r^v(k) \in \mathscr{F}$ that define the content f being requested time k at node v. The content request vector satisfies the condition

$$\sum_{k=1}^{K_t} 1\{r^v(k) = f\} \sim P_f^v \quad \forall f \in \mathscr{F}. \tag{4.26}$$

To construct the content request vector we randomly generate a total of $K_t = 20,000$ content requests in r^v such that (4.26) is satisfied. The estimated delay for a given caching decision is computed by evaluating the total delay associated with the content requests from the request vector r^v at each of the v nodes.

4.5.2 Conformal Prediction Algorithm for YouTube Content

In this section the performance of the conformal prediction algorithm presented in Sec.4.4, is evaluated using real-world YouTube datasets. This includes the performance of the discriminant analysis classifier, extreme learning machines, and the conformal prediction for video content requests.

From fig. 4.4, the first step in the conformal prediction algorithm is the offline stage in which the parameters of the group association classifier (μ_g and Σ_g) are selected, the extreme learning machines are trained for each group (β_g, θ_g, L_g), and the parameters $\varepsilon_f(g)$ in (4.23) are estimated using the training dataset $\mathscr{D}$. The performance of the group classifier is evaluated using the true positive rate

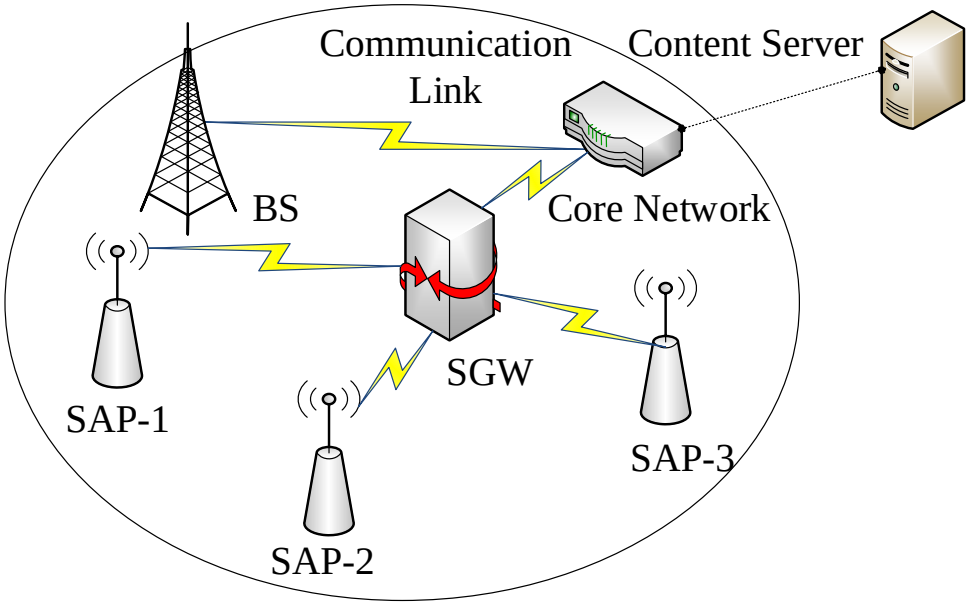

Figure 4.5: Schematic of the network. There are three smallcell access points (SAPs) in the network. These SAPs are connected with the smallcell gateway (SGW) via heterogeneous communication links. SGW and base station (BS) are connected with the core network. Each SAPs, BS and SGW has a storage of 500 GB. Storage of the core network is assumed to be 1 TB and the core network is connected to the content server which can cache the entire library via wide area network (WAN).

(TPR) and true negative rate (TNR), and the extreme learning machines using $\text{CVaR}_\alpha(\varepsilon^2)$ for an $\alpha = 0.95$. The results are provided in Table 4.2 where the parameter n_g is the total number of content in each of the groups $g \in \mathscr{G}$. As seen, the group association classifier has a reasonable TPR and TNR for classifying the group association of each video. Using the RELM in Sec.4.4.2, Table 4.2 illustrates that the selected number of neurons L_g for the ELM associated with group

$g \in \mathscr{G}$ varies between the three groups, however the error $\text{CVaR}_\alpha(\varepsilon^2)$ remains approximately equal. This illustrates that the RELM can dynamically adjust the number of neurons necessary to effectively estimate the content requests $\widehat{y}_f$ of new content while minimizing the risk of over-fitting with a confidence level $\alpha = 0.95$.

Table 4.2: Group Classification and Neuron Number Selection

Group	n_g	TPR	TNR	$\text{CVaR}_\alpha(\varepsilon^2)$	L_g
1	2405	0.80	0.96	0.3302	37
2	10314	0.94	0.79	0.3046	59
3	534	0.74	0.99	0.3133	24

To construct the conformal prediction (4.24) requires an estimate of the prediction errors $\varepsilon_f(g)$ in (4.23). Alternatively, we can construct an estimate of $\widehat{F}_Y(y_f|g,x_f)$ using the empirical cumulative distribution function $\widehat{F}_{E|g}(\varepsilon|g)$ of the random variables ε. Fig.4.6 illustrates the computed group empirical cumulative distribution function $\widehat{F}_{E|g}(\varepsilon|g)$ of the error of the ELM associated with each group $g \in \mathscr{G}$. Using maximum likelihood estimation, we find that the empirical cdf $\widehat{F}_{E|g}(\varepsilon|g)$ is approximately equal to the generalized extreme value distribution typically used in risk management, finance, and economics. Given $\widehat{F}_{E|g}(\varepsilon|g)$, the conditional distribution of content requests can be evaluated using

$$\widehat{F}_Y(y_f|g,x_f) = \widehat{F}_{E|g}(y_f - \widehat{y}_f(x_f,g)|g,x_f) \tag{4.27}$$

where $\widehat{y}_f(x_f,g)$ is the estimated content requests from the ELM associated with group $g \in \mathscr{G}$ for content f.

Given the parameters of the group association classifier, ELMs, and $\widehat{F}_Y(y_f|g,x_f)$ from the offline stage of the conformal prediction algorithm, we now evaluate the performance of the online portion of the conformal prediction algorithm for new content. The group association probability $P(g|x_f)$ and results of the conformal prediction algorithm for new content are provided in Fig.4.7. The content index

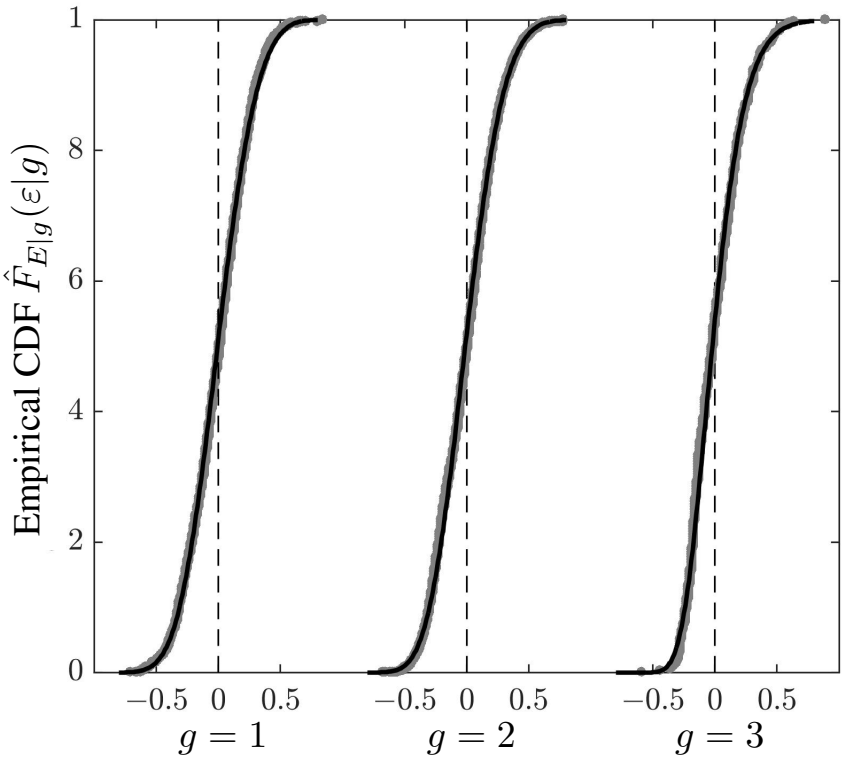

Figure 4.6: Empirical cumulative distribution function of the error $\varepsilon(g)$ in (4.22) for the groups $g \in \mathscr{G}$. The gray dots indicate the empirical cumulative distribution function $\widehat{F}_{E|g}(\varepsilon|g)$, and the black line indicates the fitted generalized extreme value distribution.

is ordered such that the least requested content is $f = 1$, and the most requested content is $f = 13,253$ based on the results of the conformal prediction. As Fig.4.7 illustrates, the group association probability $P(g|x_f)$ from the discriminant analysis classifier provides a reasonable accuracy for the probability of group association. From Fig.4.8, there are approximately $1,225$ contents (indicated in black) that have predicted number of requests from the ELMs that are outside the 90% confidence interval. Equivalently, approximately 9.2% of the content requests reside outside the 90% confidence interval computed from the conformal predic-

tion algorithm. To determine if the cumulative distribution function $\widehat{F}_Y(y_f|x_f)$ is consistent with observed content requests data, we use the quantile-quantile plot. The quantile-quantile plot in Fig. 4.9 is evaluated using the confidence interval of $\widehat{F}_Y(y_f|x_f)$. As seen from Fig. 4.9, the empirical cumulative distribution $\widehat{F}_Y(y_f|x_f)$ is in excellent agreement with the observed data. Therefore, $\widehat{F}_Y(y_f|x_f)$ provides a reasonable approximation for the actual content request distribution $F_Y(y_f|x_f)$.

The above results indicate that the conformal prediction algorithm presented in Sec.4.4, and illustrated in Fig. 4.4, can be used to estimate the cumulative distribution function $F_Y(y_f|x_f)$ of YouTube content requests. The estimated cumulative distribution function $\widehat{F}_Y(y_f|x_f)$ can then be used in the risk-averse caching schemes (RA and RANA) to optimally cache content in the smallcell network. Additionally, the trained ELMs can be used for point predictions of the content requests for the risk-neutral caching schemes (RN and RNNA).

4.5.3 Selection of the Confidence Level α for Maximum Content Retrieval Delay Guarantees

The risk-averse caching schemes (RA and RANA) account for the uncertainty of the predicted content requests using the CVaR risk measure for a given confidence level α. From (4.7), for a confidence level α, CVaR is the expected content retrieval delay given that the delay is greater than or equal to the VaR at α. The selection of the parameter $\alpha \in [0,1]$ is determined by the network operator. In this section we evaluate the cumulative distribution function $F_D(d)$ to evaluate the probability that the delay D is less than the threshold d using the RNNA and RANA caching schemes for a given confidence level α. Given $F_D(d)$, the network operator can select the confidence level α to guarantee that the delay does not exceed the given threshold d_{th}.

Fig. 4.10 provides the cumulative distribution function $F_D(d)$ for the content retrieval delay for confidence levels $\alpha = 0.9$ and $\alpha = 0.99$ using the RANA caching scheme, and the RNNA caching scheme. From Fig. 4.10, the RANA caching scheme (4.12) provides better performance compared to the RNNA caching

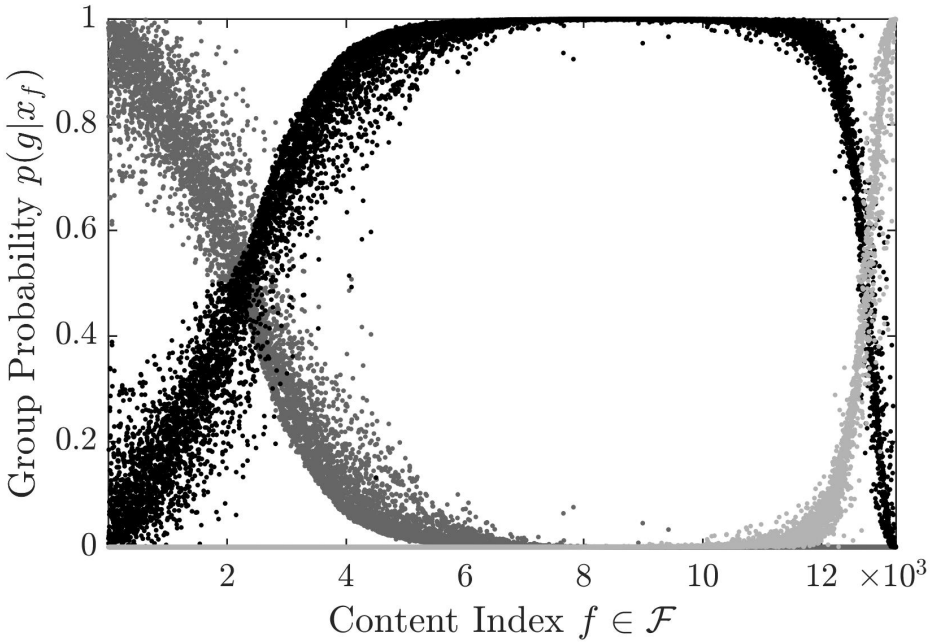

Figure 4.7: Group association probability $P(g|x_f)$. Gray dots indicate the probability of association with group $g = 1$, black dots with $g = 2$, and light-gray dots with $g = 3$. Performance of the conformal prediction algorithm that was schematically illustrated in Fig. 4.4, for YouTube content requests. Group $g = 1$ is associated with all videos that are predicted to have less than 100 requests, $g = 2$ with requests in the range of 100 to 30,000, and group $g = 3$ with more then 30,000 requests. The 90% confidence interval is evaluated using the empirical cumulative distribution function $\widehat{F}_{Y_f}(y_f|x_f)$ of content requests defined in (4.24). As seen, the computed group association, point content requests, and conformal predictions are in excellent agreement with the YouTube datasets. The training and evaluation datasets are discussed in Sec.4.5.1.

scheme (4.4) for all delay threshold values d. This results as the RANA caching scheme accounts for the uncertainty associated with estimating the content requests. For example, if we are interested in the probability the delay is less than the threshold $d = 50$ seconds, the RNNA scheme is approximately 50%, while the

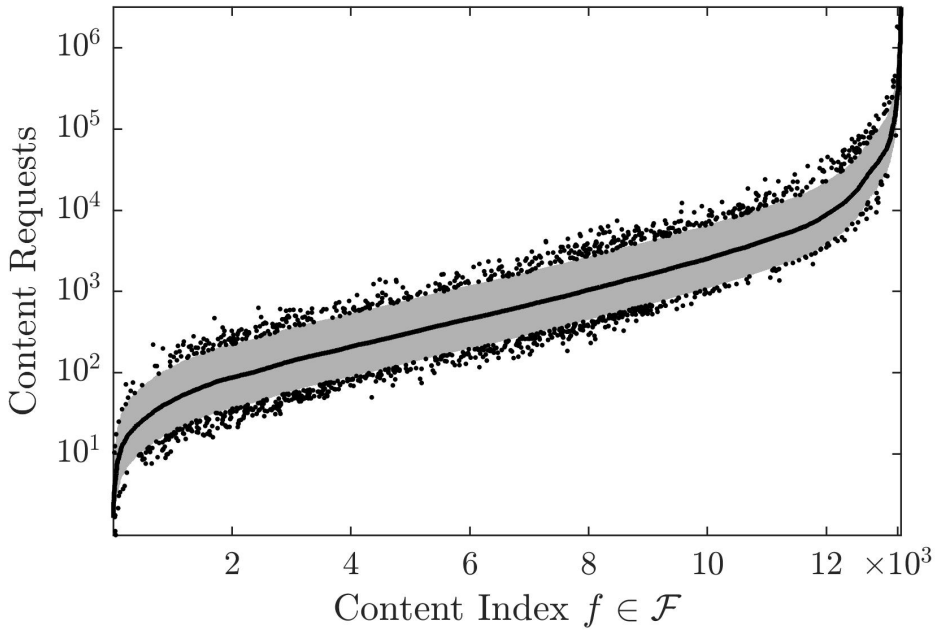

Figure 4.8: Conformal prediction of the number of content requests. The expected number of requests is the solid black line, gray region is the 90% interval where the requests are expected to reside, and the black dots are the real number of requests in $\overline{\mathscr{D}}$.

RANA schemes are approximately 98%. A substantial improvement in performance is obtained by accounting for the prediction error in the caching decisions. The associated delay of between the RANA caching schemes with $\alpha = 0.99$ and $\alpha = 0.9$ are approximately equal except in the delay range of 46 seconds to 49 seconds. In this region the selection of α is important. For example, if $d = 48$ seconds, then the probability the delay is 50% for RANA with $\alpha = 0.99$, and 60% RANA with $\alpha = 0.9$. Therefore, if the network operator wants to minimize the probability of the delay exceeding the threshold $d_{th} = 48$ seconds, an $\alpha = 0.9$ should be selected. The reason that a larger value of α does not guarantee minimizing $F_D(d)$, for a specific d, is that as $\alpha \to 1$, the maximum delay is being minimized in (4.12).

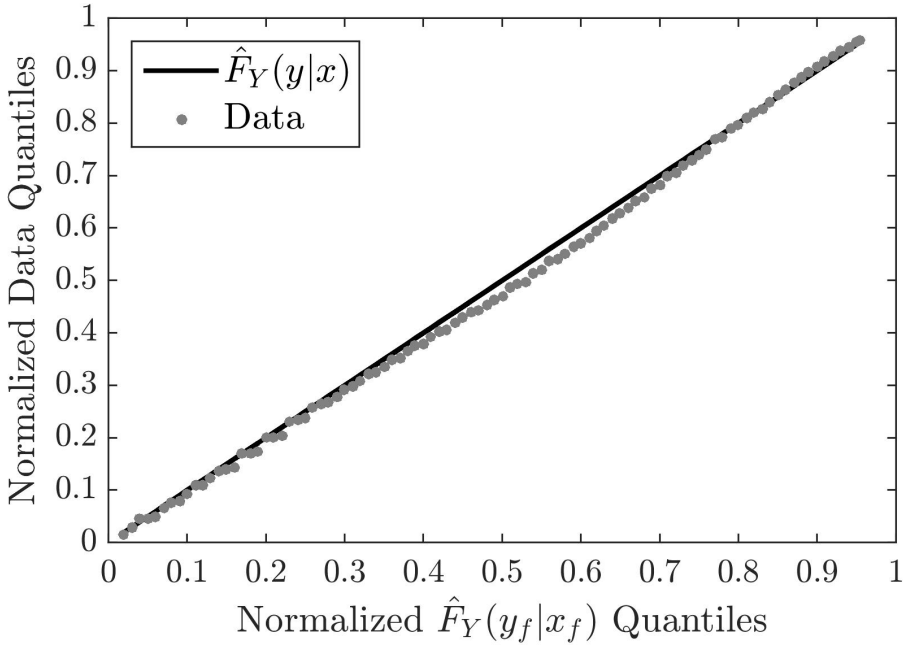

Figure 4.9: Quantile-quantile plot for the empirical cumulative distribution function $\widehat{F}_Y(y_f|x_f)$ and the YouTube content requests. The linear black line indicated perfect agreement between the data and distribution, and the grey dots indicate the quantiles computed from the YouTube data.

4.5.4 Performance of the Risk-Neutral and Risk-Aware Caching Schemes

In this section, we illustrate the performance of the four caching schemes (RN and RNNA) in Secs.4.3.1, 4.3.2, and (RA and RANA) in Secs.4.3.4, 4.3.5. Additionally, we compare the performance of these four caching schemes with the caching scheme presented in [88]. The caching scheme in [88] accounts for the content requests, and network parameters (cache size, bandwidth, latency among nodes), however it does not account for the routing protocol used in the network or the uncertainty associated with the estimated content requests. We refer to the caching scheme in [88] as the risk-neutral without routing (RNWR) caching scheme.

The performance metric we use to compare these five caching schemes is the

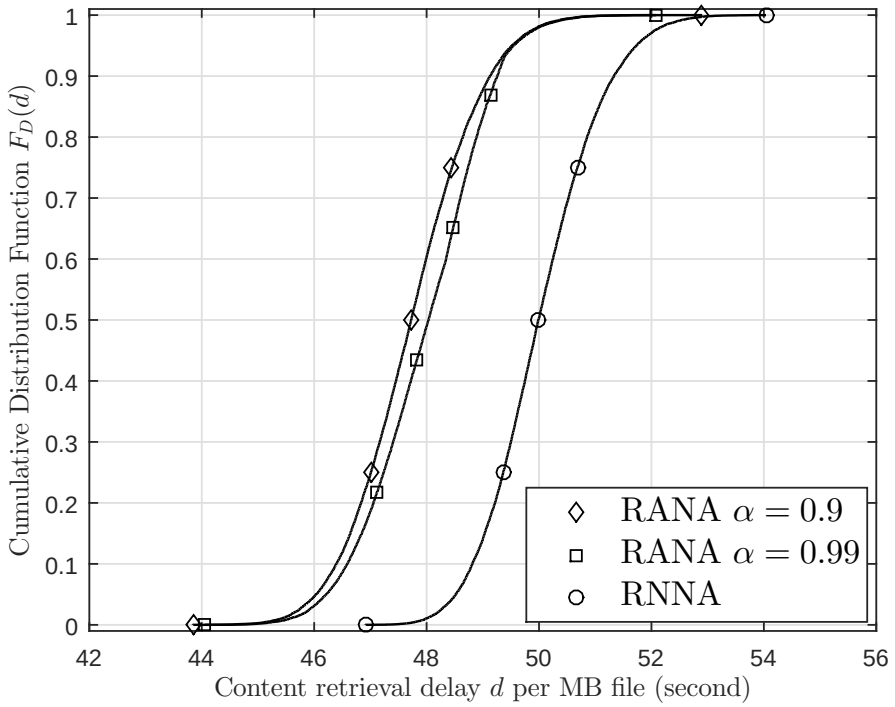

Figure 4.10: Cumulative distribution function of the content retrieval delay $F_D(d)$ using the RNNA and RANA caching schemes for a confidence level $\alpha = 0.9, 0.99$. The circles denote the RNNA caching scheme (4.4), diamond shape indicate the RANA caching scheme (4.12) with $\alpha = 0.9$, and squares for the RANA caching scheme with $\alpha = 0.99$. As the value of $F_D(d) = P(D \leq d)$ increases, the probability the delay exceeds d decreases.

cumulative distribution function $F_D(d)$ for the content retrieval delay. To estimate $F_D(d)$, we set $K = 10,000$ and $\alpha = 0.9$, and generate $20,000$ samples of the total content retrieval delay D. Fig. 4.11 illustrates empirical $F_D(d)$ from the five caching schemes. The results in Fig. 4.11 illustrate that the RA caching scheme (4.11) has a lower delay than the RN caching scheme (4.3) for all possible values of d. That is, the delay that results from the RN caching scheme illustrates a first-order stochastic dominance compared with the delay that results from the

RA caching scheme. The delay of the RN and RA schemes are approximately twice as large as compared with the RNWR scheme which accounts for the network parameters but not the routing protocol used in the network. Therefore, including network parameters can substantially reduce the delay that results when performing a caching decision. Comparing the results for the RNWR, RNNA, and RANA, both the RNNA and RANA caching schemes significantly reduce the content retrieval delay in the network compared with the RNWR caching scheme by approximately 25%. This results as the RNNA and RANA schemes both reduce the congestion of transferring content throughout the network as they account for the network routing protocol used. Finally, the RANA scheme provides the lowest content retrieval delay compared with the other four schemes as it accounts for the uncertainty of content requests, network parameters, and the routing protocol used to transfer content throughout the network.

4.6 Chapter Summary

In this work we designed risk-neutral and risk-averse caching schemes for heterogeneous networks that contain smallcell access points and base stations which have limited storage capacity and low bandwidth backhaul links. The risk-averse caching schemes employed the coherent Conditional Value-at-Risk (CVaR) measure to account for the uncertainty of estimating the content requests to perform the caching decisions. The CVaR risk measure is evaluated using information from the conformal prediction algorithm which constructs the cumulative distribution function of the content requests based on the content features. Using real-world datasets from YouTube and the NS-3 simulator, we demonstrate how the caching schemes reduce the delay of retrieving content in heterogeneous networks compared with industry standard caching schemes. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 25% reduction in average delay is achieved if both the uncertainty and network routing protocol are accounted for compared to the risk-neutral caching that neglects the routing protocol.

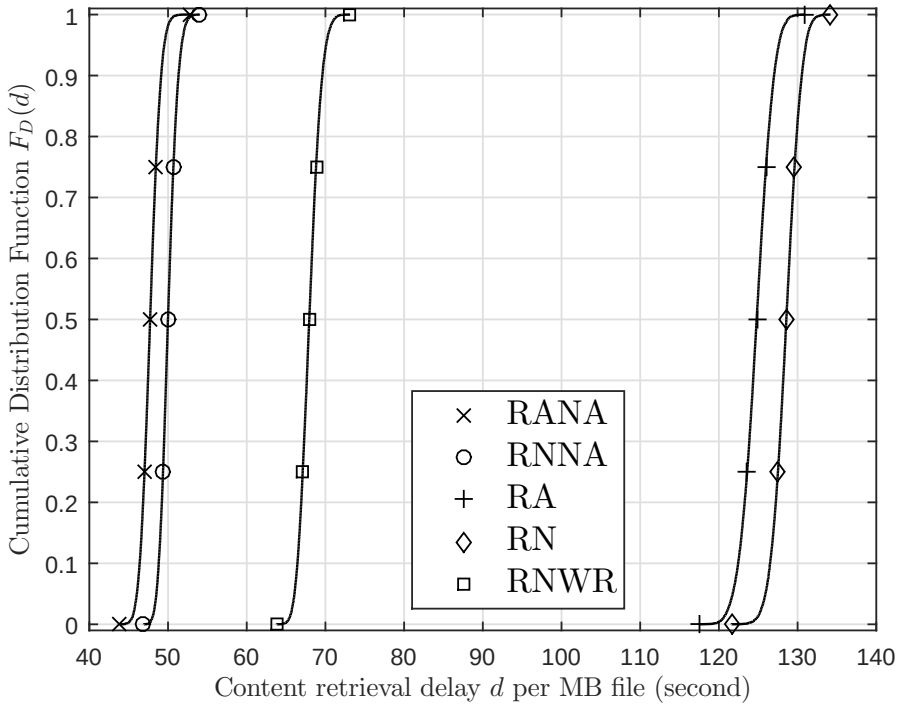

Figure 4.11: The cumulative distribution function $F_D(d)$ of the content retrieval delay for the RN, RNNA, RA, RANA, RNWR caching schemes. To evaluate $F_D(d)$ for the RA and RANA caching schemes, we set $K = 10,000$ and $\alpha = 0.9$ in (4.11) and (4.12). A total of $20,000$ samples are generated for the content retrieval delay to construct the empirical cdf $F_D(d)$. The results illustrate that the delay of the RN, RNNA, RA, and RNWR schemes all first-order stochastically dominate the delay associated with the RANA caching scheme.

Chapter 5

Conclusions and Future Research Directions

In this chapter, we summarize the main contributions of the thesis and describe future research directions.

5.1 Conclusions

The growing popularity of cloud computing offers mobile users to enjoy a wide variety of applications. However, the traditional centralized cloud computing architecture faces several challenges. Offloading tasks to a distant remote cloud incur data transferring delay which is not suitable for time-critical applications while sending data associated with tasks migration to a remote cloud increases the traffic volume in the network. On the other hand, the gaining popularity of video streaming via mobile devices surges traffic volume in the network. To address these challenges new architecture, namely, mobile edge cloud has gained attention from both industry and academia. The main idea of mobile edge cloud is to integrate computation and storage at the edge of the network, serving a portion of the users' requests locally, resulting in a reduction in data traffic volume and data transferring delay. The unified theme of the thesis was to design mecha-

nisms for mobile edge cloud to maximally utilize edge network resources. In this thesis, we made three major contributions and a summary of these contributions is as follows.

- In the first part of the thesis, we designed a distributed resource sharing mechanism for mobile edge cloud where edge nodes such as FAPs share their computational resources with the neighbours FAPs and form local clouds to reduce the number of offloading tasks to a remote cloud. As such, the resulting local *femto-clouds* reduce overall latency associated with tasks migration to a remote cloud, and hence, improve users' QoE. To motivate FAP owners to share their resources for performing tasks in *femto-clouds*, a monetary incentive mechanism was introduced. To this end, we proposed a distributed *femto-clouds* formation mechanism and formulated the problem as an optimization problem with the objective to maximize overall utilities such as higher monetary incentives and lower overall network latency while ensuring a fair division of incentives among individual FAPs within the *femto-clouds*. In particular, the problem was formulated as a coalition formation game with modified core and the proposed algorithm reached to the optimal solution in a distributed fashion. The numerical results verified the applicability of the proposed *femto-clouds* formation mechanism in a wide range of scenarios e.g., hotspot, residential, and enterprise environment of the femtocell network.

- In the second part of the thesis, we constructed a caching scheme to maximally utilize storage resources at the edge of the network. The presented caching scheme reduces traffic volume in the network by serving users' requests locally and also reduces content downloading delay. To this end, we constructed a caching scheme that takes into account users' behaviour and operating characteristics of the network. In particular, the presented caching scheme estimated content popularity from the content features and requests statistics of users as they were available. Then a mixed integer

linear program was formulated that took into account content popularity, and network parameters such as network topology, a communication link to select where to place content in the network. Numerical evaluations using YouTube videos demonstrated the efficacy of the proposed caching scheme.

- The caching scheme presented in the second part of the thesis did not include content retrieval path in the caching decision. As a result, a separate routing mechanism was employed to retrieve content from the neighbour nodes. In the third part of the thesis, we designed a caching scheme that not only considered content popularity and network parameters but also accounted for the routing mechanism in the caching decision. As such, the presented caching scheme reduced content downloading and traffic volume in the network while balancing the traffic load over communication links using the routing mechanism. To this end, we constructed a caching scheme which is referred to the risk-neutral caching scheme in this thesis. The risk-neutral caching scheme required to solve a unimodular linear program. In order to incorporate uncertainty associated with the error in the content popularity prediction, we presented a risk-averse caching scheme where uncertainty was modelled using conditional value at risk measure. The numerical evaluation demonstrated that an improved network performance can be obtained by including routing mechanism in the caching decision. Numerical results also demonstrated the benefit of the risk-averse caching scheme over the risk-neutral caching scheme.

5.2 Future Research Problems

In this section, we describe three future research problems in mobile edge cloud. Precisely, the first problem is related to the mobile edge cloud computing. The main target of this problem is to design frame allocation mechanism for edge cloud assisted mobile game to improve mobile users' gaming experience. The aim of the second problem is to design caching method where individual wire-

less node learns content popularity locally by observing content request statistics. The third problem combines both computations and caching at the edge of the network. The main target of this problem is to design a caching strategy that stores content with appropriate bit rate version. In this problem, edge computing resources are employed to perform computations related to transcoding of a content. A schematic of the future research problems is depicted in Fig.5.1.

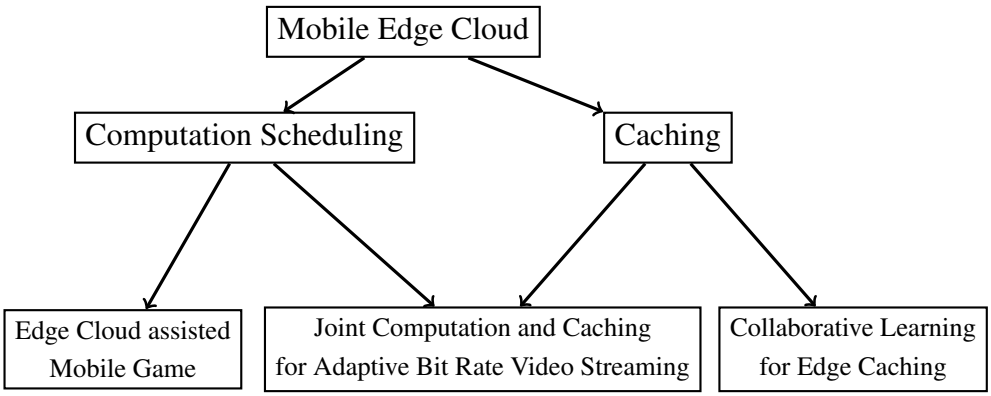

Figure 5.1: A schematic view of the future research challenges.

5.2.1 Frame Allocation Mechanism for Edge Cloud Assisted Mobile Gaming

With the proliferation of smart phones, mobile gaming is gaining popularity among young generation. Although smart phones offer a wide variety of games, they are mostly single player games due to the resource constraints of the mobile devices such as battery life and computational power. Mobile cloud gaming has been proposed to support complicated multi-player games on mobile devices without augmenting resources. The idea is to outsource computational burdens to a cloud server. In mobile cloud gaming, the mobile device behaves like a thin client and works as a platform to send gaming control inputs to the cloud server. The cloud server processes the gaming scenarios and then streams back video frames to the mobile device. Players are interacting each other in the cloud via a wireless net-

work. Mobile cloud gaming enables users to play any game using their mobile devices as long as the device has the ability to support video. Although mobile cloud gaming prolongs the battery life of the mobile device, it also introduces latency which affects players' QoE. Edge cloud assisted mobile gaming is one possible solution to overcome latency issue in mobile cloud gaming [152, 153].

Though edge cloud assisted mobile gaming takes care of the latency issue, it also introduces several challenges. On the one hand, edge cloud nodes are possessing a finite amount of resources and a number of users are sharing these resources simultaneously. Therefore, the availability of the computational resources has an impact on the processing delay. On the other hand, to make a game interactive and to avoid jerky video, a minimum number of video frames needs to be received by the mobile device in a second. However, the time-varying nature of the wireless channels limits the number of packet delivery through wireless channels. In addition, users' prefer high-quality video frames which consist of a higher number of packets and hence requires longer transmission and processing time. Studies reveal that video quality and response delay affects mobile gaming user experience and users may quit the game [154].

Motivated by the aforementioned facts, designing a frame allocation mechanism for edge cloud assisted mobile gaming is important where the main objective of the frame allocation mechanism is to maximize mobile gaming user experience while satisfying all the constraints: the target number of frames in a second, processing delay of the video frames, edge cloud nodes computational workload, and wireless channels' supportable packet rate. One possible way to solve the problem is to formulate the problem as a finite horizon Markov decision problem where the objective is to maximize mobile gaming user experience over the horizon while satisfying all the constraints. To overcome the complexity issue in Markov decision process, designing sub-optimal solutions are also important for this problem.

5.2.2 Collaborative Learning for Edge Caching

The caching methods presented in this thesis first estimate content popularity from their features using learning algorithms. Having computed the content popularity, the caching methods employ optimization techniques to decide the content to be cached in the edge nodes. As such, the presented caching methods involve two steps for caching a content. As already mentioned, there are several advantages of the presented caching methods such as:

i) the content popularity estimation technique enables to cache popular content without observing requests for the content. Thereby, content downloading delay is reduced even if the content has recently been uploaded;

ii) since content popularities are already available from the learning technique, the content caching can be performed when the network load is minimal; and

iii) the estimation techniques and optimization techniques are not tethered. As a result, different estimation techniques can be employed for the presented optimization techniques and vice versa.

Although presented caching methods have several advantages, they incur cache initialization cost that is content transferring cost at the beginning of caching. The entire cache needs to be updated with the recent popular ones when content popularity changes. As a result, when content popularity changes rapidly, learning content popularity and cache content simultaneously, provides more benefit than the presented caching methods that requires cache initialization.

Few research works consider learning and caching content simultaneously for the edge network [67],[68] where the caching methods involve solving a multi-armed bandit problem. In particular, at the beginning, content are cached randomly and edge node records requests for each of the cached content. After an interval, edge node evicts content that receive a lower number of requests than a target threshold. Then the cache is filled with some new content and previous interval popular content. This procedure continues and asymptotically the cache will be filled with the popular content [67]. Nonetheless, the presented caching methods require keeping track of the request statistics for each of the content.

To overcome such limitation, contextual multi-armed bandit caching methods are proposed where content are clustered based on context and the eviction decisions are made based on the cluster the content belongs to[68]. Although the presented caching methods improve learning speed, the caching methods still require a certain number of content requests to identify popular clusters.

Collaborative learning can provide a significant improvement in learning speed when edge nodes are densely populated such as in today heterogeneous hotspot areas. According to the collaborative learning, edge nodes are grouped together according to their content popularity pattern. As such, edge nodes that receive significantly different content requests reside in separate groups. The main advantage of collaborative learning compared to the methods presented in [68] is that collaborative learning not only exploits hidden information of the context but also shares the gathered information with the neighbour edge nodes, as such edge nodes with same popularity pattern learn content popularity as a single entity. The resulting procedure thus takes advantage of content popularity patterns in the data in a way akin to collaborative filtering methods[155],[156].

5.2.3 Joint Computation and Caching for Adaptive Bit Rate Video Streaming

The caching methods exist in the literature do not account for different bit rate version of the content. However, in reality, a content is requested from heterogeneous platforms such as a laptop, smart phone, tablet and a transcoding mechanism is required to adapt the bit rate version of the content according to the platform. In addition, the time-varying nature of the wireless channels advocates the necessity for considering different bit rate version of the content in the caching decision.

One way to handle this issue is to deploy some computational resources at the edge caching node and perform computations associated with the transcoding. The main benefit of such solution is that the edge node can cache diverse content as such only one copy of a content is cached at the edge node. However, this solution may not perform well when some popular content contribute to a large

portion of the content requests from the heterogeneous platform. In such case, transcoding for each and every requests not only costs a significant amount of computational resources but also introduces latency, resulting degradation in the users' QoE. Therefore, we need a joint computational and caching methods that not only prescribe which content to cache but also recommend which version(s) of the content to cache so that computational and caching resources are maximally utilized. Authors of [157] have recently considered this issue which needs further attention from the research community.

Bibliography

[1] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in Proceedings of the IEEE Global Internet of Things Summit, pp. 1–6, 2017. → pages xii, 5

[2] N. C. Nguyen, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: a survey," IEEE Communications Surveys & Tutorials, 2017. → pages 3

[3] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol. 29, no. 1, pp. 84–106, 2013. → pages 3

[4] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," Computer Networks, vol. 57, no. 9, pp. 2093–2115, 2013. → pages 3

[5] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," Wireless Communications and Mobile Computing, vol. 13, no. 18, pp. 1587–1611, 2013. → pages 3, 12

[6] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," Mobile Networks and Applications, vol. 19, no. 2, pp. 133–143, 2014. → pages 3

[7] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in Proceedings of the IEEE INFOCOM, pp. 1285–1293, 2013. → pages 3, 4

[8] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Communications Surveys & Tutorials, 2017. → pages 3

[9] 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, no. 11, pp. 1–16, 2015. → pages 4

[10] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in Proceedings of the ACM Workshop on Mobile Big Data, pp. 37–42, 2015. → pages 4

[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," vol. 8, no. 4, pp. 14–23, 2009. → pages 4, 12, 13

[12] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in Proceedings of the International Conference on Mobile Systems, Applications, and Services, (San Francisco, CA), pp. 49–62, 2010. → pages 4, 12, 27

[13] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications, (Darmstadt, Germany), pp. 26–30, 2013. → pages 5, 16

[14] J. Zhang, W. Xie, F. Yang, and Q. Bi, "Mobile edge computing and field trial results for 5g low latency scenario," China Communications, vol. 13, no. Supplement2, pp. 174–182, 2016. → pages 7

[15] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in Proceedings of the IEEE Standards for Communications and Networking, pp. 1–7, 2016. → pages 7

[16] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, "Are cloudlets necessary?," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-139, 2015. → pages 8

[17] S. H. Chae and W. Choi, "Caching placement in stochastic wireless caching helper networks: Channel selection diversity via caching," IEEE Transactions on Wireless Communications, vol. 15, no. 10, pp. 6626–6637, 2016. → pages 8

[18] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020," 2016. → pages 8

[19] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in Proceedings of the IEEE INFOCOM, pp. 1–9, 2016. → pages 8

[20] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," IEEE Communications Magazine, vol. 52, no. 2, pp. 131–139, 2014. → pages 9

[21] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and d2d networks," IEEE Journal on Selected Areas in Communications, vol. 34, no. 5, pp. 1222–1234, 2016. → pages 9

[22] N. Zhao, X. Liu, F. R. Yu, M. Li, and V. C. Leung, "Communications, caching, and computing oriented small cell networks with interference alignment," IEEE Communications Magazine, vol. 54, no. 9, pp. 29–35, 2016. → pages 9

[23] C. V. N. Index, "Global mobile data traffic forecast update, 2015-2020," Cisco white paper, 2016. → pages 9

[24] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," IEEE Transactions on Wireless Communications, vol. 15, no. 1, pp. 131–145, 2016. → pages 9

[25] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," IEEE/ACM Transactions on Networking (TON), vol. 22, no. 5, pp. 1444–1462, 2014. → pages 10, 17

[26] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed

caching helpers," IEEE Transactions on Information Theory, vol. 59, no. 12, pp. 8402–8413, 2013. → pages 10, 17, 19, 78, 93

[27] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," IEEE Communications Magazine, vol. 51, no. 4, pp. 142–149, 2013. → pages 10

[28] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in Proceedings of the IEEE INFOCOM, pp. 2426–2434, 2012. → pages 11

[29] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, "Mobility-aware caching for content-centric wireless networks: Modeling and methodology," IEEE Communications Magazine, vol. 54, no. 8, pp. 77–83, 2016. → pages 11

[30] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," IEEE Communications Magazine, vol. 52, no. 8, pp. 82–89, 2014. → pages 11, 93

[31] S. Agarwal, M. Philipose, and P. Bahl, "Vision: The case for cellular small cells for cloudlets," in Proceedings of the International Workshop on Mobile Cloud Computing & Services, (Bretton Woods, NH), pp. 1–5, 2014. → pages 12

[32] P. Bahl, R. Y. Han, E. E. Li, and M. Satyanarayanan, "Advancing the state of mobile cloud computing," in Proceedings of the ACM Workshop on Mobile Cloud Computing & Services, (Ambleside, UK), pp. 21–28, 2012. → pages 13

[33] S. Barbarossa, P. Di Lorenzo, and S. Sardellitti, "Computation offloading strategies based on energy minimization under computational rate constraints," in Proceedings of the European Conference on Networks and Communications, (Bologna, Italy), pp. 1–5, 2014. → pages 16

[34] F. L. Vilela, A. J. Ferrer, M. A. Puente, Z. Becvar, M. Rohlik, T. Vanek, P. Mach, O. M. Medina, J. V. Manzano, H. Hariyanto, et al., "TROPIC-D22 design of network architecture for femto-cloud computing," 2013. → pages 13, 26

[35] T. Arnold and U. Schwalbe, "Dynamic coalition formation and the core," Journal of Economic Behavior & Organization, vol. 49, no. 3, pp. 363–380, 2002. → pages 14, 34, 36, 37

[36] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," IEEE Journal on Selected Areas in Communications, vol. 31, no. 12, pp. 2685–2700, 2013. → pages 15

[37] M. Guazzone, C. Anglano, and M. Sereno, "A game-theoretic approach to coalition formation in green cloud federations," in Proceedings of the IEEE International Symposium on Cluster, Cloud and Grid Computing, (Chicago, IL), pp. 618–625, 2014. → pages 15

[38] C. A. Lee, "Cloud federation management and beyond: Requirements, relevant standards, and gaps," IEEE Cloud Computing, vol. 3, no. 1, pp. 42–49, 2016. → pages 15

[39] T. Truong-Huu and C.-K. Tham, "A novel model for competition and cooperation among cloud providers," IEEE Transactions on Cloud Computing, vol. 2, no. 3, pp. 251–265, 2014. → pages 15

[40] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," IEEE Transactions on Cloud Computing, vol. 3, no. 1, pp. 14–27, 2015. → pages 15

[41] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "Spectrum leasing as an incentive towards uplink macrocell and femtocell cooperation," IEEE Journal on Selected Areas in Communications, vol. 30, no. 3, pp. 617–630, 2012. → pages 15

[42] O. N. Gharehshiran, A. Attar, and V. Krishnamurthy, "Collaborative sub-channel allocation in cognitive lte femto-cells: a cooperative game-theoretic approach," IEEE Transactions on Communications, vol. 61, no. 1, pp. 325–334, 2013. → pages 15

[43] Z. Zhang, L. Song, Z. Han, and W. Saad, "Coalitional games with overlapping coalitions for interference management in small cell networks," IEEE Transactions on Wireless Communications, vol. 13, no. 5, pp. 2659–2669, 2014. → pages 15

[44] F. Pantisano, M. Bennis, W. Saad, M. Debbah, and M. Latva-Aho, "Interference alignment for cooperative femtocell networks: A game-theoretic approach," IEEE Transactions on Mobile Computing, vol. 12, no. 11, pp. 2233–2246, 2013. → pages 15

[45] R. Langar, S. Secci, R. Boutaba, and G. Pujolle, "An operations research game approach for resource and power allocation in cooperative femtocell networks," IEEE Transactions on Mobile Computing, vol. 14, no. 4, pp. 675–687, 2015. → pages 15

[46] S.-Y. Yun, Y. Yi, D.-H. Cho, and J. Mo, "The economic effects of sharing femtocells," IEEE journal on selected areas in Communications, vol. 30, no. 3, pp. 595–606, 2012. → pages 15

[47] L. Gao, G. Iosifidis, J. Huang, and L. Tassiulas, "Economics of mobile data offloading," in Proceedings of the IEEE INFOCOM Workshops, (Turin, Italy), pp. 351–356, 2013. → pages 15

[48] Y. Chen, J. Zhang, and Q. Zhang, "Utility-aware refunding framework for hybrid access femtocell network," IEEE Transactions on Wireless Communications, vol. 11, no. 5, pp. 1688–1697, 2012. → pages 15

[49] S. Hua, X. Zhuo, and S. S. Panwar, "A truthful auction based incentive framework for femtocell access," in Proceedings of the IEEE WCNC, (Shanghai, China), pp. 2271–2276, 2013. → pages 15

[50] L. Duan, J. Huang, and B. Shou, "Economics of femtocell service provision," IEEE Transactions on Mobile Computing, vol. 12, no. 11, pp. 2261–2273, 2013. → pages 15

[51] N. Shetty, S. Parekh, and J. Walrand, "Economics of femtocells," in Proceedings of the IEEE GLOBECOM, (Honolulu, HI), pp. 1–6, 2009. → pages 15

[52] Y. Zhang and M. van der Schaar, "Peer-to-peer multimedia sharing based on social norms," Signal Processing: Image Communication, vol. 27, no. 5, pp. 383–400, 2012. → pages 15

[53] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," in Financial

Cryptography (R. N. Wright, ed.), vol. 2742 of Lecture Notes in Computer Science, pp. 15–33, 2003. → pages 15

[54] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz, "Selfish caching in distributed systems: A game-theoretic analysis," in Proceedings of the Annual ACM Symposium on Principles of Distributed Computing, (Elche, Spain), pp. 21–30, 2004. → pages 15

[55] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in Advances in Cryptology (R. Cramer, ed.), vol. 3494 of Lecture Notes in Computer Science, pp. 302–321, 2005. → pages 15

[56] L. Buttyán and J.-P. Hubaux, "Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks," tech. rep., 2001. → pages 15

[57] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," IEEE Transactions on Vehicular Technology, vol. 64, no. 10, pp. 4738–4755, 2015. → pages 16

[58] M. Ali, Q. Rabbani, M. Naeem, S. Qaisar, and F. Qamar, "Joint user association, power allocation, and throughput maximization in 5g h-cran networks," IEEE Transactions on Vehicular Technology, vol. 66, no. 10, pp. 9254–9262, 2017. → pages 16

[59] J. Oueis, E. C. Strinati, and S. Barbarossa, "Small cell clustering for efficient distributed cloud computing," in Proceedings of the IEEE PIMRC, (Washington, DC), pp. 1474–1479, 2014. → pages 16

[60] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in Proc. of IEEE VTC, vol. Spring, (Glasgow, Scotland), pp. 1–6, 2015. → pages 16

[61] S. Andreev, O. Galinina, A. Pyattaev, J. Hosek, P. Masek, H. Yanikomeroglu, and Y. Koucheryavy, "Exploring synergy between communications, caching, and computing in 5g-grade deployments," IEEE Communications Magazine, vol. 54, no. 8, pp. 60–69, 2016. → pages 17

[62] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5g wireless networks," IEEE Transactions on

Wireless Communications, vol. 15, no. 4, pp. 2995–3007, 2016. → pages 17

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

[64] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," IEEE Transactions on Mobile Computing, vol. 16, no. 5, pp. 1382–1393, 2017. → pages 17

[65] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," IEEE Transactions on Communications, vol. 64, no. 4, pp. 1674–1686, 2016. → pages 17

[66] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," IEEE Transactions on Wireless Communications, vol. 16, no. 2, pp. 1024–1036, 2017. → pages 17, 87, 93

[67] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in Proceedings of IEEE International Conference on Communications (ICC), pp. 1897–1903, 2014. → pages 17, 60, 130

[68] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in Proceedings of the International Symposium on Wireless Communications Systems, pp. 917–921, 2014. → pages 17, 130, 131

[69] 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," Journal of Communications and Networks, vol. 17, no. 6, pp. 549–557, 2015. → pages 17

[70] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," IEEE Communications Magazine, vol. 52, no. 8, pp. 82–89, 2014. → pages 17

[71] E. Baştuğ, M. Bennis, and M. Debbah, "Proactive caching in 5g small cell networks," Towards 5G: Applications, Requirements and Candidate Technologies, pp. 78–98, 2016. → pages 17, 19, 78

[72] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in Proceedings of the 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), pp. 161–166, 2015. → pages 17

[73] E. Bastug, M. Bennis, and M. Debbah, "Anticipatory caching in small cell networks: A transfer learning approach," in Proceedings of the 1st KuVS Workshop on Anticipatory Networks, 2014. → pages 17

[74] G. Huang, G. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," Neural Networks, vol. 61, pp. 32–48, 2015. → pages 18, 64

[75] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale vod system," IEEE/ACM Transactions on Networking, vol. 24, pp. 2114–2127, 2016. → pages 19, 78, 112

[76] H. Pinto, J. Almeida, and M. Gonçalves, "Using early view patterns to predict the popularity of YouTube videos," in Proceedings of the 6th ACM International Conference on Web Search and Data Mining, pp. 365–374, 2013. → pages 20

[77] Z. Tan, Y. Wang, Y. Zhang, and J. Zhou, "A novel time series approach for predicting the long-term popularity of online videos," IEEE Transactions on Broadcasting, vol. 62, no. 2, pp. 436–445, 2016. → pages 20

[78] J. Wu, Y. Zhou, M. Chiu, and Z. Zhu, "Modeling dynamics of online video popularity," IEEE Transactions on Multimedia, vol. 18, no. 9, pp. 1882–1895, 2016. → pages 20

[79] C. Li, J. Liu, and S. Ouyang, "Characterizing and predicting the popularity of online videos," IEEE Access, vol. 4, pp. 1630–1641, 2016. → pages 20

[80] R. Zhou, S. Khemmarat, L. Gao, J. Wan, J. Zhang, Y. Yin, and J. Yu, "Boosting video popularity through keyword suggestion and

recommendation systems," Neurocomputing, vol. 205, pp. 529–541, 2016. → pages 20

[81] W. Hoiles, A. Aprem, and V. Krishnamurthy, "Engagement and popularity dynamics of YouTube videos and sensitivity to meta-data," IEEE Transactions on Knowledge & Data Engineering, no. 7, pp. 1426–1437, 2017. → pages 20

[82] A. Tay and K. Wallis, "Density forecasting: a survey," Journal of forecasting, vol. 19, no. 4, p. 235, 2000. → pages 20

[83] D. Hamilton, Time series analysis, vol. 2. Princeton university press, 1994. → pages 20

[84] L. Fang and D. Bessler, "Stock returns and interest rates in china: the prequential approach," Applied Economics, pp. 1–14, 2017. → pages 20

[85] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," IEEE Transactions on Information Theory, vol. 59, no. 12, pp. 8402–8413, 2013. → pages 20, 21

[86] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," IEEE Transactions on Communications, vol. 64, no. 4, pp. 1674–1686, 2016. → pages 20, 21

[87] J. Song, H. Song, and W. Choi, "Optimal content placement for wireless femto-caching network," IEEE Transactions on Wireless Communications, vol. 16, no. 7, pp. 4433–4444, 2017. → pages 20, 21

[88] S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," IEEE Access, vol. 5, pp. 5870–5881, 2017. → pages 21, 121

[89] J. He and W. Song, "Optimizing video request routing in mobile networks with built-in content caching," IEEE Transactions on Mobile Computing, vol. 15, no. 7, pp. 1714–1727, 2016. → pages 21

[90] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks,"

IEEE/ACM Transactions on Networking, vol. 25, no. 3, pp. 1635–1648, 2016. → pages 21

[91] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in Proceedings of the IEEE INFOCOM, pp. 1078–1086, 2014. → pages 21

[92] T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," Journal of risk, vol. 2, pp. 21–42, 2000. → pages 22, 97, 98

[93] T. Rockafellar and S. Uryasev, "Conditional value-at-risk for general loss distributions," Journal of banking & finance, vol. 26, no. 7, pp. 1443–1471, 2002. → pages 97

[94] P. Krokhmal, J. Palmquist, and S. Uryasev, "Portfolio optimization with conditional value-at-risk objective and constraints," Journal of risk, vol. 4, pp. 43–68, 2002. → pages 22

[95] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood, "The value of reputation on ebay: A controlled experiment," Experimental Economics, vol. 9, no. 2, pp. 79–101, 2006. → pages 31

[96] J. P. Kahan and A. Rapoport, Theories of Coalition Formation. New York, NY: Psychology Press, 2014. → pages 34

[97] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar, "Coalitional game theory for communication networks," IEEE Signal Processing Magazine, vol. 26, no. 5, pp. 77–97, 2009. → pages 34

[98] G. Owen, Game Theory. New York, NY: Academic Press, 1995. → pages 34

[99] O. N. Gharehshiran, Distributed Dynamic Coalition Formation for Bearings-only Localization in Wireless Sensor Networks. PhD thesis, University of British Columbia (Vancouver), 2010. → pages 38

[100] B.-Y. Su, Parallel Application Library for Object Recognition. PhD thesis, EECS Department, University of California, Berkeley, 2012. → pages 40

[101] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in Proceedings of the IEEE CVPR, (Anchorage, AK), pp. 1–8, 2008. → pages 40

[102] J. Li and J. Z. Wang, "Studying digital imagery of ancient paintings by mixtures of stochastic models," IEEE Transactions on Image Processing, vol. 13, no. 3, pp. 340–353, 2004. → pages 40

[103] K. Tan and S. Chen, "Adaptively weighted sub-pattern PCA for face recognition," Neurocomputing, vol. 64, pp. 505–511, 2005. → pages 40

[104] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented LTE network simulator based on ns-3," in Proceedings of the ACM MSWiM, (Miami Beach, FL), pp. 293–298, 2011. → pages 41

[105] N. Baldo, M. Requena-Esteso, M. Miozzo, and R. Kwan, "An open source model for the simulation of LTE handover scenarios and algorithms in ns-3," in Proceedings of the ACM MSWiM, (Barcelona, Spain), pp. 289–298, 2013. → pages 41

[106] J. Zhang, X. Zhang, and W. Wang, "Cache-enabled software defined heterogeneous networks for green and flexible 5g networks," IEEE Access, vol. 4, pp. 3591–3604, 2016. → pages 54

[107] J. Liu, Q. Yang, and G. Simon, "Optimal and practical algorithms for implementing wireless cdn based on base stations," in Proceedings of the IEEE Vehicular Technology Conference, pp. 1–5. → pages 54

[108] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale vod system," in Proceedings of the International Conference on Emerging Networking Experiments and Technologies, p. 4, 2010. → pages 60

[109] J. Till, S. Engell, S. Panek, and O. Stursberg, "Empirical complexity analysis of a milp-approach for optimization of hybrid systems," in Proceedings of the Conference on Analysis and Design of Hybrid Systems, pp. 129–134, 2003. → pages 60

[110] K. Genova and V. Guliashki, "Linear integer programming methods and approaches–a survey," Journal of Cybernetics and Information Technologies, vol. 11, no. 1, 2011. → pages 60

[111] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," Operations research, vol. 46, no. 3, pp. 316–329, 1998. → pages 60

[112] R. H. Bartels and G. H. Golub, "The simplex method of linear programming using lu decomposition," Communications of the ACM, vol. 12, no. 5, pp. 266–268, 1969. → pages 60

[113] F. A. Potra and S. J. Wright, "Interior-point methods," Journal of Computational and Applied Mathematics, vol. 124, no. 1, pp. 281–302, 2000. → pages 60

[114] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of facebook photo caching," in Proceedings of the ACM Symposium on Operating Systems Principles, pp. 167–181, 2013. → pages 61, 89, 91

[115] M. Guzelsoy and T. K. Ralphs, "Duality for mixed-integer linear programs," International Journal of Operations Research, vol. 4, no. 3, pp. 118–137, 2007. → pages 63

[116] J. N. Hooker, "Integer programming: lagrangian relaxation integer programming: Lagrangian relaxation," in Encyclopedia of Optimization, pp. 1667–1673, Springer, 2008. → pages 63

[117] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on mapreduce," Neurocomputing, vol. 102, pp. 52–58, 2013. → pages 63, 64, 106

[118] A. Basu, S. Shuo, H. Zhou, M. Lim, and G. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," Neurocomputing, vol. 102, pp. 125–134, 2013. → pages 63, 64, 106

[119] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological review, vol. 65, no. 6, p. 386, 1958. → pages 64

[120] G. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," Neurocomputing, vol. 71, no. 16, pp. 3460–3468, 2008. → pages 64, 67, 106

[121] W. Huang, Z. Tan, Z. Lin, G. Huang, J. Zhou, C. Chui, Y. Su, and S. Chang, "A semi-automatic approach to the segmentation of liver parenchyma from 3d ct images with extreme learning machine," in Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 3752–3755, 2012. → pages 64

[122] J. Zhao, Z. Wang, and D. Park, "Online sequential extreme learning machine with forgetting mechanism," Neurocomputing, vol. 87, pp. 79–89, 2012. → pages 64, 106

[123] Y. Ye, S. Squartini, and F. Piazza, "Online sequential extreme learning machine in nonstationary environments," Neurocomputing, vol. 116, pp. 94–101, 2013. → pages 64, 106

[124] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: theory and applications," Neurocomputing, vol. 70, no. 1, pp. 489–501, 2006. → pages 64, 106

[125] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in Proceedings of the International AAAI conference on weblogs and social media, 2014. → pages 67

[126] Q. Yu, M. Heeswijk, Y. Miche, R. Nian, B. He, E. Séverin, and A. Lendasse, "Ensemble delta test-extreme learning machine (dt-elm) for regression," Neurocomputing, vol. 129, pp. 153–158, 2014. → pages 67

[127] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: optimally pruned extreme learning machine," IEEE Transactions on Neural Networks, vol. 21, no. 1, pp. 158–162, 2010. → pages 67

[128] J. Spall, Introduction to stochastic search and optimization: Estimation, simulation, and control, vol. 65. John Wiley & Sons, 2005. → pages 68

[129] H. Liu and H. Motoda, Feature selection for knowledge discovery and data mining, vol. 454. Springer Science & Business Media, 2012. → pages 70

[130] U. Stańczyk and L. Jain, Feature Selection for Data and Pattern Recognition. Springer, 2015. → pages 70

[131] F. Benoit, M. Heeswijk, Y. Miche, M. Verleysen, and A. Lendasse, "Feature selection for nonlinear models with extreme learning machines," Neurocomputing, vol. 102, pp. 111–124, 2013. → pages 70

[132] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in Proceedings of the IEEE ICC, pp. 2889–2894, 2012. → pages 72

[133] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Technical Report NDN-0028, NDN, January 2015. → pages 72

[134] J. Friedman, "Stochastic gradient boosting," Computational Statistics & Data Analysis, vol. 38, no. 4, pp. 367–378, 2002. → pages 77

[135] A. Hyvärinen, J. Karhunen, and E. Oja, Independent component analysis, vol. 46. John Wiley & Sons, 2004. → pages 77

[136] W. Venables and B. Ripley, Modern applied statistics with S-PLUS. Springer Science & Business Media, 2013. → pages 77

[137] J. Hu, J. Zhang, C. Zhang, and J. Wang, "A new deep neural network based on a stack of single-hidden-layer feedforward neural networks with randomly fixed hidden neurons," Neurocomputing, 2015. → pages 77

[138] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85–117, 2015. → pages 77

[139] R. Quinlan, C4.5: programs for machine learning. Elsevier, 2014. → pages 77

[140] G. Zhang, T. Q. Quek, M. Kountouris, A. Huang, and H. Shan, "Fundamentals of heterogeneous backhaul design–analysis and optimization," IEEE Transactions on Communications, vol. 64, no. 2, pp. 876–889, 2016. → pages 85

[141] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, UK: Cambridge Univ. Press, 2004. → pages 96

[142] M. Zeni, D. Miorandi, and F. De Pellegrini, "YOUStatAnalyzer: A tool for analysing the dynamics of youtube content popularity," in Proceedings of the International Conference on Performance Evaluation Methodologies and Tools, pp. 286–289, 2013. → pages 96

[143] G. L. Nemhauser and L. A. Wolsey, "Integer programming and combinatorial optimization," Wiley, 1988. → pages 100

[144] D. Silva, "Two-group classification with high-dimensional correlated data: A factor model approach," Computational Statistics & Data Analysis, vol. 55, no. 11, pp. 2975–2990, 2011. → pages 104

[145] W. Griffiths, "A Gibbs' sampler for the parameters of a truncated multivariate normal distribution," Contemporary issues in economics and econometrics: Theory and application, pp. 75–91, 2004. → pages 104

[146] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in Proceedings of the International Conference on Machine Learning, pp. 1139–1147, 2013. → pages 105

[147] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification," Soft Computing, vol. 19, no. 6, pp. 1511–1522, 2015. → pages 105

[148] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," Artificial Intelligence and Statistics, pp. 192–204, 2015. → pages 105

[149] M. Chen, E. Zadok, A. O. Vasudevan, and K. Wang, "Seminas: A secure middleware for wide-area network-attached storage," in Proceedings of the ACM International on Systems and Storage Conference, pp. 1–13, 2016. → pages 112

[150] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, and F. Xia, "An experimental analysis on cloud-based mobile augmentation in mobile cloud computing," IEEE Transactions on Consumer Electronics, vol. 60, no. 1, pp. 146–154, 2014. → pages 112

[151] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," Technical Report NDN-0028, Revision 2, NDN, November 2016. → pages 112

[152] W. Cai, V. C. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," Mobile Networks and Applications, vol. 19, no. 2, pp. 144–152, 2014. → pages 129

[153] W. Cai, Z. Hong, X. Wang, H. C. Chan, and V. C. Leung, "Quality-of-experience optimization for a cloud gaming system with ad hoc cloudlet assistance," IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 12, pp. 2092–2104, 2015. → pages 129

[154] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in Proceedings of the IEEE GLOBECOM, pp. 1–7, 2009. → pages 129

[155] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval, pp. 539–548, 2016. → pages 131

[156] C. Gentile, S. Li, and G. Zappella, "Online clustering of bandits," in Proceedings of the International Conference on Machine Learning, pp. 757–765, 2014. → pages 131

[157] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in Proceedings of the IEEE Wireless On-demand Network Systems and Services, pp. 165–172, 2017. → pages 132