Reinforcement Learning for Content Caching and Crowdsourcing Leveraging Context Awareness

Xingchi Liu

A Doctoral Thesis submitted in partial fulfilment of the requirments for the award of the degree of

Doctor of Philosophy

Signal Processing and Networks Research Group,

Wolfson School of Mechanical, Electrical and Manufacturing Engineering,

Loughborough University

July 12, 2021

To my parents and my wife!

Acknowledgements

First and foremost I am extremely grateful to my PhD supervisor Dr. Mahsa Derakhshani, for her constant support and insightful guidance throughout my PhD study. I have always been inspired by her knowledge as well as her vision for high-quality research. From Dr. Derakhshani, I have learned how to do research efficiently and keep up to strive for the best. I feel very honoured to have been working under her supervision and I simply can't thank her enough for her invaluable patience and enthusiasm which helped me on my life and research career.

I also would like to thank Prof. Sangarapillai Lambotharan for guiding me throughout my PhD. His continuous support, suggestions, and deep expertise helped to develop my research skills. I also want to thank Dr. Ziming Zhu and Peizheng Li for their guidance and help during my internship at Toshiba Research Europe Ltd.

I am grateful for the studentship from the doctoral college of Loughborough University that allowed me to conduct this thesis. And I am also thankful to my colleagues and friends at Loughborough University: Shenhong Li, Ye Liu, Xinruo Zhang, Mao Ye, Xiaoyan Hu, Wenheng Zhang, Ashraf, and Tianrui Chen.

Last, but not least, I want to thank my wife, Dr. Shuang Qian, for her support and companion in the past ten years, which heartened me during the difficult times and helped me become a better person. Her love provided my inspiration and was my driving force. I am also thankful to my parents Yaoxiang Liu and Ping Wang, for their unconditional encouragement throughout my life. Thank you all for giving me the strength to chase my dreams.

Abstract

With the evolution of content demand characteristics and the emergence of crowdsourced streaming services, Internet video traffic including on-demand and live videos has grown explosively. This leads to new challenges for video streaming systems in terms of controlling the core network congestion and meeting the quality of service (QoS) requirements of users. To solve these problems, leveraging edge computing and storage resources has become a promising solution to enable content caching and transcoding. For on-demand videos, by caching popular contents at the network edge, the user requests for those contents will not be transmitted to the core network and hence can be served with less delay. Thus, the backhaul link load and the network congestion can be alleviated. However, how to determine the optimal caching placement under content popularity dynamics aiming to maximize the caching efficiency remains to be an open issue. On the other hand, to ensure the adaptive streaming of live videos, various formatting and quality versions need to be transcoded concurrently. Utilizing the abundant computational resources at the user end (UE) is a promising solution to provide adaptive streaming and meet the stringent latency requirements of those services. The goal of this thesis is to study reinforcement learning (RL) to solve online decision-making problems in content caching and video transcoding systems at the network edge leveraging the contextual information.

First, we study how to dynamically update the content placement at the edge server assuming the unknown and time-varying content popularity profile. The caching decision problem is modelled as a non-stationary Markov decision process (MDP) with varying states and transition probabilities. A context-aware pop-

ularity learning algorithm is designed to learn the time-varying file popularities via incremental clustering scheme. With the assistance of the learned knowledge, an RL-based content caching scheme is designed via state-action-reward-state-action (SARSA) and linear function approximation. Next, enlightened by the RL-based caching scheme, a reactive caching algorithm is proposed to reduce the computational complexity by directly comparing the popularities between the requested file and the cached files for cache replacement decision.

Secondly, an edge-assisted transcoding system is proposed for crowdsourced live streaming services, and a new quality of experience metric is defined which considers the influences from both the quality and the genre of the received live video. The transcoding task assignment and viewer association problem is formulated as a non-convex integer optimization problem aiming to maximize the network utility of the transcoding system, which is then solved by the computationally attractive complementary geometric programming (CGP).

Thirdly, a more complex edge transcoding system is studied taking into account the delay requirements of the viewers. To identify the risk of choosing highly unstable transcoders while learning the transcoding capabilities of transcoders, we first study to solve a risk-aware multi-armed bandit (MAB) problem with refined upper confidence bounds (UCBs) of the arms' variances. Based on the UCBs, a risk-aware contextual learning algorithm is designed to decide which transcoders are more stable and more efficient. In addition, an epoch-based transcoding task assignment and viewer association algorithm is proposed to maximize the network utility and maintain low transcoding task switching cost.

Finally, a structured bandit problem is studied to solve the transcoder selection problem from a different perspective. Here, assuming that there are performance correlations among multiple transcoders but the context information used to build the correlations is not available. To tackle the structured bandit problem which assumes the arm rewards are functions of globally shared parameters, an enhanced Thompson sampling (TS)-based algorithm is designed to sequentially select fog transcoders while handling the exploration-exploitation (EE) dilemma.

Contents

List of Figures

List of Tables

List of Publications

- X. Liu, M. Derakhshani and S. Lambotharan, "Joint Transcoding Task Assignment and Association Control for Fog-Assisted Crowdsourced Live Streaming," *IEEE Communications Letters*, vol. 23, no. 11, pp. 2036-2040, Nov. 2019.

- X. Liu, M. Derakhshani, S. Lambotharan and M. van Der Schaar, "Risk-Aware Multi-Armed Bandits with Refined Upper Confidence Bounds," *IEEE Signal Processing Letters*, vol. 28, pp. 269-273, Dec. 2020.

- X. Liu, M. Derakhshani and S. Lambotharan, "Contextual Learning for Content Caching with Unknown Time-Varying Popularity Profiles via Incremental Clustering," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 3011-3024, May. 2021.

- X. Liu, M. Derakhshani, Z. Zhu, and S. Lambotharan, "Globally Informative Thompson Sampling for Structured Bandit Problems with Application to CrowdTranscoding," *International Conference on AI in Information and Communication (ICAIIC)*, 2021.

List of Acronyms

5G	Fifth Generation
ABR	Adaptive Bit Rate
AR	Augmented Reality
ARA-UCB	Asymptotic Risk Aware-Upper Confidence Bound
BS	Base Station
CCN	Content-Centric Networking
CDN	Content Delivery Network
CP	Content Provider
CPU	Central Processing Unit
CGP	Complementary Geometric Programming
CHR	Cache Hit Rate
CLSP	Crowdsourced Live Streaming Platform
CMAB	Contextual Multi-Armed Bandit
CVaR	Conditional Value at Risk
D2D	Device-to-Device
DRL	Deep Reinforcement Learning
EE	Exploration-Exploitation
GI-TS	Globally-Informative Thompson Sampling
GRA-UCB	Gaussian Risk Aware-Upper Confidence Bound
HetNets	Heterogeneous Networks

ICN Information-Centric Networking

i.i.d. Independent and Identically Distributed

IP Internet Protocol

IRM Independent Reference Model

LFRU Least Frequent Recently Used

LFU Least Frequent Used

LFUDA LFU with Dynamic Aging

LR Linear Regression

LRU Least Recently Used

MAB Multi-Armed Bandit

MBS Macro Base Station

MDP Markov Decision Process

MV Mean-Variance

MV-LCB Mean-Variance Lower Confidence Bound

MV-UCB Mean-Variance Upper Confidence Bound

NN Neural Network

PMF Probability Mass Function

POP Popularity-Driven Caching

PPP Poisson Point Process

QoE Quality of Experience

QoS Quality of Service

RL Reinforcement Learning

RTMP Real Time Messaging Protocol

SARSA State-Action-Reward-State-Action

SBS Small Base Station

SNM Shot Noise Model

TD Temporal-Difference

TS Thompson Sampling

UCB Upper Confidence Bound

UE User End

VR Virtual Reality

Chapter 1

Introduction

1.1 Motivation: Video Streaming Systems

Over the last few years, with the advance of machine-type communications and explosive user demands for video sharing, online video services including the video content sharing platforms (such as YouTube, TikTok, Facebook, etc.) and the crowdsourced live video platforms (such as Twitch, Periscope, Huya, etc.) have become enormously popular. Using these platforms, professional content providers (CP) and independent users have been releasing the sheer amount of videos. Moreover, most videos in these services need to be offered with different quality versions. This further enlarges the increasing trend of video traffic. According to [3], Internet video traffic is expected to account for 82 percent of all business Internet traffic by 2022. This tendency enforces the network operators to upgrade video streaming system with adaptive bit rates (ABRs) by carefully allocating and scheduling computing, storage, and transmission resources.

There are mainly two types of videos in the video streaming systems. The first type is the on-demand video and the second type is the live video. There are several differences between the two types of videos. First, on-demand videos are relatively more delay-tolerant compared with live videos and can maintain popular in a period of time. Besides, only a part of files is frequently requested by the viewers. For instance, only one percent of the popular Facebook videos represents 83 percent of the total watch time [4]. Therefore, studying content popularity plays

an important role in on-demand video streaming. Furthermore, caching some on-demand contents in the network to serve the requests without directly querying the CPs can greatly improve the video streaming by serving requests with less delay and relieving the congestion of the core network.

However, for live video viewers, the delay requirement has become more stringent because the former live video chunks of a broadcaster can be outdated very soon and the viewers usually care much more about the latest video chunks. In addition, recent live streaming services have also enabled real-time chatting among viewers and broadcasters, which needs the live video to be streamed with very low latency for seamless chatting experience. Moreover, given the specific live video characteristics and requirements, caching the videos at network and servers is not a viable solution for live video streaming, which may bring extra delay and energy cost. In fact, due to the latency requirement of the live video streaming, providing viewers with multiple quality versions for a large number of live videos simultaneously becomes even more challenging than on-demand videos.

To solve the network congestion problems in the late 1990s and 2000s, content delivery network (CDN) was proposed as a highly distributed platforms of servers to serve the end users with the replications of contents cached in those servers and reduce the streaming delays experienced at the UEs. With the help of the CDN, user requests can be transmitted through the domain name servers of the CDN to the nearest CDN servers with the requested content [5]. The decision on which server is selected for replications storage is based on the constant monitoring and load balancing on data traffic in the network [6]. In the existing CDNs, in order to maintain a session for video streaming, the user should have the Internet protocol (IP) address of the CP. Another drawback of CDN is that most of the communication features such as mobility, security, and management are not built-in [7], while incorporating new services and features to existing video streaming system relies on complex protocol design which may fail at time.

To ensure the quality of experience (QoE) of users, traditional CDN faces various challenges. The problem emerges since many users can request for the same

content and make a large number of repetitive data streams being transmitted in the core network. This leads to the network congestion issue. Besides, to serve a large number of requests, the energy and bandwidth cost for the CP cannot be ignored. Moreover, since most users can be far from the CP, the resulting content delivery latency can drastically affect the users' QoE.

In recent years, with the proliferation of video sharing systems and social networks, end users are enabled to publish their own video contents and share them globally, which leads to new challenges such as congestion problems. Under this situation, the performance and scalability of the traditional CDN could not meet the requirement of the content demand evolution and thus, CDN faces challenges such as large network delay, heavy core network load, and huge network energy consumption. To solve the emerging challenges, the communication network is now shifting towards the information-centric networking (ICN) paradigm [8]. The main idea of ICN is to evolve the communication functions from the IP-based fashion to a content-based fashion. More specifically, the current Internet infrastructure which is a host-centric paradigm based on perpetual connectivity and the end-to-end principle can be replaced with a novel network architecture that focuses on 'named information' such as contents or files. Based on ICN, an instantiation called content-centric networking (CCN) [9] is designed to make content addressable and routable. Besides, the content-based security is adopted that secures the content rather than the communication channel. Therefore, the CCN architecture can make use of the storage of network nodes to achieve in-network caching thus improving the network performance, which is referred to as content caching. In Figure 1.1, a cache-enabled network architecture is presented.

Given all the advancements, there are still challenges in video streaming systems as follows:

- The content placement problem studying which contents to cache in network nodes are still challenging especially with popularity dynamics. This is because the content popularities can be unknown and highly dynamic in real-world scenarios. Besides, to reduce the experienced delays at the UE, recent

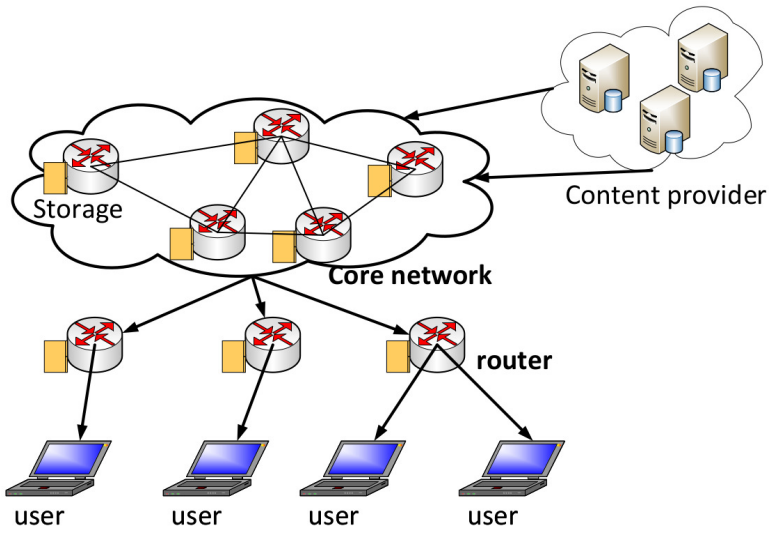

Figure 1.1: Architecture of a cache-enabled wired network

works study to proactively cache some contents to serve the user requests rather than serving the user requests reactively [10], which also needs accurate knowledge of the popularity dynamics.

- Due to the heterogeneity in the user network conditions such as user habits, the ABR service has played a significant role. By providing ABR service, multiple quality versions of the same video will be generated for various user preferences and network conditions to achieve adaptive streaming. However, implementing ABR service under the gigantic video contents requires massive computing resources, especially for live videos, which leads to high computational costs for the platforms.

- Finally, recent developments in edge computing have brought the additional possibility of improving the video streaming systems by shortening the distance and reducing the network delay between the CP and the users. For instance, we can utilize cache-enabled edge nodes to serve content requests. Besides, the computational resources of edge nodes can be applied to do transcoding, which can relieve the pressure of the core network and reduce the streaming delay. How to efficiently utilize the edge caching and computing has become a crucial challenge.

1.1.1 Content Caching

Content caching has been widely studied for the fifth generation (5G) communication networks in the past few years [11–13]. Content caching is the key technology of CCN architecture and enables nodes in the network to cache some popular content files to serve user requests. By utilizing this technology, nodes with caching capability can greatly reduce the delay experienced by the users since a large proportion of requests can be served locally without being fetched from the CPs. Besides, content caching will also reduce network traffic since much fewer repetitive files will be transmitted in the backhaul link thus alleviating the overall traffic load and reducing the power consumption.

In the on-demand video streaming system, content caching problems have been studied in the following directions. First, introduced by [14], the server placement problem has been investigated to study where to deploy the cache-enabled servers. Normally this problem is solved by minimizing either the users' latency, or the total bandwidth consumption, or an overall cost function. Similarly, there are also studies on the cache allocation problem that focus on determining how much caching capacity should be allocated to each cache-enabled server [15]. Finally, in recent works, with the knowledge of content popularity profile, the cache placement problem is discussed to decide which files should be cached at each server (e.g., [16]).

With the advances in wireless communications and the edge computing, content caching techniques at the network edge have become an interesting research topic [17]. For example, as presented in Figure 1.2, content caching at the base station (BS) and the user devices are potentially good choices to reduce the backhaul load of the network and the playback latency experienced by the users. However, caching at the network edge is fundamentally different from in-network caching and has raised new challenges. First, since the storage resources at the edge are highly limited as compared to the cloud data centres, how to efficiently utilize the cache capacity becomes a crucial issue. Second, at the network edge, the volume of requests for contents and files are much smaller and the request patterns and content popularity profiles are highly dynamic over time and location. These fea-

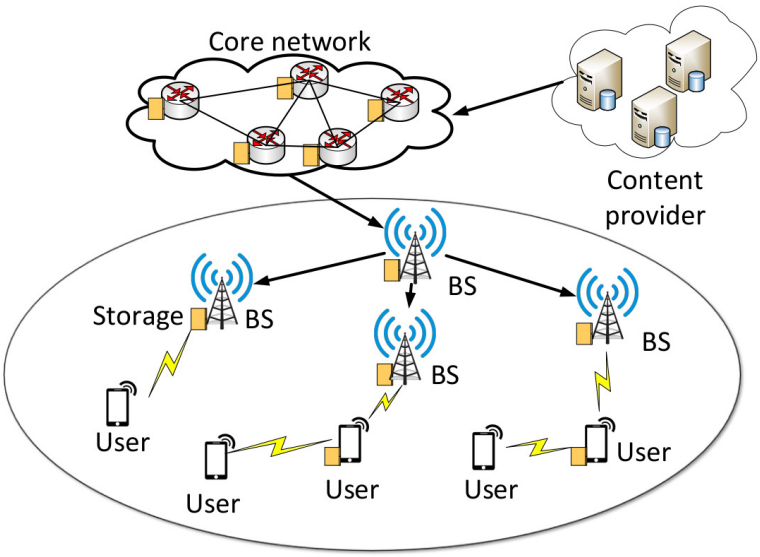

Figure 1.2: Architecture of a cache-enabled cellular network

tures, together with the inherent volatility of the wireless channels, make it highly challenging to identify the variation of the popular files and to optimize the cache placement scheme.

Classical content placement schemes such as least frequent used (LFU) and least recently used (LRU) are widely applied in current content caching systems and there are still some studies based on them (e.g., [18]) or use them as the benchmarks (e.g., [1], [2]). Applying LRU scheme requires to keep the request sequence number of each cached content, the node always tries to update its cache by replacing the least recently requested content (which has the smallest request sequence number) with the newly requested content. In the LFU scheme, the node always caches the most frequently requested content. However, the caching performance might drop drastically if the content popularity changes. The LFU can be treated as optimal under an independent reference model (IRM) which is described in Section 2.1.4.1. On the other hand, the LRU reaches the optimal competitive ratio when the requests are made adversarially. However, both schemes have limitations to handle non-stationary popularity profiles, as it is the case in practical caching networks. In addition, although it is possible to estimate the global popularity in systems such as the on-demand video streaming, the global popularity may not match the local popularity because the servers locating at the network edge can only serve a small

geographical area with limited requests, and the variation tendency of the global popularity and the local popularity can be totally different. These challenges motivate us to study the content caching scheme of an edge server under non-stationary and time-varying popularity profiles without any global information.

There are two types of study for content popularities. First, a large proportion of works employ the IRM to track the content popularities and synthesize the user request pattern assuming the user requests are made in an independent and identically distributed (i.i.d.) fashion from a predefined distribution [11]. On the other hand, there are many works discussing about learning the content popularity profiles and then optimizing the caching system. The social networks and the user mobility information are considered to learn the content popularities and various types of machine learning (ML) schemes are utilized. However, most of works either assume that the prior knowledge of the content popularities is known for the caching placement or alternatively, requires training sessions to learn the file popularities. These schemes need considerable time for learning and could incur an economic cost to acquire the data for popularity learning, otherwise, it can be outdated and deem to be invalid with non-stationary content popularity.

1.1.2 Edge Computing and Video Transcoding

The revolution of the mobile Internet driven by the powerful mobile devices and social networks has greatly enriched the sources of video platforms. As an outcome of the revolution, crowdsourced live streaming platforms (CLSP) such as Twitch, Huya, and Periscope have emerged as a new type of video platforms, that not only serve tremendous viewers all over the world but also receive live videos from various sources in the crowd [19]. Different from on-demand video sharing platforms, CLSPs have been allowing a growing number of people to broadcast their live videos over the Internet. Meanwhile, each viewer can directly discuss with the broadcasters and the other viewers via real-time chatting.

However, due to the heterogeneity of broadcasters' devices, different quality versions of live videos need to be created and uploaded to the CLSP [20]. As a result, there is a strong need to transcode the original live videos into several industrial

standard representations and to serve viewers with a set of proper versions of representations. To provide the ABR service, there are massive computational demands due to real-time processing requirements which need to be met. For instance, in 2020, there are about 3.84 million monthly broadcasters which are active on Twitch, and in average 56,000 of these broadcasters will broadcast concurrently [21]. In addition, according to the previous study [22], the number of online users may change dramatically over time. Therefore, instead of building private data centres to facilitate ABR, cloud computing has become a natural solution to perform transcoding because of its powerful computing ability and the 'pay as you go' feature. Furthermore, the emergence of cloud computing releases CLSP from building large, expensive private data centres. In Figure 1.3, a cloud computing-based live streaming system is illustrated. In such a system, the CLSP controller will decide the number of representations that need to be transcoded for each broadcaster based on parameters such as viewer capacity, playback delay, bandwidth consumption etc. The original live videos will be directly transmitted to the cloud data centre for transcoding. When multiple versions are generated in the cloud, CDNs will be utilized to deliver proper versions of live videos to the corresponding viewers.

On the downside, in current CLSPs, the cloud transcoding is not able to provide the ABR service to most of the broadcasters. For instance, in Twitch.TV, only the premium broadcasters have access to the ABR service, and for the rest of the broadcasters, only the original versions are available for their viewers [19]. The reason behind this is that a general cloud instance can only deal with at most two transcoding tasks simultaneously. Therefore, an enormous cost will be incurred when a large number of original live videos are planned for transcoding. Moreover, in cloud transcoding systems, the cloud data centre can be far from the viewers or the broadcasters. This will cause high latency. In addition, most of CLSPs support the broadcasters and viewers with interactive chat service. Under such a scenario, the latency problem has become even more significant than the traditional live streaming platforms.

The development of edge computing has brought a potential transcoding solu-

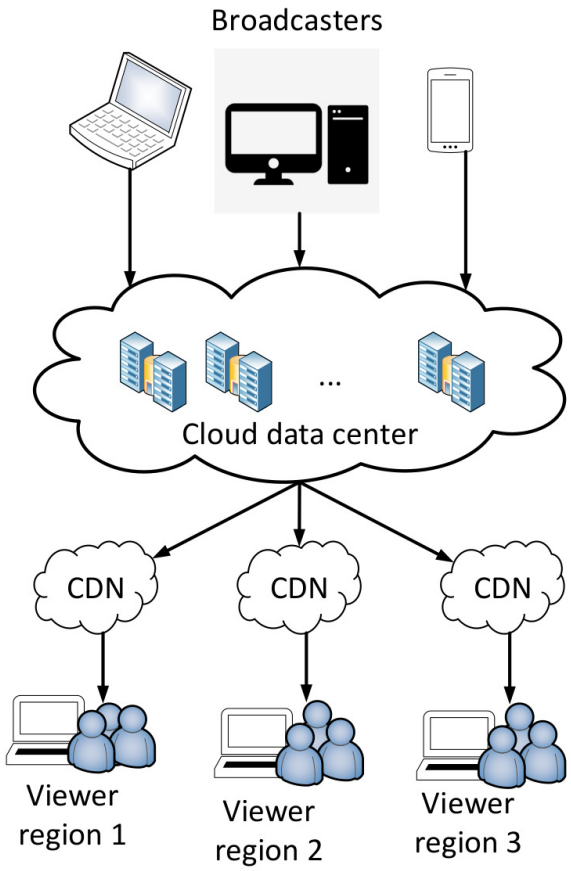

Figure 1.3: Architecture of a cloud transcoding assisted CLSP

tion for CLSP. Since edge computing [23] is more suitable for real-time processing and low-latency applications, it can be treated as a viable replacement (e.g., [24,25]) to address the weakness of the cloud transcoding. In [24], a case study is presented for Twitch.TV. This case demonstrates that with the advance of personal computing devices, a significant fraction of CLSP viewers potentially has appropriate computing resources for stable real-time transcoding. In addition, the viewers have already expressed the willingness to support the broadcasters and the CLSPs in terms of donation and subscription [26]. Thus, the cost by involving them into transcoding can be much lower. Moreover, edge-assisted transcoding can lead to lower latency and avoid the network traffic traversing through the core network since different versions of videos will be created at the network edge.

1.2 Edge Computing Resource Management and Reinforcement Learning

The development of edge computing has migrated data computation and storage to the network edge closer to the UEs [27], which can provide considerable computational and storage resources for delay-sensitive applications such as video transcoding and streaming.

However, efficient utilization of the resources at the network edge remains to be highly challenging since in problems such as cache replacement and transcoding task assignment, decisions need to be made in an online fashion with minimum prior information and latency. Although large-scale convex optimization [28] has provided powerful tools for edge resource management, in most of practical video streaming systems, there are parameters such as the file popularity [1] and computational stability [24] which are unknown before decision-making. Therefore, directly solving optimization problems is infeasible and extra learning processes are necessary to facilitate the large-scale optimization.

To enhance the decision-making process in edge-assisted video streaming system, online decision-making models and RL have been studied and applied. The challenge of solving the decision-making problem is how to balance the EE dilemma. If the agent only exploits the knowledge learned so far to make decisions, the opportunity of reaching better performance will be missed, however, if the agent keeps exploring the unknown environment to gain knowledge, obviously some poor decisions can be made, which can cause performance loss.

There are two well-known decision-making models which are MDP and multi-armed bandit (MAB). MDP is a discrete-time stochastic control process. It offers a framework which can model decision-making problems in situations where outcomes are partly random and partly under the control of a decision agent. The standard MDP is a 4-tuple $(\mathbf{A}, \mathbf{S}, P_a(s,s'), r_a)$ where

- $\mathbf{S}$ is the state space which is a set of process states.
- $\mathbf{A}$ is the set of possible actions a decision agent can take.

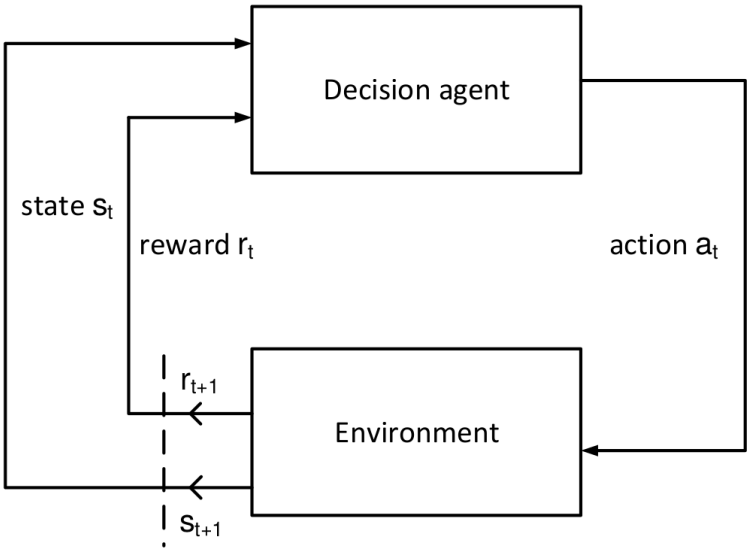

Figure 1.4: The agent–environment interaction in RL

- $P_a(s, s')$ is the transition probability that the state moves from s to s', given action a is taken.

- r_a is the immediate reward when action a is taken.

In Figure 1.4, in a specific state, the agent will take an action based on a selection policy, after that, a reward will be returned from the environment and the process will move to the next state with a transition probability. Based on the MDP, RL can be applied to train a policy to take optimal actions within an environment to maximize the returned cumulative reward. Particularly, the MDP is assumed to follow the Markov property which means that the future state transition is independent of the past states given the current state.

In some decision-making problems, the state of the decision process will not change with the taken actions. In this case, MDP is not suitable any more, therefore, MAB is proposed to model this type of problems. In an MAB problem, a machine (which is referred to as an arm) can instantly generate a random reward if it is pulled and an agent aims to maximize the received cumulative reward by sequentially pulling a number of arms. MAB can be applied into a wide range of applications including cognitive radio networks [29], reconfigurable antennas [30], crowdsourcing system [31], online recommendation [32], and dynamic hybrid net-

works [33] etc. In a standard stochastic MAB problem, a random reward of an arm can be observed when it is played by an agent and the reward of each arm is assumed to be independent of each other. The objective is to select a number of arms to play to maximize the cumulative rewards in a certain amount of rounds.

The objective of the standard MAB implies that the arm with the highest expected reward is always treated as the optimal choice. However, in reality, there are many applications for which not only the expected rewards but also the uncertainty of the rewards imposed by the variations are key when making online decisions. This is because an arm with a large variation may generate very low reward which can lead to unaffordable failure. For instance, in clinical trials, instead of directly taking a treatment which reaches the best average therapeutic result but may lead to an unacceptable poor result, a treatment that works consistently well for every patient is more reliable and hence more desirable. Therefore, for such applications, the tradeoff between the means and the variances of arms should be considered by studying the risk-aware MAB framework.

In the classical MAB problem, it is assumed that the reward of an arm follows an independent probability distribution. Therefore, the reward observed from one arm cannot reveal the reward information of other arms. This type of MAB can be categorized as the non-informative bandit problem and many algorithms are designed to solve this problem including UCB [34], TS [35] and ε-greedy [36] (The standard MAB algorithms are discussed in Section 2.3.1). However, this assumption does not hold in many applications. For example, in a news recommendation system, a company needs to decide which news to present to the users to maximize the click-through rate [37]. If we assume this model is non-informative, then the correlation among different users will be ignored. However, users with similar ages, occupations or levels of education may have similar news preference, which should also be considered to help solve the bandit problem. Therefore, the correlation among different arms should be taken into account in order to solve the MAB problem in a faster and more accurate way.

1.3 Technical Challenges

In content caching networks, understanding and predicting the dynamic time-varying popularity profile can greatly affect the performance of the caching system. There are many recent papers focusing on learning the popularity profile. Specifically, some works look at how a file's trending evolves [16] or utilizing social networks to predict the popularity profile [38]. More recently, ML technologies (such as bandit scheme [39], Q-learning [40] and transfer learning [41]) have also been utilized to learn the popularity profile. However, due to the non-stationarity of the popularity profile, it is still challenging to explicitly model the popularity dynamics for caching algorithm design. Besides, there are still requirements for accurate non-stationary file requests model, which can be beneficial for large scale analysis and evaluation.

For the emerging crowdsourced live streaming systems, given the explosive amount of independent broadcasters, how to ensure reliable ABR service while satisfying the cost and delay requirements is still in its infancy. In [24], the computational capability of viewer devices at the network edge has been tested and the results demonstrated that viewer devices are capable to complete real-time transcoding task. Thus, involving these abundant computational resources into transcoding may greatly relieve the burden of the CLSP in terms of delay and cost. However, the transcoding performance of these devices is unknown and can be highly dynamic. Thus, online learning schemes which identify the proper fog transcoders and the risk during exploration of the transcoders' performance should be considered. Moreover, although we can find that fog devices which show similar computational abilities and are from users with similar patterns may yield close transcoding performance, how to accurately measure and utilize the correlation to assist learning are worth studying. Besides, it is also challenging to design appropriate auction and rewarding methods to incentivize and recruit capable viewer devices for edge transcoding [26, 42].

1.4 Thesis Contributions and Organization

The remainder of this thesis is organized as follows. Chapter 2 presents a review of the related works in on-demand video caching and live video transcoding. We mainly discuss several aspects, such as RL-based content caching models, context-aware popularity-based caching schemes, cloud and edge-assisted transcoding system, and some recent works in decision-making problem, which considers the risk-sensitivity and correlated arm structure.

Chapter 3 studies the cache replacement problem at the network edge. We first propose a context-aware popularity learning algorithm to learn the time-varying file popularity profiles. Particularly, an incremental clustering algorithm is applied to requests with varying context information. This helps to obtain the similarities among requests, which enhances the learning rate and accuracy. After that, the caching decision problem is modelled as a non-stationary MDP. By invoking a linear function approximation, an RL-based content caching scheme is designed via SARSA. By incorporating the knowledge learned from the context-aware popularity learning algorithm, the RL is accelerated and the caching decisions are improved. Next, enlightened by the RL-based caching, a reactive caching algorithm is proposed to reduce the computational complexity. A rigorous theoretical analysis on the popularity learning performance is provided and the sublinear learning error over time is demonstrated. We prove that the proposed reactive caching scheme converges to the optimal caching scheme with an increasing number of requests and given true file popularity. Moreover, the time complexities of the proposed algorithms are shown to be competitively low, which enhances the scope of the algorithms for practical applications. Finally, multiple settings of time-varying popularity profiles are designed for performance evaluation, by simulating both the independent and temporal correlated file request processes. Numerical results confirm that both algorithms provide a more robust caching performance as compared to several solutions when the file popularities are unknown and time-varying.

Chapter 4 studies a joint transcoding task assignment and viewer association problem with edge computing. In this chapter, the transcoding success rates of the

fog devices are introduced and a new QoE metric is defined for the edge-assisted transcoding system by considering the effects of both the quality of the received video but also the genre of the video. Next, an optimization algorithm based on CGP is designed to solve the resultant non-convex integer programming. Finally, trace-driven simulations demonstrate that the proposed algorithm outperforms existing benchmark schemes and can dynamically decide the transcoding schedule over time.

Chapter 5 formulates an extended edge-assisted crowdsourced live video transcoding problem where the transcoding capabilities of the fog transcoders are unknown and dynamic. To learn the risk of choosing highly unstable transcoders while making decisions, given the risk-sensitivity of the studied problem, two risk-aware bandit algorithms are designed to balance the mean-variance tradeoff with refined UCB of the arms' variances. Based on the studies, a risk-aware contextual learning scheme is applied to estimate the transcoding capabilities of the fog devices. Combining the context awareness and risk sensitivity, a novel transcoding task assignment and viewer association algorithm is proposed. Moreover, to further reduce the switching cost which is incurred by assigning the same transcoding task to different transcoders, an epoch-based assignment strategy is designed. Finally, numerical results demonstrate that the proposed algorithm reaches a competitive network utility and significantly reduces the switching cost, as compared to the benchmark scheme.

Chapter 6 studies a different version of the fog transcoder selection problem. By considering the correlations of transcoding stability among different fog transcoders, a structured bandit problem is formulated by assuming that some parameters in the arm reward functions are shared by all the arms. To solve the formulated decision-making problem, we first build a confidence set which is a set of parameters whose expected reward is close to the empirical mean rewards of all arms, and then a novel technique is designed to estimate the true value of the unknown parameter based on the established confidence set. Moreover, a novel TS-based algorithm is proposed to handle the EE dilemma. Simulation results demonstrate

that the proposed globally-informative Thompson sampling (GI-TS) algorithm can solve the transcoder selection problem with a noteworthy improvement of the learning regret compared with the existing benchmarks.

Finally, chapter 7 concludes the thesis and discusses potential future research.

Chapter 2

Literature Review

Edge computing is a distributed computing paradigm that brings storage and computational resources to where it is needed, utilizing the collaborative multitude of end-user or near-user devices [43]. With the development of both on-demand and live video streaming services, the deployment of caching and transcoding at the network edge has become a cost-efficient and low-latency solution for video streaming. In this chapter, we provide a state-of-the-art review of the most relevant works in the field of content caching, video transcoding, and online decision-making in edge-assisted video streaming systems.

2.1 Content Caching

The caching placement problems have been investigated in wired networks, with applications such as CDNs, peer-to-peer networks, ICNs and IP-based television networks [44]. Given the NP-hard nature of the general form of caching placement optimization problems, previous works focused on designing heuristic or approximation algorithms for caching placement [45–47]. Nowadays, content caching at the network edge has attracted considerable attention and many works have been conducted on new caching frameworks and algorithms. In this section, first, a review of edge caching frameworks is provided. After that, we discuss the content caching strategies and ML-based edge caching schemes. Then we focus on research of edge caching exploring the case with dynamic popularities. This remains to be an open issue and we present the research problems under this chal-

lenging area.

2.1.1 Edge Caching Scenarios

Internet-based online video sharing services have gradually replaced the traditional video services and led to the user request revolution. The novel video specifications such as UHD and 4K have become more and more popular and hence significantly increased the bandwidth consumption per request. Moreover, normally multiple quality versions and encoding formats of the same video are generated. This has further raised the need for caching capacity in networks [11]. In addition, new challenges have arisen as novel types of services [48] (e.g., augmented reality (AR) and virtual reality (VR)). As a result, how to utilize the edge storage resources for content caching has become a crucial problem.

Edge-assisted caching is mainly studied in two scenarios, i.e., wireless cellular networks and device-to-device (D2D) networks. In wireless cellular networks, edge-assisted caching enables BSs to cache a number of files to serve the user requests. By involving BSs, some content requests can be served at the BSs without being transmitted to the core network or the content provider. Thus, the network congestion in the highly throughput-limited backlinks can be alleviated and the experienced service delay can be greatly reduced. Furthermore, with multi-BS cooperation, a BS can retrieve contents from its neighbour BSs thus improving the BS's ability to serve user requests in cellular networks. Wireless cellular networks caching are studied in both macro-cellular networks and heterogeneous networks (HetNets). For instance, in [49], a content placement method was designed to minimize the average download delay and in [50], user mobility was considered to design a content placement strategies for hitting probability maximization at BSs. In [51], a HetNet caching problem was studied. In this work, a group of users is served by a small base station (SBS) which is connected with a macro base station (MBS) and the MBS has access to the core network. Considering both the pre-download gain and the caching gain, a continuous optimization problem was formulated to minimize the energy cost of the system and to determine the optimal transmission and caching policies. In [52], the caching problem was studied in a

cluster-centric small cell network where SBSs are clustered into different hexagonal grids and SBSs from the same cluster can cooperatively serve users inside the cluster. The authors proposed a caching scheme in which a part of the cache is reserved for caching the most popular contents and the remaining part is used for cooperatively caching a fraction of less popular files to reach the largest content diversity.

In addition to caching at the BSs, caching files at the user devices to leverage D2D networks is also a potential solution for content delivery. In D2D networks, each user device is able to serve requests from its neighbours thus reducing the network delay. Besides, cache-enabled D2D networks can improve the area spectral efficiency while providing low backhaul cost. In this scenario, normally the total cache of devices is much larger than the library of files but each device can only cache very limited number of files. As a result, both caching decision and file delivery strategies should be considered when applying content caching in D2D networks. In [53], a spatial content placement method for D2D network was proposed considering the location information in content replacement. In this work, the distribution of device locations was modelled by Poisson point process (PPP) and each device can cache one file to serve a range of close users. The authors formulated an optimization problem aiming to maximize the density of successful reception by optimizing a file caching distribution. In [54], to maximize the data offloading ratio, the user mobility pattern was considered to design a mobility-aware caching placement scheme. This research demonstrated that user devices with a very low or high moving speed should cache the more popular files, while the rest of devices should cache less popular files to avoid duplication.

2.1.2 Content Caching Strategies

There are many studies about investigating content caching strategies in edge computing including deterministic, probabilistic and coded caching. In light of the partial knowledge of network and contents, deterministic caching schemes aim to deterministically design content placement strategies by solving cache placement optimization problems. The objective is to optimize some performance metrics

including cache hit rate (CHR), area spectral efficiency, backhaul link cost, system throughput and outage probability [55,56]. Deterministic caching can optimally determine which specific contents should be cached. In [57], a deterministic caching problem in a small-cell caching system was studied to maximize the localized system throughput. In this work, it is assumed that file $f \in \mathcal{F} = \{1,2,\ldots,F\}$ where $\mathcal{F}$ is the file library with the size of F. The file library contains all the files that the users may request. Moreover, it is defined that user $m \in \mathcal{M}$, where $\mathcal{M}$ is the user set. Subsequently, the deterministic caching problem is formulated as

$$\max_{b_f} \sum_{f\in\mathcal{F}} \sum_{m\in\mathcal{M}} p_f^m \times \left(b_f c_{1,m} + (1-b_f)c_{0,m}\right), \tag{2.1a}$$

$$\textbf{s.t.} \quad \sum_{f\in\mathcal{F}} b_f \leq Q_{\text{SBS}}, \tag{2.1b}$$

$$b_f \in \{0,1\}, \tag{2.1c}$$

where b_f indicates whether the content f has been cached by the SBS, p_f^m denotes the request probability of user m for f, and Q_{SBS} represents the storage capacity of the SBS. Moreover, $c_{1,m}$ denotes the channel capacity between the SBS and the user m and $c_{0,m}$ denotes the channel capacity between the MBS and the user m. In this work, it has been proved that the formulated optimization problem (2.1) is NP-hard. Given the NP-hardness of the problem [57], a deterministic caching algorithm based on RL was proposed to maximize the system throughput by optimizing the content placement in SBSs.

Probabilistic caching studies stochastically caching contents according to the content popularities (e.g., [58, 59]), assuming that each cache-enabled node has a probability mass function (PMF) for caching different content files. At each caching decision round, rather than deterministically caching the popular files, the nodes will fetch some contents from the CP according to the PMF, which could optimize a performance metric such as cache hit probability and successful download probability. Probabilistic caching can be used in random wireless networks with spatially distributed network nodes. Besides, probabilistic caching can serve user requests in

a proactive way, which means the network nodes can prefetch some contents before the arrival of user requests. In [60], a probabilistic caching placement in wireless D2D caching networks was studied. Define P_{hit} as the cache hit probability. Assuming that the locations of mobile user devices are modelled by a homogeneous PPP with intensity λ_u. In this system, each user has the probability $\rho \in [0,1]$ to make an active request for a file and the "inactive" devices will serve as potential D2D transmitters. Therefore, the distributions of receivers and potential transmitters (within the distance d) follow homogeneous PPPs with intensity $\rho\lambda_u$ and $(1-\rho)\lambda_u$, respectively. The probabilistic caching problem in a D2D network can be formulated as

$$\max_{\boldsymbol{q}} P_{\text{hit}} = 1 - \sum_{f\in\mathcal{F}} p_f(1-q_f)e^{-\pi(1-\rho)\lambda_u q_f d^2}, \tag{2.2a}$$

$$\textbf{s.t.} \quad 0 \le q_f \le 1 \quad f = 1,\cdots F, \tag{2.2b}$$

$$\sum_{f\in\mathcal{F}} q_f \le S, \tag{2.2c}$$

where $\boldsymbol{q} = [q_1,\cdots,q_F]$ represents the caching probabilities of file f and the second inequality reflects the caching storage limit. This formulated optimization problem was solved by applying the Karush-Kuhn-Tucker conditions to maximize the cache hit probability.

Noting the dichotomy of network traffic amount between peak and off-peak periods, coded caching (e.g., [61–63]) is another caching strategy which can be applied into the edge-assisted caching systems. In coded caching, during the off-peak traffic period, a central server carefully puts some contents in the user caches without the knowledge of users' future demands. Then, during the peak period, each content is partitioned into several coded fragments and then processed by certain coding methods (e.g., Raptor codes [64] or fountain codes [65]). To serve the users, coded caching utilizes the coding techniques where contents are first aggregated (encoding) and then forwarded to multiple users. At the UE, the aggregate message is decoded into different contents for specific users. This technique can increase network throughput and reduce delays by reducing the number of transmissions [66].

In addition, coded caching provides a novel method to mitigate network congestion during peak traffic hours by creating and exploiting coded multicasting opportunities across users [67].

2.1.3 Machine Learning Solutions for Content Caching

In the conventional caching scheme designs, optimizing the cache placement relies on prior knowledge such as content popularities and user mobility features. However, practically this information might not be fully and perfectly available and hence needs to be learned. Therefore, combining learning and content caching has become a promising research orientation since learning algorithms can be used to predict file popularities and user request patterns especially with the help of some extra side information. Both supervised and unsupervised learning have been studied in content caching algorithm design. Supervised learning algorithms such as linear regression (LR), neural network (NN), and deep learning have been used to predict the traffic levels and the content demand given some labelled data [68]. Unsupervised learning schemes such as clustering are used to group UEs into different sets based on the side information. Hence, the network nodes can predict the request patterns based on the entire set of UEs and cache the contents that serve the most UEs in the set. Besides, technologies such as transfer learning [69] and RL [70] have also been studied to estimate the content popularity and design cache placement schemes.

There have been many works focusing on offline learning in content caching. In [71], the NN was used for content popularity prediction by utilizing the learned popularity to decide which files should be cached at each BS. Authors in [18] collected not only the real request information from wireless access points and BSs but also the user mobility information and location information to design a geo-collaborative caching strategy, then, user request pattern is predicted through LR. In [72], DL was used to train an ML model to determine cache placement and viewer association instead of directly applying optimization in real-time caching or scheduling. However, the main drawback of these learning algorithms is that they work in an offline manner and hence need to collect data first before training a model

using ML algorithms. Since the data for offline learning is static, it could become outdated over time and the performance based on such offline learning schemes could be unpredictable.

Different from offline learning, online learning algorithms such as incremental clustering and RL have been utilized in caching algorithm design since online learning is more responsive to the dynamic content popularity. Besides LRU and LFU, there are many content replacement algorithms proposed in the literature to study how to efficiently update the cache online without offline training. For example, LFU with dynamic aging (LFUDA) [73] adds dynamic age to accommodate shifts of content popularities and punishes the access frequencies of older contents. Moreover, least frequent recently used (LFRU) [74] divides the cache into two parts and combines the benefits of LRU and LFU schemes. For schemes such as LFUDA and LFRU, although they can achieve competitive caching performance, the context information is not considered during the cache update. In addition, the parameters of the schemes need fine tuning to achieve good performance.

On the other hand, context-aware online learning schemes have been explored as a new strategy to enhance content caching algorithm design. In such schemes, context information of requests which is the side information reflecting the system conditions and features, is collected incrementally to help learning. In [75], each content request with context information was mapped to a point in a context space, and, by grouping different points via a grid-based clustering method, videos with similar context information maintain an average value for content popularity forecast in the future. Moreover, in [76], a probabilistic caching scheme is designed for D2D networks that utilizes context information of the user's request history, user similarity, and social ties to achieve reasonably well-optimized caching performance. Furthermore, [1] and [2] utilized context information by designing a grid-based partitioning method to group requests into different hypercubes incrementally. Then each received request with context information can be grouped into a unique hypercube and the average forecast popularity can be calculated for each hypercube based on the revealed real popularity of requests points in the hypercube

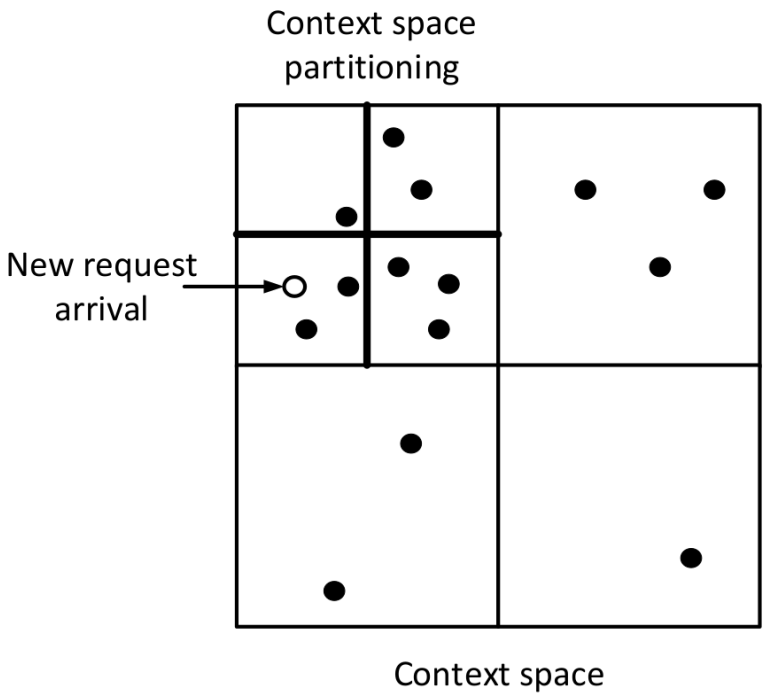

Figure 2.1: The context space partitioning method used in [1, 2]

(Figure 2.1 illustrates the adaptive partition process). When a file request arrives, the estimated file popularities of the requested file and the least popular file that has been cached will be compared, and the more popular file will be cached. The main drawback of this approach is that the partitioning method is not sufficiently flexible. Besides, these schemes do not consider the long-term effect of caching decisions. To improve the clustering process and to learn file popularities better, we propose a context-aware popularity learning scheme to group points into adaptive clusters based on the Euclidean distance via incremental clustering.

In addition, RL has been considered as an appropriate tool for caching algorithm designs due to its ability to solve online decision-making problems in an interactive environment rather than using only a fixed dataset. In [77], a Q-learning based caching policy was proposed to dynamically cache files at SBSs aiming at maximizing the CHR. The work in [78] leveraged RL to perceive the file popularities and to solve a proactive caching problem in wireless networks to minimize the average energy cost. In [79], a model-free RL algorithm was proposed to solve the caching problem for energy harvesting access point. In [80], a D2D caching scheme

was designed using multi-agent RL without the knowledge of content popularity profile. In [81], deep reinforcement learning (DRL) is utilized to dynamically decide D2D content delivery with the goal of minimizing the content delivery energy consumption and latency. In [70], an MAB problem was formulated to learn the file popularities and to maximize the caching reward. Moreover, in [82] and [83], caching decision problem was modelled as a contextual multi-arm bandit problem and RL-based algorithms were proposed to choose the file that should be cached.

2.1.4 Content Caching considering Popularity Dynamics

The file popularity profile plays an essential role in the content caching framework and algorithm designs since it can directly affect the cache placement and further influence other communication-related decisions in cache-enabled networks. The state-of-the-art studies in content caching mainly focus on the stationary file popularity profile which assumes the file popularity distribution is fixed over time. However, the practical popularity profile is always time-varying and highly dynamic. For instance, the requests for Wikipedia pages show a day-to-day variation in popularities, for instance, half of the top 25 articles change in only one day [84]. Therefore, understanding, tracking, and predicting the file popularity profile are essential for caching algorithm design and need further studying.

2.1.4.1 Independent Reference Model

There are two types of frameworks studying how to model the content requests in the real world. The first and the best-known model is the IRM [85]. IRM assumes that file requests are made in an i.i.d. fashion from a pre-defined distribution and the most commonly used distribution in content caching literature is the Zipf distribution [53, 86, 87]. The PMF of the Zipf distribution can be written as

$$P_f = \frac{f^{-\alpha}}{\sum_{f=1}^{F} f^{-\alpha}}, \tag{2.3}$$

where P_f is the popularity of content f. F is the file library size, and α is the skewness factor ($\alpha \geq 0$). A larger α means the popular file becomes more popular. When $\alpha = 0$, the Zipf distribution will be converted to a uniform distribution. The

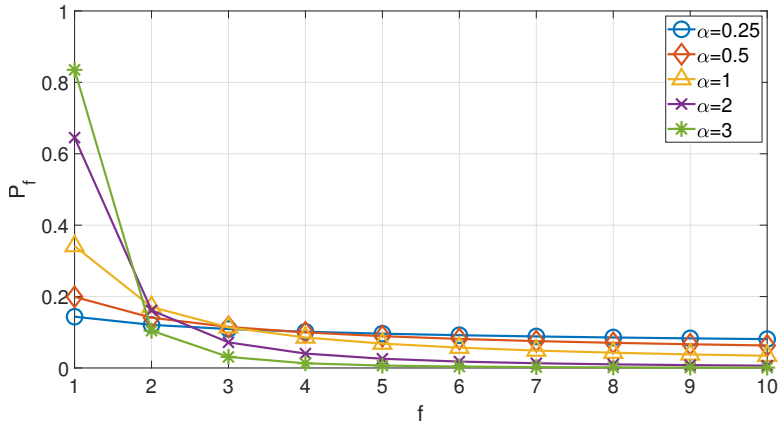

Figure 2.2: PMF of Zipf distribution when $F = 10$

PMF of Zipf distribution when $F = 10$ is presented in Figure 2.2 for different α values.

On the downside, although IRM can provide a tractable file request model and can accurately model the file popularity profile in a short time period assuming the profile is fixed, it cannot characterize the non-stationarity of the file popularity in a long time. Moreover, the so-called spatial locality and temporal locality among requests cannot be captured by the IRM [88].

There are some extensions to add non-stationarity and temporal correlation file requests into IRM. In [89], an evolution law of time-varying content popularity was proposed based on the Ornstein-Uhlenbeck process [90]. Moreover, in [86], the IRM was extended to make an effort to capture the correlations among requests. Two request models were proposed as Bernoulli and Poisson request models. These models assume the requests of users are first generated according to Bernoulli or Poisson distribution, then the file of the request will still follow Zipf distribution. The authors demonstrated that these models can generate requested files which follow a non-stationary dependent random process. Besides, in [86], to model the time-varying file popularity, a popularity permuting mechanism was introduced to change the popularity profile periodically.

2.1.4.2 Shot Noise Model

In [91], a new file request model was proposed as a viable substitute of IRM to capture macroscopic factors related to content popularity dynamics. Instead of

modelling the independent request, the shot noise model (SNM) represents the overall request process as the superposition of many independent processes (shots), each referring to an individual content. The request process for a given content f can be described by an inhomogeneous Poisson process, with the instantaneous rate ϕ_t at time t as

$$\phi_t = V_f h_f(t - \tau_f), \tag{2.4}$$

where V_f denotes the average number of requests for content f and τ_f is the time instant at which the content becomes available in the caching system. Moreover, $h_f(t)$ is the request profile which describes how the request rate for content f evolves over time. There are also refined SNM models which are discussed in [92, 93].

In addition to the discussed methods, there are some other works about simulating the popularity dynamics. In [40], the global popularity profile is modelled using a two-state Markov chain where the states are drawn from Zipf distributions with different skewness factors. In this work, the popularity profile is non-stationary and can change following the Markov rule. In [94], the correlations between the users and contents are explicitly modelled by kernel functions, so that the requests for the same content from different users are correlated.

2.2 Video Transcoding

With the revolution of the Internet which is driven by social networks and powerful devices, a new type of video platform called CLSPs such as Twitch, Huya, and Periscope has emerged. There has been a tendency that more and more people start to join such platforms and broadcast their live videos. However, because of the heterogeneity of broadcasters' source contents, to ensure the QoE of viewers, the original live videos need to be transcoded into several industrial standard representations before they are consumed by the viewers [20]. In this section, we review the works in both cloud and edge-based live video transcoding systems and discuss the challenges in designing the edge-based transcoding.

2.2.1 Cloud Transcoding

Due to the powerful computing ability and the 'pay as you go' feature of cloud computing, previous works tended to implement the transcoding system with the help of the cloud computing resources and designed various QoE metrics for cloud transcoding systems. For instance, [95] collected data from Twitch.TV including some information about the broadcasters. By analysing the data, they found that both the broadcasters and the viewers are heterogeneous across network conditions and regions. Therefore, a cloud-based scheme was designed to transcode crowdsourced video contents. In this scheme, the comprehensive cost (which was defined as the cost minuses the viewer QoE) was minimized. This QoE is a function of the bit rate of the received live stream and the broadcaster's popularity. [96] proposed a cloud-based transcoding scheme considering delay constraints. In this work, the QoE and the cost of the whole system were optimized with the constraint of transcoding and transmission delays. In addition, the defined QoE was a non-decreasing concave function of the received bit rate. [97] designed a multi-view crowdsourced live streaming framework that consists of multiple video streams captured simultaneously from different visual angles. In this work, a cloud transcoding scheme was proposed, and the QoE was related to the viewer's network condition and the received quality of each live video. In [98], a new crowdsourced live streaming framework was designed to minimize the content delivery delay with cloud transcoding. [99] proposed a cloud transcoding scheme for both delay-tolerant and delay-sensitive videos with different priorities. In order to minimize the cost of the transcoding system, a neural network was trained to predict the arrival rate of each video. In [100], a multi-content delivery network-based cloud transcoding framework is designed which enhances the CLSP's capacity and reduces the operational cost.

2.2.2 Edge Transcoding

Due to the abundance of concurrent live broadcasters and the heterogeneity of source contents, a substantial amount of transcoding tasks are generated which are delay-sensitive and computationally intense. As a result, even cloud transcoding

cannot meet these requirements with affordable cost [26]. Therefore, edge computing has been considered as a viable replacement for cloud transcoding because of the fast processing and quick application response time [101]. However, it is highly challenging to achieve optimal transcoding task assignment and viewer association due to the massive heterogeneous video contents and diversified QoE demands [102].

In [103], a collaborative joint caching and transcoding scheme was proposed to reduce the backhaul link usage and the viewer perceived delay. In [104], an RL-based scheme was designed to solve the edge transcoding decision and wireless spectrum resources allocation problems. To better schedule edge transcoding under large state space, DRL was used to explicitly accommodate personalized QoE optimization for crowdcast services with edge transcoding [102]. In addition, [26] combined both the cloud and the edge resources to collaboratively transcode live videos from multiple broadcasters. In this thesis, we propose an edge transcoding and content delivery algorithm for the CLSP that aims to optimize the tradeoff between the QoE of viewers and the cost incurred to edge nodes for video transcoding.

In the discussed works, although the system dynamics were considered in designing transcoding schemes, the performance uncertainties of transcoders are ignored, assuming that the fog device can complete the assigned transcoding task deterministically.

However, in real-world applications, the transcoder's performance may be unknown and unstable. For example, in [24], a case study was presented based on Twitch.TV demonstrating that a significant fraction of CLSP viewers have potential computing resources for real-time transcoding and since they are very close to the viewers, the latency can be greatly reduced. However, the viewer devices can be offline or leave the CLSP since they are not professional workers and this will lead to transcoding failures. To solve this problem, [25] found that the transcoding stabilities of viewers are proportional to their existing online durations. Thus, solving an optimization problem that maximizes the mathematical expectation of serving time of the viewer device, an optimal waiting threshold was acquired to select stable

viewer devices for transcoding. In this work, the authors only considered the online stability of viewer devices which is one of the factors that can affect the transcoding performance. However, there exist other factors that will have an impact on the performance, such as the available computational resources and the RAM usage. Moreover, in [25], the transcoder selection relies on the prior data collection and processing, which may become inaccurate over time. Therefore, a novel strategy which can do transcoder selection in an online form by considering multiple factors is highly desirable.

These studies demonstrate the potential of incentivizing the viewer devices to do transcoding. However, since the viewer devices are not professional devices and can be highly heterogeneous, an optimal decision-making strategy is needed to learn devices' performance and select devices which are more capable for transcoding. Such an edge-assisted transcoding system is similar with the crowdsourcing system which exploits the collective intelligence of crowd, provides an effective paradigm for large-scale data acquisition and distributed computing [105, 106]. In the edge-assisted transcoding system, the viewer devices assigned with transcoding tasks can be treated as the crowd workers for distributed computing. The crowdsourcing system has been introduced into many areas such as text translation, consumer research, and hiring workers to develop complex software.

There have been extensive works on task assignment problems in the crowdsoucing systems [107]. As a classical decision-making model, MAB have been used to model this type of problems, and bandit algorithms are used to solve the task assignment problems. For instance [108] proposed a UCB-based task assignment algorithm with a limited budget for crowdsensing. In [109], a spatial crowdsourcing system was considered and a contextual UCB-based algorithm was designed to maximize the number of assigned tasks. [31] modelled the crowdsourcing system as an MAB and proposed a bounded ε-first algorithm (which is a budget-aware greedy based algorithm) to maximize the overall utility of completing a number of tasks. [110] proposed a budget-limited UCB-based greedy approach to learn the worker performance of a crowdsourcing system and to select workers with high per-

formance to maximize the long-term utility. In [83], a hierarchical context-aware learning algorithm is proposed to learn and estimate the worker's context-specific performance in mobile crowdsourcing.

Overall, although the task assignment problem in crowdsourced systems has been studied in recent years, several technical problems have not been addressed. First, the context information of the transcoders has not been efficiently utilized. This can help to accelerate the learning of performance knowledge by explicitly building the relationship between the context information and the performance. In addition, the studied task assignment decision-making model in crowdsourcing system is not practical enough since the tasks are assumed to arrive sequentially so that standard MAB algorithms can solve the problem. Although some papers studied to assign multiple tasks at the same time, the number of tasks per time slot are fixed (which can be solved by combinatorial MAB algorithms). Moreover, in studied task assignment problems, most works only focus on identifying the arm with the highest mean reward. However, problems such as the risk of selecting poor arms with high performance variation and the switching cost of assigning a task to different arms have not been considered yet.

2.3 Online Decision-Making Problems

In this section, we describe some of the RL approaches such as MAB-based schemes. Moreover, we also discuss the state-of-the-art works in extended MAB models that consider the arm correlation and risk sensitivity.

There are works in edge-assisted online decision-making systems that employ standard MAB framework to model the decision-making problem in which each arm can be the decision for edge device selection (e.g., [31, 108]). Most of these works assume that different arms are independent with each other, hence, to learn the reward information of a specific arm, the agent has to play that arm. However, this assumption cannot capture the possible correlation among arms, since arms with similar context information can perform similarly. For example, in edge-assisted transcoding task assignment problem, different viewer devices with similar

hardware configurations might show close transcoding delays. By capturing the correlation among arms, the performance of the under-explored arms can be estimated given learned knowledge and the decision-making process can be improved.

Besides, to identify the best arm, standard MAB algorithms only take the mean reward into account. Other factors such as the risk and performance variations are ignored. In the edge-assisted transcoder selection problem, choosing a fog transcoder which performs averagely well may not be enough, since its performance can be highly unstable. Assigning transcoding task to such devices cannot provide viewers with stable QoE and will lead to frequent task reassignments.

2.3.1 Standard MAB

In a standard MAB problem, assume there are K arms, i.e., $\mathcal{K} = \{1, 2, \ldots, K\}$. In each round, an arm is played by an agent and a random reward is returned. The reward of an arm i at round t is sampled from an independent distribution with the mean reward defined as μ_i. The objective of the agent is to maximize the cumulative reward in a number of rounds. Define the number of rounds that arm i has been played during the T rounds as $n_i(T)$, the learning regret which is the expected cumulative reward difference between the designed algorithm and the optimal policy can be defined as

$$\mathrm{Reg}(T) = \mu_{i^*} T - \sum_{j=1}^{K} \mu_j \mathbb{E}\left[n_j(T)\right], \tag{2.5}$$

where i^* is the optimal arm that can be defined as

$$i^* = \underset{i \in \mathcal{K}}{\operatorname{argmax}}\, \mu_i. \tag{2.6}$$

To maximize the cumulative reward, different bandit algorithms have been proposed which can be categorized into two types. The first type is the index-based schemes, and the most classical scheme is the UCB scheme [111]. In this scheme, at the beginning of round t, according to the reward history, an index $I_i(t)$ will be

assigned to arm i as

$$I_i(t) = \bar{\mu}_i + \sqrt{\frac{2\ln t}{n_i(t)}}, \tag{2.7}$$

where $\bar{\mu}_i$ represents the empirical reward of the arm i. After assigning the index, the arm with the highest index will be played in this round. In the index, the first term is simply the current average reward which can be treated as the exploitation of the existing knowledge. The second term relates to the size of the confidence interval for the average reward, which can be treated as the exploration of the arm's uncertainty, combining the two terms, a UCB of the mean reward can be derived as the most optimistic estimation of the possible reward, which balances the EE tradeoff.

The second type of bandit algorithms is Bayesian algorithms such as TS [35]. Compared with UCB, TS is a randomized Bayesian algorithm that chooses an action with the same probability that the action is optimal. In TS, a prior distribution of the reward is placed for each arm. Before the arm selection, instead of assigning an index to an arm, a random sample is drawn from the corresponding distribution for each arm, and the arm with the highest sample will be played. After that, the prior distribution of the played arm will be updated according to the returned reward and the Bayes' theorem. In TS, the EE tradeoff is balanced implicitly. The mean value of the distribution is maintained from the historical rewards, which can be treated as exploitation. However, since the samples are drawn from the distribution, randomness is introduced for exploration.

Both UCB and TS are near-optimal in statistical MAB problems. UCB is more popular since strong theoretical guarantees on the learning regret are proved and the computational complexity is competitively low. The extensive experimental evaluation carried out in [112] revealed that TS is a very effective and can outperform UCB in some settings. However, to maintain and sample from a posterior distribution over models, TS is more computationally onerous [113].

2.3.2 Risk-aware MAB

As compared to the standard MAB, the risk-aware MAB considers the performance variance while playing arms. There are multiple approaches to measure the reward uncertainty in a risk-aware MAB. One of the mainstream measures to balance the tradeoff between maximizing the expected reward and minimizing the uncertainty of the reward is the mean-variance (MV), which is proposed in [114]. This paradigm considers a linear combination of the mean and the variance of the reward when determining the optimal arm. The MV metric can be presented as

$$\eta_i = \sigma_i^2 - \rho \mu_i, \tag{2.8}$$

where μ_i represents the mean and σ_i^2 denotes the variance of the reward of arm i. In (2.8), $\rho \geq 0$ is a risk-tolerance factor which is introduced to balance the tradeoff between high reward and low risk. As $\rho \to \infty$, the risk-aware MAB problem degenerates to a standard risk-neutral MAB problem, and when $\rho = 0$, the problem becomes a risk-only MAB which aims to find the arm with the lowest variance.

In [115], an algorithm was designed based on the MV paradigm to minimize the cumulative learning regret by deriving a lower confidence bound of the MV. The proposed mean-variance lower confidence bound (MV-LCB) algorithm achieves a $\mathcal{O}(\log^2 T)$ learning regret which is worse than the learning regret of the classical risk-neutral MAB algorithms. In [116–118], finer analyses of the theoretical performance of MV-LCB were presented, and a new definition of the cumulative learning regret was derived for the MV measurement. By extending the MV-LCB, a new algorithm called mean-variance upper confidence bound (MV-UCB) was designed and proved to reach a $\mathcal{O}(\log(T))$ learning regret. However, the regret bound only holds for a limited class of reward distributions.

Another way to measure the risk is to use conditional value at risk (CVaR). To be more specific, instead of using all the returned rewards to estimate the variance of each arm, CVaR only focuses on the poor rewards and uses these rewards to estimate the uncertainty of each arm. In [119], the arm quality is set to its CVaR which equals to the average of a number of the lowest rewards, and the arm with the

maximum CVaR will be played. CVaR has also been studied in [120–122]. Besides, the risk-averse MAB is also studied in the non-stochastic MAB framework [123] and explore-then-commit MAB [124].

2.3.3 Structured MAB

In the structured MAB [125], instead of assuming the reward distributions of arms are independent of each other and estimating the performance of each arm only based on the arm's own reward history, the potential correlation among arms are taken into account to solve the MAB problem in a faster and more accurate way.

Contextual multi-armed bandit (CMAB) can be categorized as a type of structured MAB problem and it is a promising model to handle the correlation of arms because by playing one arm, the reward information of the arms with similar context information can also be learned. In [126], a CMAB problem was proposed which assumes the expected reward of an arm is a linear function of the arm's contextual information and a set of unknown parameters. Then, an algorithm called LinUCB was designed to solve the problem by learning the unknown parameter vector of the reward function. In [127], a TS-based algorithm was proposed to solve the CMAB problem with the same assumption that the expected reward function is a linear combination of parameters and context information. In [128], instead of explicitly assuming the mapping from the context to the reward to be a linear function, a scheme called GP-UCB was proposed to use Gaussian process to model the relationship between the context and the reward. Moreover, in [129], the arms were clustered into different groups based on context information, and each group of arms share the same parameter of the reward function.

Under some circumstances, due to the privacy problem, the context information cannot be collected by the agent for arm selection but the reward mappings as a function of this hidden context are known [130]. In this case, the expected reward of each arm can be assumed as a function of one or multiple common unknown parameters, and a different type of structured bandit problem is studied. For example, in a dynamic pricing problem, an agent sequentially selects a price (arm) from a finite set of prices to maximize the cumulative revenue without the information of

the market size (unknown parameter). In this problem, the expected revenue is a function of the selected price and the market size. The forms of the reward functions are typically known [131] but the market size which is shared by all the arms are unknown. Therefore, each time a price is selected, and the resulting reward is observed, the market size can be estimated for future price selection. In [132] and [133], the globally-informative bandit problem was studied, which assumes the forms of the expected reward functions are known and the unknown parameter is shared by all the arms. In these works, the unknown parameter is estimated by directly solving the reward function with empirical mean reward. However, the proposed algorithm in [132] was limited to only one unknown parameter and [133] only considered monotonic reward functions. In [134], a regional bandit problem was modelled which assumes different groups share different parameters. In [135], a unified approach was designed to translate classical bandit algorithms such as UCB and TS to the structured bandit setting.

Chapter 3

Contextual Learning for Content Caching with Unknown Time-Varying Popularity Profiles

3.1 Introduction

The problems identified in both the classical and novel caching schemes reemphasize a practical need for a robust online caching scheme to provide stable caching performance under time-varying content popularity profile. In this chapter[1], we model a realistic caching decision problem in an edge server as an MDP and strictly demonstrate how to learn the popularities of contents online and use the learning results to improve cache management (caching decision).

We first design a context-aware popularity learning algorithm to track the time-varying file popularity which is then used in RL to solve the MDP for dynamically updating cached contents in edge servers. Moreover, a reactive caching algorithm is studied to reduce the complexity of the RL-based caching scheme. Our algorithms require neither any prior knowledge about the users and their content requests nor any training sessions, which may be inaccurate, outdated, and expensive to obtain. The theoretical analysis proves that the learning error of the context-aware popularity learning scheme only grows sublinearly with the increasing of file requests.

[1]Part of Chapter 3 has been published in the IEEE Transactions on Communications [136].

Moreover, by utilizing the popularity learning scheme, the CHR of the reactive caching scheme is proved to converge to the optimal CHR as compared to the optimal cache scheme with full knowledge of popularity profile.

To simulate the time-varying popularity profile, we build various types of simulation. First, the IRM is utilized to generate the file request. To add the popularity dynamics, the file library and the popularities of a part of files are randomly changed periodically. Furthermore, to simulate the temporal locality of file requests, the SNM is utilized. The simulations with varying settings demonstrate the performance and robustness of the proposed algorithms.

The rest of this chapter is organized as follows. Section 3.2 describes the system model. The context-aware popularity learning algorithm is presented in Section 3.3. The detailed RL-based caching algorithm and the reactive caching algorithm are described in Section 3.4, followed by their theoretical performance in Section 3.5. The simulation setup and results are presented in Section 3.6, and the conclusions are drawn in Section 3.7.

3.2 System Model

We consider a content delivery network where a CP has a library of files, i.e., $\mathcal{F} = \{1, 2, \ldots, F\}$. The number of files F may be very large and caching is a viable solution to improve quality of service. We assume a cache-enabled server at the network edge, the caching capacity of which is M representing the maximum number of contents that can be locally stored. We focus on caching decision problem for a single server to design a decentralized caching scheme and to maximize the CHR of each node independently. Without loss of generality and for the sake of simplicity, we assume the file sizes are identical. Consider req_n as the n^{th} content request made by the users, where $n \in \mathcal{N} = \{1, 2, ..., N\}$. Each request is represented by a 3-tuple as $\text{req}_n = <f_n, t_n, v_n>$, where $f_n \in \mathcal{F}$ is the file being requested, t_n is the time that the request was made, and v_n is the context vector associated with the request. Generally, the context data is a d-dimensional vector and each element represents one context information.

In this section, we model the caching decision process as a non-stationary MDP, which is an appropriate mathematical framework to model sequential decision making considering the random dynamics of the system under study [137]. MDP formulation is useful to model and study the long-term effect of each caching decision on the CHR. In this problem, the server is an agent who decides whether and how to cache the requested file to maximize the long-term CHR. In particular, at each time step $n \in \{1,2,...\}$, the server picks a caching action a_n from the action space $\mathcal{A}$ given the current state of the system $g_n \in \mathcal{G}$, which is the state space. Given the current action and state pair (g_n, a_n), the server moves to some state g' with the probability of $\Pr(g'|g_n,a_n)$ and receives a reward $r_n(g_n,a_n)$ [138]. In the following, the action space, states, transition probabilities, and reward function are defined in detail.

3.2.1 Action Space

When a request arrives, the server checks whether it can be served locally. No change is required if the file is available in the cache. Otherwise, the requested file will be retrieved from CP, and the server decides whether and how to update its cache taking into account the possibility of future requests. Since we assume a time-varying popularity profile in this work, replacing the least popular file in the cache with the requested file is not always optimal. In such a case, it is imperative to take precautions and explore all caching possibilities to some extent to balance the exploitation-exploration trade-off and track time-varying popularity profile. Therefore, the server should be able to cache the requested file by replacing any of the files currently in the cache. Given M is the cache capacity, this implies that the action space of the server has $M+2$ possible actions as $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_{M+2}\}$ where

- α_1: The server caches the requested file by replacing the 1$^{\text{st}}$ popular file in the cache.
- α_M: The server caches the requested file by replacing the M^{th} popular file in the cache.

- α_{M+1}: The requested file is retrieved from CP but not cached.
- α_{M+2}: The server serves the requested file locally without changing its cache.

The server chooses between the first $M+1$ actions if the requested file is not cached currently. Otherwise, if the requested file is locally available, the last action α_{M+2} is invoked.

3.2.2 State Space and Transition Probability

The state of the process is determined by the currently cached files and the requested file. In particular, the state at time-step n is defined as $g_n =< \mathcal{S}_n, f_n >$, where $\mathcal{S}_n$ represents the set of cached files and f_n represents the requested file at time-step n. More specifically, suppose the set of cached files is $\mathcal{S}_n = \{s_n^1, s_n^2, \cdots, s_n^M\}$, in which the cached files are indexed in a decreasing order of the file popularities and $\mathcal{S}'$ as the subsequent cache state after taking an action. Moreover, define $f' \in \mathcal{F}$ as the next requested file.

Let us define the transition probability from state g_n to state g'. When an action is taken, the cache state transfers from $\mathcal{S}_n$ to $\mathcal{S}'$ deterministically. Therefore, the transition probability depends solely on the file popularity of the next requested file f' at time step n, which is defined as $P_{f'}(t_n)$.

If the requested file $f_n \notin \mathcal{S}_n$ (i.e., cannot be locally served) and $\alpha_m \in \{\alpha_1, \alpha_2, \cdots, \alpha_M\}$, the transition probability can be written as

$$\Pr(g'|g_n, a_n = \alpha_m) = \begin{cases} P_{f'}(t_n), \text{ if } g' =< s_n^1, ..., s_n^{m-1}, f_n, s_n^{m+1}, ..., s_n^M, f' >, \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.1}$$

If the requested file $f_n \notin \mathcal{S}_n$ and action α_{M+1} is taken, the requested file will be served by the CP without replacing the cached files, namely $\mathcal{S}' = \mathcal{S}_n$. In this case, the transition probability is

$$\Pr(g'|g_n, a_n = \alpha_{M+1}) = \begin{cases} P_{f'}(t_n), \text{ if } g' = <s_n^1, ..., s_n^M, f'>, \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.2}$$

Otherwise, if $f_n \in \mathcal{S}_n$, the requested file can be served locally and α_{M+2} will be taken. In this case, the cached files will not be replaced and action and the transition probability is

$$\Pr(g'|g_n, a_n = \alpha_{M+2}) = \begin{cases} P_{f'}(t_n), \text{ if } g' = <s_n^1, ..., s_n^M, f'>, \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.3}$$

The sizes of the state space and the transition probability matrix depend on the file library size and the cache size of the server. In this regard, the size of the state space is $F \times \binom{F}{M}$ and the size of the transition probability matrix is $F^2 \times \binom{F}{M} \times \binom{F}{M}$.

3.2.3 Reward Function

In MDP, the reward is the return of the process after transferring from one state to another under a taken action. The CHR is an appropriate metric to evaluate the performance of content caching schemes. The CHR H can be calculated as $H = N_{\mathrm{L}}/N_{\mathrm{Total}}$, where N_{L} is the number of locally served requests and N_{Total} is the total number of received requests.

Caching relatively popular files can increase the CHR by locally serving more requests. Therefore, we consider the file popularity when defining the reward. To prioritize caching relatively popular files, the reward function is defined as

$$r_n(g_n, a_n) = \begin{cases} P_{f_n}(t_n) - P_{s_n^m}(t_n), & \text{if } a_n = \alpha_m, \ \forall m \in \{1, 2, \cdots, M\} \\ 0, & \text{if } a_n = \alpha_{M+1} \\ P_{f_n}(t_n), & \text{if } a_n = \alpha_{M+2}, \end{cases} \tag{3.4}$$

where $P_{f_n}(t_n)$ represents the popularity of the requested file and $P_{s_n^m}(t_n)$ represents the popularity of the replaced file. If a cached file is replaced by the requested file, the reward reflects the change of the file popularity as $P_{f_n}(t_n) - P_{s_n^m}(t_n)$. It should be noticed that in this case, the reward can be negative if the requested file is less popular than the replaced file. In addition, if the action α_{M+1} is taken, the cache remains unchanged and hence the reward is set to zero since the requested file is

not locally served. Finally, if a requested file is locally served, which refers to the action α_{M+2}, the popularity of the requested file is used as the reward.

3.2.4 Action-Value Function

The action-value function is used to quantify how beneficial it is for the agent to perform a given action in a given state under a specific policy. Define $\pi_n(g) = a$ as the policy of choosing the action a under the state g at decision time step n. The action-value function of taking action a in state g at time t under a policy π_n can be defined as

$$Q_n^{\pi}(g,a) = \mathbb{E}_{\pi}\left[\sum_{i=n}^{N} \gamma^{i-n} r_i(g_i, \pi_n(g_i)) | g_n = g, a_n = a\right], \tag{3.5}$$

where γ is the discount factor showing the importance of the long-term reward compared to the current reward.

3.3 Context-Aware Popularity Learning

In the formulated MDP in the previous section, the reward function and transition probabilities depend on file popularities. Therefore, first, we need to learn and track popularities in order to solve the MDP. To learn the popularity of files, a simple and effective way is to calculate the frequency of the files that are being requested based on the request history. However, this method faces several problems. First, since the server is located at the network edge, the number of requests is limited in the early stage, thus calculating the frequency might be inaccurate especially when the file popularities are non-stationary and time-varying. In addition, the similarity between files are not considered while multiple files with similar context information may tend to yield close popularities. Therefore, in this section, we propose a context-aware popularity learning algorithm to track the time-varying content popularities.

3.3.1 Context Information Management

Context information represents the feature of the content and captures the situation under which the request is made. File request history can be treated as the

context information [2]. For example, the context information may include the request counts of a file in the last hour, the last day or the last week. With context information, each request req_n is associated with a multi-dimensional context vector v_n (depends on how many types of context information are available), which can be mapped as a request point in the context space $\mathcal{V}$, where each context information is treated as one axis.

In order to overcome the two deficiencies of popularity learning which were mentioned above, we apply clustering algorithm to the request points to group them into multiple sets. After the clustering process, the request points with the similar context information will be expected to be in the same set. Hence, instead of learning the popularity of each file, we learn the popularity of each set. By utilizing this context information, we will have much more points to accurately estimate the time-varying popularities, which will greatly enhance the accuracy and the learning rate of the popularity estimation.

However, the key challenge is the efficient and dynamic clustering of different request points with a similar context information. K-means clustering is a classic scheme that is able to partition N request points into K clusters. However, K-means is designed only for a fixed number of points, however, in our problem, the number of request points is increasing over time. To solve this problem, inspired by and advancing [139], we propose an incremental clustering-assisted learning algorithm which is suitable for learning the time-varying file popularities by processing the context information incrementally.

Figure 3.1 illustrates an example of clustering in a two-dimensional context space $\mathcal{V}$. The red points demonstrate various requests received in the course of time, which has been grouped into four clusters. By doing this, the request points with similar context information are grouped into the same cluster and treated to have the same popularity. Since the instant frequency of a requested file can be easily gathered, we consider the average instant frequencies of the request points in a cluster as the popularity of that cluster.

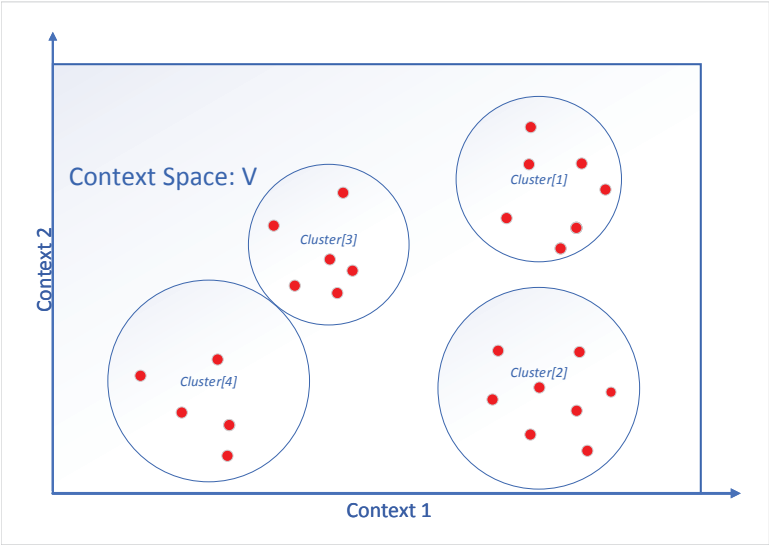

Figure 3.1: Context space

3.3.2 Incremental Clustering-assisted Popularity Learning

Define c_k as the centre of cluster $k \in \{1, ..., K_n\}$, where K_n represents the total number of clusters created till t_n. The clustering and learning process is divided into three steps as follows.

In the *first* step, once a new request arrives, if K_n is smaller than a certain threshold κ, the server will calculate the Euclidean distances between v_n (the context point of req_n) and each existing cluster centre. Let us define $D_n^{\min}$ as the distance between v_n and the closest cluster centre. If $D_n^{\min}$ is larger than the initial clustering distance threshold δ (which is a predefined parameter), this new context point will be selected as a new cluster centre, otherwise, it will be grouped into the closest cluster. The logic behind this step is to create a number of clusters which are not too close to each other for future clustering.

Once a request point is clustered into the existing cluster k, the average popularity $\bar{P}_k$ of cluster k is used as the estimated popularity of this requested file. $\bar{P}_k$ is updated as follows

$$\bar{P}_k = \Sigma_k / \Theta_k, \tag{3.6}$$

where Σ_k is the sum of the estimated popularity of the points in cluster k and Θ_k is

the number of points in cluster k. When a new point is added into cluster k, Σ_k and Θ_k are updated as

$$\begin{aligned} \Sigma_k &= \Sigma_k + q_n, \\ \Theta_k &= \Theta_k + 1, \end{aligned} \tag{3.7}$$

where q_n is the real popularity of f_n that can be calculated as the received number of requests for file f_n so far divided by the total number of requests.

In the *second* step, when K_n is equal to κ, the server calculates distances between any two centres. The set of these distances can be represented as $\mathcal{L}$. Define $d_x \in \mathcal{L}$ as the x^{th} smallest distance between any two centres. Then, the sum of the z smallest distances is defined as

$$\Delta_1 = \sum_{x=1}^{z} d_x, \tag{3.8}$$

where Δ_1 is the clustering distance threshold in phase 1, which will be used to decide whether a new cluster should be created or not once a new request point arrives.

As the number of clusters increases, the clustering distance threshold Δ_r should increase to control the growing rate of the cluster quantity. Specifically, the clustering distance threshold is updated in different phases. Let us define Δ_r as the clustering distance threshold in phase r. Each phase includes the creation of κ clusters. Define l_r as the number of clusters created in phase r. When l_r is larger than κ, it means a sufficient number of clusters has been created based on the current threshold Δ_r. Therefore, Δ_{r+1} will be scaled up to $\xi\Delta_r$ and l_r will be reset to zero. It should be ensured that $\xi > 1$ so that the clustering distance threshold is non-decreasing. By doing this, we can avoid the algorithm from creating too many clusters in a relatively small area in the context space or it may cause clusters overlapping with each other.

In the *third* step, when K_n becomes larger than κ, the proposed algorithm searches for the closest cluster centre for an arrived request point and calculates $D_n^{\min}$. Depending on how large $D_n^{\min}$ is compared to the clustering distance threshold

Δ_r, the algorithm would decide to map the newly arrived point to the closest cluster or to select it as a new cluster centre. This mechanism is implemented stochastically with the probability defined as

$$\rho_{n,r} = \min\left(D_n^{\min}/\Delta_r, 1\right). \tag{3.9}$$

More specifically, this new point will be selected as a new cluster centre with probability $\rho_{n,r}$, otherwise, it will join the closest cluster with probability $1-\rho_{n,r}$. Accordingly, if the distance between the new point and the closest cluster centre (i.e., $D_n^{\min}$) is much smaller than Δ_r, it is more likely that this point will be added to the closest existing cluster. On the other hand, if $D_n^{\min}$ is comparable with Δ_r, this point would create a new cluster with a higher probability. In particular, if $\rho_{n,r}$ is larger than 1, it means this new point is quite far from existing cluster centres and it will definitely be selected as a new cluster centre. The complete context-aware popularity learning algorithm is depicted in Algorithm 3.1.

3.4 Caching Update Algorithms

We modeled the caching decision process as an MDP in Section 3.2 and proposed a learning algorithm to track content popularities in Section 3.3. To solve the caching management MDP, since the transition probabilities are uncertain due to time-varying content popularities, in this section, one of the model-free temporal-difference (TD) learning schemes, namely the SARSA, is utilized to design an optimal caching algorithm [36] without having the dynamics of the environment (transition probabilities).

3.4.1 RL-based Caching

In SARSA [140], an agent interacts with the environment and updates the policy based on the taken actions. In particular, SARSA consists of taking an action on a state, noting the reward of the action and the next state, then choosing the next action on the following state, and updating the Q value. The update of the state-action value (i.e., the Q-value) only depends on the previous Q-value, the reward,

Algorithm 3.1 Context-Aware Popularity Learning

Require: $l_r \leftarrow 0$, $K_n \leftarrow 0$, $k \leftarrow 1$, $r \leftarrow 1$
  for $n \leftarrow 1$ **to** N **do**
    if $K_n < \kappa$ **then**
      Calculate $D_n^{\min}$
      if $D^{\min} > \delta$ **then**
        $c_k \leftarrow \nu_n$, $K_n \leftarrow K_{n-1} + 1$, $k \leftarrow k+1$
      else
        Update Σ_k , Θ_k and $\bar{P}_k$ based on (3.6) and (3.7)
      end if
    end if
    if $K_n = \kappa$ **then**
      Calculate Δ_1
    end if
    if $K_n \geq \kappa$ **then**
      Calculate $D_n^{\min}$
      Generate a Bernoulli binary variable $x_{n,r}$ with $\rho_{n,r} = \min(D_n^{\min}/\Delta_r, 1)$
      if $x_{n,r} = 1$ **then**
        $c_k \leftarrow \nu_n$, $K_n \leftarrow K_{n-1} + 1$, $l_r \leftarrow l_r + 1$, $k \leftarrow k+1$
      else
        Update Σ_k , Θ_k and $\bar{P}_k$ based on (3.6) and (3.7)
      end if
      if $l_r > \kappa$ **then**
        $\Delta_r \leftarrow \xi \Delta_{r-1}, l_r \leftarrow 0, r \leftarrow r+1$
      end if
    end if
  end for

and the Q-value of the next state-action; The dynamics of the environment is not needed. After updating the Q-value, the agent moves to the next state and executes the action which has been chosen earlier. The Q-value for a state-action is updated by

$$Q_{n+1}^{\pi}(g,a) = Q_n^{\pi}(g,a) + \omega[r_n(g,a) + \gamma Q_n^{\pi}(g', \pi_n(g')) - Q_n^{\pi}(g, \pi_n(g))], \quad (3.10)$$

where ω represents the learning rate which determines to what extent the Q-value is updated based on the newly acquired information and γ is the discount factor.

To decide which action to choose on the current state, the ε-greedy policy is introduced with $0 < \varepsilon < 1$. According to this policy, the agent either chooses the action which maximizes the Q-function at time step n given the current state with

probability $1-\varepsilon$ or randomly chooses an action with probability ε. The ε-greedy action selection method can be described as

$$\begin{aligned}&\Pr\left(\pi_n(g)=\underset{a\in\mathcal{A}}{\operatorname{argmax}}\,Q_n^{\pi}(g,a)\right)=1-\varepsilon,\\&\Pr\left(\pi_n(g)=\operatorname{randi}\left(\mathcal{A}\right)\right)=\varepsilon.\end{aligned}\tag{3.11}$$

In the TD algorithms such as SARSA, a table is maintained to save all the Q-values of different state-action pairs as the basis of the Q-value update and action selection. However, the size of the state space in our model is very large especially when a large file library and a large cache size are considered, thus directly applying the standard SARSA causes extremely huge memory cost [141]. In addition, this tabular method ignores the correlation between different Q-values,which makes it inefficient to learn the Q-value of each state-action individually, which shows poor generalization quality [142]. As a result, the standard SARSA is difficult and inefficient to be implemented. Therefore, a linear function approximation is utilized to reduce the storage cost and to accelerate the learning due to the fact that the algorithm can generalize its earlier experiences to previously unseen states. The Q-function is represented by a linear combination of a number of features which are able to appropriately reflect the inherent characteristics of the caching system. Therefore, the Q-function can be approximated as

$$Q_n^{\pi}(g,a)\approx\tilde{Q}_n^{\pi}(g,a,\theta_n)=\eta_n(g,a)^{\top}\theta_n,\tag{3.12}$$

where $\eta_n(g,a)^{\top}$ is the feature vector and θ_n is the parameter vector at n. $\eta(g,a)_n^{\top}$ is defined as

$$\begin{aligned}\eta_n(g,a)^{\top}=&[1,I_n^{\mathbf{req}}(1)P_1(t_n),\cdots,I_n^{\mathbf{req}}(F)P_F(t_n),I_n^{\mathbf{cache}}(1)P_1(t_n),\cdots,I_n^{\mathbf{cache}}(F)P_F(t_n),\\&I_n^{\mathbf{action}}(1),\cdots,I_n^{\mathbf{action}}(M+2)],\end{aligned}\tag{3.13}$$

where $I_{\mathbf{req}},I_{\mathbf{cache}}$, and $I_{\mathbf{action}}$ are binary indicators denoting which file is requested, which M files are cached, and which action is taken. Based on the definition, the

features of different states and actions are distinguishable, which helps to accurately update the value of the Q-function. To depict the contribution of each feature, the parameter vector $\theta_n \in \mathbb{R}^{2F+M+3}$ is introduced as the weight of each feature. To minimize the function approximation error [143], the gradient descent is applied to update the parameter vector as

$$\begin{aligned}\theta_{n+1} =& \theta_n + \omega[r_n(g,a) + \gamma\tilde{Q}_n^{\pi}(g',a',\theta_n) - \tilde{Q}_n^{\pi}(g,a,\theta_n)]\nabla_{\theta}\tilde{Q}_n^{\pi}(g,a,\theta_n),\\ =& \theta_n + \omega[r_n(g,a) + \gamma\tilde{Q}_n^{\pi}(g',a',\theta_n) - \tilde{Q}_n^{\pi}(g,a,\theta_n)]\eta_n(g,a),\end{aligned} \tag{3.14}$$

where the second equation holds since linear approximation is used.

In the RL-based caching, each time a request is received, Algorithm 3.1 is called to do the clustering and to estimate the file popularity. In the beginning, the cache of the server is assumed to be empty so it always caches the requested files. When the cache is full, the initial state g_0 is observed and the action a_0 is selected based on (3.11). It is noted that we add a hint to exclude α_{M+2} from the action selection process, thus the server always checks if a request can be locally served to make sure α_{M+2} is always taken if it is possible. After executing the selected action, the algorithm moves to a new state g_{n+1}, and an immediate reward is received. Then the parameter vector is updated by selecting the next action a_{n+1} and following (3.14). The detailed RL-based caching is described in Algorithm 3.2.

3.4.2 Reactive Caching

Due to the requirement of the real-time processing, the caching decision should be made with the least delay to improve CHR. For the proposed RL-based caching scheme, the computing load still cannot be ignored when a large size of the file library is considered. In addition, in the proposed scheme, there are three parameters that need to be adjusted to reach the optimal performance, which is time-consuming and can cause a generalization problem.

In the RL-based caching scheme, the objective is to maximize the Q-function which is a function of the discounted summation of the file popularities. Solving this problem ensures high CHR since the algorithm tends to cache those popular files.

Algorithm 3.2 RL-based caching

Require: γ, ω, ε, randomly initialize θ, observe g_0
  if the request can be locally served **then**
    $a_0 \leftarrow \alpha_{M+2}$
  else
    Select action a_o based on (3.11)
  end if
  $a_n \leftarrow a_0$, $g_n \leftarrow g_0$
  while $n \leq N$ **do**
    Play action a_n, and receive a request req_n
    Call Algorithm 3.1
    Observe r_n and the next state g_{n+1}
    if f_n is cached **then**
      $a_{n+1} \leftarrow \alpha_{M+2}$
    else
      Select the next action a_{n+1} based on (3.11)
    end if
    Update θ based on (3.14), $g_n \leftarrow g_{n+1}$, $a_n \leftarrow a_{n+1}$, $n \leftarrow n+1$
  end while

Enlightened by this setting, a reactive caching scheme is designed to overcome the above mentioned problems of the RL-based caching scheme. In particular, in this scheme, the server will definitely cache the more popular requested file if it cannot be locally served, and drop the least popular file without considering the long-term effect. It should be noted that the reactive caching also can explore the dynamic popularity profiles. The exploration is done by calling the Algorithm 3.1 which utilizes the context information to track the time-varying file popularity.

To ensure the reactive caching scheme works, the server maintains an extra record of the popularity of the cached files. Each time a request arrives, Algorithm 3.1 is called to group it into a cluster and estimate the popularity of the requested file, which is defined as $\tilde{P}_{f_n}(t_n)$. If the cache is not full, the server caches the requested file to improve the CHR. Otherwise, for each coming request, the server checks if it can be locally served or not. If this requested file can be locally served, the corresponding popularity of it can be updated by the latest estimated popularity $\tilde{P}_{f_n}(t_n)$. If the requested file cannot be locally served, the popularity of the cached least popular file f_{least} and the requested file f_n are compared, and the server de-

Algorithm 3.3 Reactive caching

for $n \leftarrow 1$ **to** N **do**
  Receive a request $\text{req}_n \leftarrow < f_n, t_n, \nu_n >$
  Call Algorithm 3.1, gets $\tilde{P}_{f_n}$
  if The cache is not full **then**
    $\mathcal{M}_n \leftarrow \mathcal{M}_n \cup f_n$
  else
    if $f_n \in \mathcal{M}_n$ **then**
      $P_{f_n}(t_n) \leftarrow \tilde{P}_{f_n}(t_n)$
    else
      Find f_{least}
      if $\tilde{P}_{f_n}(t_n) > P_{f_{\text{least}}}(t_n)$ **then**
        $\mathcal{M}_n \leftarrow \{\mathcal{M}_n \backslash f_{\text{least}}\} \cup f_n$
        $P_{f_n}(t_n) \leftarrow \tilde{P}_{f_n}(t_n)$
      end if
    end if
  end if
end for

termines to cache the more popular file. The detailed reactive caching scheme is described in Algorithm 3.3.

3.5 Performance Analysis

In this section, we first bound the learning regret of the file popularity and then utilize it to derive the bound of the CHR of the proposed reactive caching scheme.

3.5.1 Learning Regret of File Popularity

There is a widely applied assumption [144] that the expected popularity of files with similar context information is similar. This assumption can be mathematically formulated as follows.

Assumption 3.1. *(Uniform Lipschitz continuity) There exists a real number $\beta > 0$ such that for any two requests, the popularity difference between the requested files can be bounded as*

$$\mathbb{E}\Big[|q_n - q_{n'}|\Big] \leq \beta ||\nu_n - \nu_{n'}||, \tag{3.15}$$

where $||\cdot||$ represents the Euclidean norm.

Based on this assumption, we can bound popularity difference of files by the Euclidean norm of the corresponding context points in the context space $\mathcal{V}$. As a result, to bound the learning regret of file popularity, the worst case of the regret should be defined as the largest Euclidean norm between the new point and the farthest point in the same cluster. Before we estimate the total forecast popularity error brought by the first N requests, we prove the following lemma.

Lemma 3.1. *Let us assume a sequence of N independent experiments denoted by $x_1, x_2, \cdots, x_N$, where each experiment succeeds with a probability of $p_n = \min\{\frac{A_n}{B}, 1\}$ where $B \geq 0$ and $A_n \geq 0$ for $n = 1, 2, \cdots N$. If u denotes the random number of consecutive unsuccessful experiments, then with $\mu < 1$, we have*

$$\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i - \frac{{A_i}^2}{B}\right)\Big] \leq B^{\mu}, \tag{3.16}$$

Proof: Let u' be the maximal index for which $p_i < 1$ for all $i \leq u'$. Therefore, we have

$$\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i - \frac{{A_i}^2}{B}\right)\Big] = \sum_{i=1}^{u'}\Big[(A_i - \frac{{A_i}^2}{B})\sum_{i'=i}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\Big]. \tag{3.17}$$

By utilizing mathematical induction, we can rewrite (3.17) (see Appendix A for the proof) as

$$\begin{aligned}\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i - \frac{{A_i}^2}{B}\right)\Big] &\leq \sum_{i=1}^{u'}(A_i - \frac{{A_i}^2}{B})\prod_{j=1}^{i}(1-\frac{A_j}{B}) \\ &\leq \sum_{i=1}^{u'}(A_i - \frac{{A_i}^2}{B})\prod_{j=1}^{i-1}(1-\frac{A_j}{B}).\end{aligned} \tag{3.18}$$

Let us divide the right-hand side of (3.18) into two parts and write as

$$\begin{aligned}&\sum_{i=1}^{u'}\left(A_i\prod_{j=1}^{i-1}(1-\frac{A_j}{B}) - \frac{{A_i}^2}{B}\prod_{j=1}^{i-1}(1-\frac{A_j}{B})\right) \\ =&B\sum_{i=1}^{u'}\frac{A_i}{B}\prod_{j=1}^{i-1}(1-\frac{A_j}{B}) - \sum_{i=1}^{u'}\frac{{A_i}^2}{B}\prod_{j=1}^{i-1}(1-\frac{A_j}{B}).\end{aligned} \tag{3.19}$$

Define $q_i = A_i/B < 1$ for $i \leq u'$ as the probabilities of different events to be successful, then (3.19) can be represented as

$$B\sum_{i=1}^{u'} q_i \prod_{j=1}^{i-1}(1-q_j) - \sum_{i=1}^{u'} A_i q_i \prod_{j=1}^{i-1}(1-q_j). \tag{3.20}$$

It is obvious that $q_i \prod_{j=1}^{i-1}(1-q_j)$ is the probability that i is the first successful event. Since these events (when i takes different values) are mutually exclusive, $\sum_{i=1}^{u'} q_i \prod_{j=1}^{i-1}(1-q_j) \leq 1$. Therefore, (3.20) can be upper bounded as

$$\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i - \frac{{A_i}^2}{B}\right)\Big] \leq B - \sum_{i=1}^{u'} A_i p_i \prod_{j=1}^{i-1}(1-p_j). \tag{3.21}$$

In (3.21), $\sum_{i=1}^{u'} A_i p_i \prod_{j=1}^{i-1}(1-p_j)$ can be treated as the expected value of A_i corresponding to the first successful experiment x_i (i.e., $\mathbb{E}[A_i|x_i$ is the first successful experiment]). Obviously, this value is larger than the expected value of A_i no matter x_i succeeds or not. Intuitively, this can be explained as the first succeeding experiment has a larger A_i *on average* than the average value of A_i. Therefore, the upper bound can be written as

$$\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i - {A_i}^2/B\right)\Big] \leq B - \mathbb{E}[A_i] = B^{\mu}. \tag{3.22}$$

Therefore, it can be concluded that there exists an upper bound which is smaller than B (since $\mathbb{E}[A_i] > 0$) and this can be further represented as B^{μ} with $\mu < 1$.

Lemma 3.2. *The number of clusters created while serving the first N requests can be upper bounded by* $\mathcal{O}(\kappa \log_{\xi}(\upsilon N))$ *where* $\upsilon \geq 1$ *is defined as the dataset aspect ratio of the maximum distance to the minimum distance between any two points in the context space.*

Proof: Based on Theorem 3 in [139], consider the phase r' of our algorithm where, for the first time, $\Delta_{r'} \geq \frac{W}{\kappa \log n}$ where n represents the n^{th} point and W is defined as the summation of all distances between each point and its cluster centre in the optimal solution. The total number of clusters created in our algorithm before r' can be upper bounded by $\mathcal{O}(\kappa \log_{\xi}(\upsilon n))$. In addition, during and after phase r'

can be upper bounded by $\mathcal{O}(\log_{\xi} n)$. Combining both bounds we can conclude that the number of clusters after N requests is the order of $\mathcal{O}(\kappa \log_{\xi}(\upsilon N))$.

Proposition 3.1. *The expected total popularity learning error of the first N requests can be upper bounded by* $\mathcal{O}(N^{\mu})$ *for some* $\mu < 1$.

Proof: In the first step of the proposed scheme, when a point is chosen as a cluster centre (which means it is the first point in that cluster), it will not cause any error. This is because the caching decision is made only based on the instant request frequency of that file. In the third step, for each coming point, there is a probability $\rho_{n,r}$ for it to be selected as a new cluster centre and a probability $1-\rho_{n,r}$ for it to be grouped into an existing cluster. In the first case of the third step, obviously the forecast popularity error is zero. In the second case, the largest distance (which is the worst-case error) can be defined as $\lambda_n D_n^{\min}$ where λ_n is a factor to scale $D_n^{\min}$ to the worst case (largest distance). Therefore, the expectation of forecast popularity error of one point can be expressed as

$$\mathbb{E}[|\tilde{q}_n - q_n|] \leq \lambda_n D_n^{\min}(1-\rho_{n,r}) \leq \lambda^{sup} D_n^{\min}\left(1 - \frac{D_n^{\min}}{\Delta_r}\right), \tag{3.23}$$

where $\tilde{q}_n$ represents the learned popularity of f_n, q_n represents the file popularity in the optimal scheme, and λ^{sup} is the upper bound of λ_n.

In the phase r, κ clusters are created and this can be interpreted as having κ sequences of unsuccessful experiments if we define a success as creation of a new cluster. For each of these sequences, considering that the upper bound of error for each experiment is given by (3.23) and success probability is $\rho_{n,r} = \min\left(D_n^{\min}/\Delta_r, 1\right)$, the expected error can be upper bounded by $\lambda^{sup}(\Delta_r)^{\mu}$ according to Lemma 3.1. Consequently, the expected value of the sum of forecast popularity errors in phase r can be upper bounded by $\kappa\lambda^{sup}(\Delta_r)^{\mu}$. Mathematically,

$$\mathbb{E}\left[\sum_{n=1}^{n_r} |\tilde{q}_n - q_n|\right] \leq \kappa\lambda^{sup}(\Delta_r)^{\mu}, \tag{3.24}$$

where n_r is the number of requests received in phase r. Therefore, the expected total

forecast popularity error of the first N requests is at most

$$\mathbb{E}\Big[\sum_{n=1}^{N}|\tilde{q}_n - q_n|\Big] \leq \sum_{r=1}^{R}\kappa\lambda^{sup}(\Delta_r)^{\mu}, \tag{3.25}$$

where R is the total number of phases corresponding to N requests. Considering that $\Delta_r = \xi^{r-1}\Delta_1$, (3.25) can be further represented as

$$\mathbb{E}\Big[\sum_{n=1}^{N}|\tilde{q}_n - q_n|\Big] \leq \sum_{r=1}^{R}\kappa\lambda^{sup}(\xi^{r-1}\Delta_1)^{\mu}. \tag{3.26}$$

Since (3.26) can be treated as the summation of a geometric progression with a common ratio of ξ^{μ}, the expected total forecast popularity error can be upper bounded by

$$\mathbb{E}\Big[\sum_{n=1}^{N}|\tilde{q}_n - q_n|\Big] \leq \frac{\kappa\lambda^{sup}\Delta_1^{\mu}\left((\xi^{\mu})^{R}-1\right)}{\xi^{\mu}-1}. \tag{3.27}$$

Based on Lemma 3.2, the number of clusters created while serving the first N requests can be upper bounded by $\mathcal{O}(\kappa\log_{\xi}(\upsilon N))$. Therefore, the total number of phases R corresponding to N requests can be upper bounded by

$$R = \mathcal{O}\left(\kappa\log_{\xi}(\upsilon N)\right)/\kappa = \mathcal{O}\left(\log_{\xi}(\upsilon N)\right), \tag{3.28}$$

Since the algorithm will step into a new phase when κ cluster centres are created. Consequently, based on (3.28), (3.27) can be written as

$$\begin{aligned}\mathbb{E}\Big[\sum_{n=1}^{N}|\tilde{q}_n - q_n|\Big] &\leq \frac{\kappa\lambda^{sup}\Delta_1^{\mu}\left((\xi^{\mu})^{\log_{\xi}(\upsilon N)}-1\right)}{\xi^{\mu}-1}\\ &= \mathcal{O}\left((\xi^{\mu})^{\log_{\xi}(\upsilon N)}\right)\\ &= \mathcal{O}(N^{\mu})\end{aligned} \tag{3.29}$$

3.5.2 Learning Regret of Cache Hit Rate

Similar to [1], let us divide time into periods with each of them containing ϕ requests. Based on the proposed caching scheme, the server will always cache the M-most popular files. In addition, define Q^{sort} as the sorted vector of the popularities of all files in a time period s. For each requested file, assume that the popularity error of file f satisfies

$$|Q_{\mathrm{r}}^{sort}(f) - Q_{\mathrm{e}}^{sort}(f)| \leq \Delta Q,\ f \in \mathcal{F} \tag{3.30}$$

where $Q_{\mathrm{e}}^{sort}(f)$ represents the estimated popularities of file f learned by the proposed algorithm and $Q_{\mathrm{r}}^{sort}(f)$ represents its corresponding real popularity. Based on Proposition 2 from [1], the CHR of the proposed algorithm in time period s can be lower bounded as

$$\tilde{H}_s \geq Q(M) - \frac{2M}{\phi} - \frac{M \cdot \Delta Q_s}{\sum_{f\in\mathcal{F}} Q_{\mathrm{r}}^{sort}(f)}, \tag{3.31}$$

where $Q(M) = \frac{\sum_{f=1}^{M} Q_{\mathrm{r}}^{sort}(f)}{\sum_{f=1}^{F} Q_{\mathrm{r}}^{sort}(f)}$ represents the normalized total popularity of the M most popular files. This lower bound can be divided into two parts. The first part $(Q(M) - 2M/\phi)$ depends on the caching capacity M and the file popularity distribution (related to $Q(M)$). On the other hand, $\tilde{H}_s$ also depends on the total learning error of file popularities in s (which is denoted as $\Delta Q_s = \sum_{n=1}^{\phi} \Delta Q_n$), since a larger error will lead to a lower CHR. For the optimal scheme, since the full knowledge of file popularity is known, ΔQ_s is set to 0, and the CHR in s is therefore defined as $H_s \geq Q(M) - 2M/\phi$.

Theorem 3.1. *The CHR of the reactive caching converges to the optimal CHR, namely* $\mathbb{E}[\tilde{H}] = \mathbb{E}[H]$.

Proof: First, we consider the CHR difference between the optimal scheme and the proposed scheme in time period s. The difference can be represented as

$$H_s - \tilde{H}_s \leq \frac{M \cdot \Delta Q_s}{\sum_{f\in\mathcal{F}} Q_{\mathrm{r}}^{sort}(f)} \leq \frac{M \cdot \Delta Q_s}{Q^{inf}}, \tag{3.32}$$

where Q^{inf} is defined as the lower bound of $\sum_{f\in F} Q_r^{\mathrm{sort}}(f)$.

After that, we wish to calculate the expected difference over all time periods. Therefore, the expected CHR difference can be represented as

$$\begin{aligned}\mathbb{E}[H-\tilde{H}] &\leq \lim_{N\to\infty} \frac{\sum_{s=1}^{\frac{N}{\phi}}(H_s-\tilde{H}_s)}{\frac{N}{\phi}},\\ &= \lim_{N\to\infty} \frac{\sum_{s=1}^{\frac{N}{\phi}}\frac{M\cdot\Delta Q_s}{Q^{inf}}}{\frac{N}{\phi}},\\ &= \frac{M}{Q^{inf}}\lim_{N\to\infty}\frac{\phi}{N}\sum_{s=1}^{\frac{N}{\phi}}\Delta Q_s, \end{aligned} \tag{3.33}$$

where H and $\tilde{H}$ are the final CHRs of the optimal and the proposed caching scheme after receiving N requests. To derive a bound for the mean CHR difference from the first time period to the last, the forecast popularity error can be summed from the first request to the last. Therefore, by considering Proposition 3.1, the expected difference can be represented as

$$\begin{aligned}\mathbb{E}[H-\tilde{H}] &\leq \frac{M}{Q^{inf}}\lim_{N\to\infty}\frac{\phi}{N}\sum_{n=1}^{N}|\tilde{q}_n-q_n|,\\ &\leq \frac{M\phi}{Q^{inf}}\lim_{N\to\infty}\frac{\mathcal{O}(N^{\mu})}{N}=0. \end{aligned} \tag{3.34}$$

According to (3.34), the expected CHR difference between the proposed algorithm and the optimal algorithm is zero, and our algorithm converges to the optimal scheme. Besides, it should be noted that the conclusion holds for any number of file library size.

3.5.3 Time Complexity

For the proposed RL-based caching scheme, each time a request arrives, Algorithm 3.1 is called to estimate the popularity of the requested file, followed by the SARSA algorithm to make a decision with the knowledge of the learned file popularity. In Algorithm 3.1, we need to find the nearest cluster centre of the newly requested file. According to Lemma 3.2, the number of clusters created while serving n requests can be upper bounded by $\mathcal{O}(\kappa\log_{\xi}(\upsilon n))$. Therefore, the time complexity of finding the nearest cluster centre can be represented as $\mathcal{O}(\log_{\xi}(\upsilon n))$. In

SARSA, before deciding a action to take, matrix multiplication is needed to calculate the Q-value of each action. Recall that M denotes the cache size and F denotes the file library size. Since the size of feature vector is $(M+2, 2F+M+3)$ and the size of the parameter vector is $(2F+M+3, 1)$, the time complexity of calculating the multiplication of the matrices is $\mathcal{O}(MF)$. Therefore, the time complexity of the proposed RL-based caching is $\mathcal{O}(\log_{\xi}(\upsilon n)+MF)$.

For the proposed reactive caching scheme, it does not use matrix multiplication but only calls Algorithm 3.1 and sorts the cached files in a decreasing order of popularities. Therefore, the time complexity can be represented as $\mathcal{O}(\log_{\xi}(\upsilon n)+M)$. Comparing both algorithms, we can find that both algorithms reach low time complexity. However, in practice, there is always a limited cache size but a potentially large file library. In this case, RL-based caching leads to higher time complexity, which highlights the significance of designing the reactive caching scheme.

3.6 Simulation Results

In this section, the performance of the proposed caching schemes is evaluated and compared with other existing algorithms. The effect of parameters used in the context-aware popularity learning scheme is also investigated.

3.6.1 Simulation Setup

The file popularities are modelled using a Zipf distribution which is widely used in content caching literature [52, 53, 86]. In addition, the request arrival is assumed to follow a Poisson process which implies the time interval between the two consecutive requests follows an exponential distribution with a rate parameter ζ which increases for more frequent requests.

The simulation lasts for 72000 time slots. The duration of each time slot is one minute and $F = 100$. To evaluate the robustness of the proposed algorithms against time-varying popularity profiles, a deterministic variation is introduced to the file popularity distribution by randomly permuting all the files at every 12000 time slots. The learning rate, the discount factor and the ε-greedy factor are set to $0.1, 0.05$ and 0.1, respectively. All the results are generated by running the simulation for five

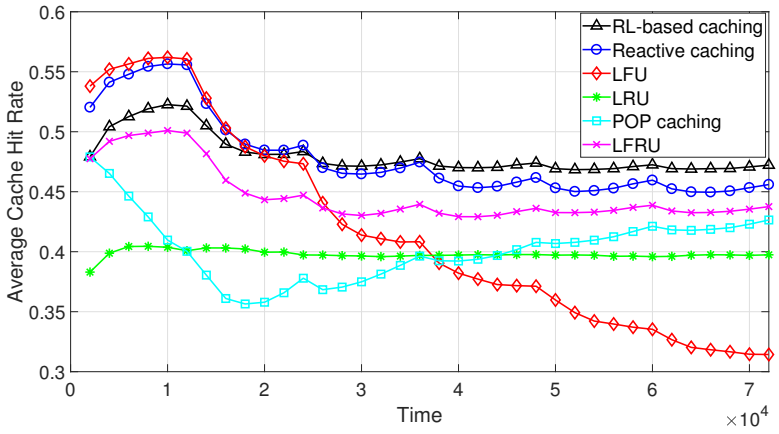

Figure 3.2: CHR over time

times and calculating the average CHR. In addition, the request history is chosen to be the context information in the simulation. Specifically, we use the total number of requests for a file in the past week and the past day as the context information in the simulation.

As a *benchmark*, we simulate LRU, LFU, LFRU, and a popularity-driven caching (POP) [1] algorithm. For the LRU scheme, the node always tries to update its cache by replacing the least recently requested file with the newly requested file. Besides, LRU is suitable for the case that the popularity of file changes over time because it does not consider the request history when making a decision. In LFU scheme, the node always cache the most frequently requested file. However, the caching performance might drop drastically if the file popularity changes. In addition, LFRU combines benefits of both LRU and LFU by partitioning the cache into two parts. It is a well known scheme for content caching network. Finally, POP caching is also simulated which is a context-aware caching algorithm with the ability to handle the time-varying file popularity.

3.6.2 Numerical Results

Figure 3.2 illustrates the CHR of different caching algorithms over time. The caching capacity is set to 10 files and ψ is set to 1. The proposed algorithms can identify the changes of the popularity profiles. In addition, the RL-based caching scheme reaches the highest CHR and competitive robustness among all six schemes because it introduces a reasonable amount of exploration to deal with the variability

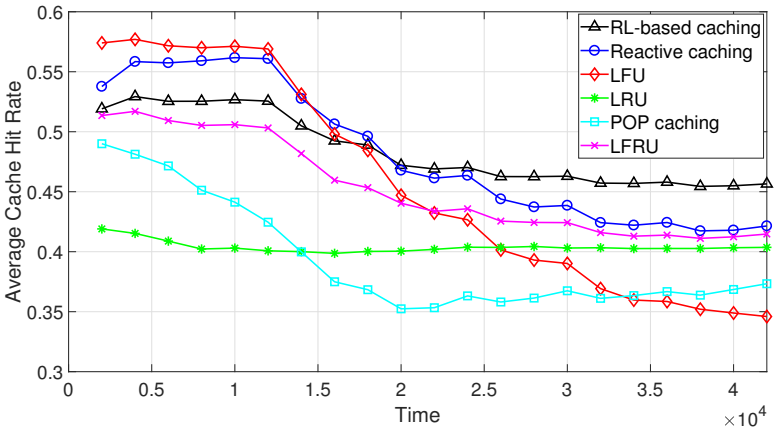

Figure 3.3: CHR over time

of the popularity profiles thus caching the more popular files when the popularity profiles change. Moreover, the reactive caching reaches a competitive CHR and robustness as well. Note that before the variation happens, this scheme performs well because the proposed online learning algorithm is able to quickly learn the popularity profiles and the incremental clustering is more efficient than the grid-based clustering scheme utilized in the POP caching scheme. Finally, for the rest benchmark schemes, it is obvious that LFU can perform well before any variations but its robustness is very poor. LRU is very robust but cannot reach a good CHR when the storage resource is limited. As a combination of LRU and LFU, LFRU performs much better, however, there is an obvious gap because LFRU does not consider the context information. Besides, this scheme is quite heuristic which means it needs tuning to reach a better performance.

Figure 3.3 shows the caching performance over time when the period of the popularity variation decreases from 12000 to 6000 time slots. Therefore, the variation happens more frequently and the RL-based caching scheme can reach the highest CHR. In addition, due to the more frequent variations, the gap between the RL-based caching and the reactive caching becomes larger, which further demonstrates the robustness of the proposed RL-based caching scheme. Moreover, the reactive caching scheme still reaches a higher CHR as compared to the benchmark schemes.

Figure 3.4 depicts the caching performance versus the capacity which ranges

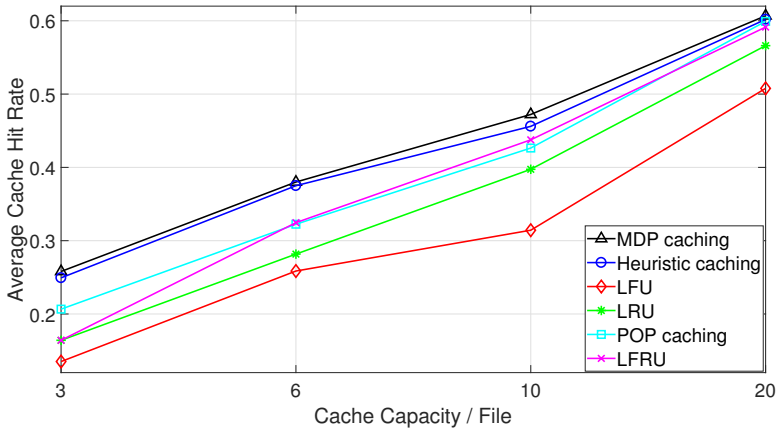

Figure 3.4: CHR versus caching capacity

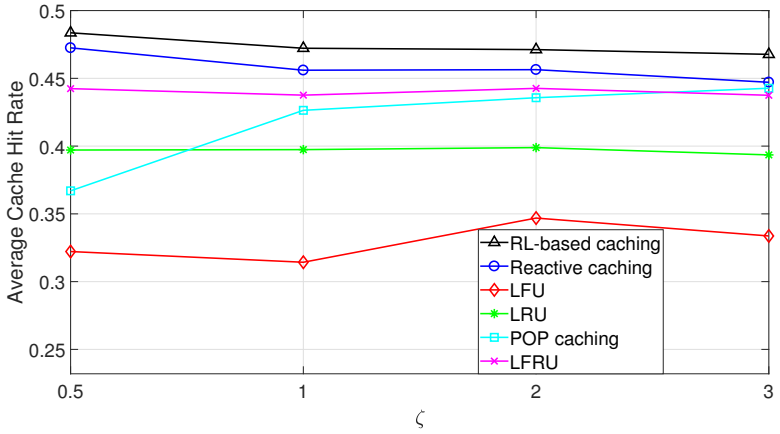

Figure 3.5: CHR versus ζ

from 3 to 20 (files). The CHRs are collected at the end of the simulation with $\psi = 1$. Obviously, the proposed RL-based caching scheme outperforms other benchmark schemes for different cache capacities. Furthermore, the reactive caching scheme reaches a competitive CHR as well which proves that the proposed schemes are suitable and robust for the cache-limited scenario.

In Figure 3.5, the effect of the request arrival rate is studied by changing the rate parameter ζ of the exponential distribution. The caching capacity here is set to 10 files and ψ is set to 1. The performance of the proposed caching schemes is robust against ζ because according to the figure, the choice of the length of context information is not sensitive to the request arrival rate. To further study the effect of the file library size on the CHR, we simulate a new scenario with 1000 files in the file library and the cache size is set to 20. As we observe in Figure 3.6, the

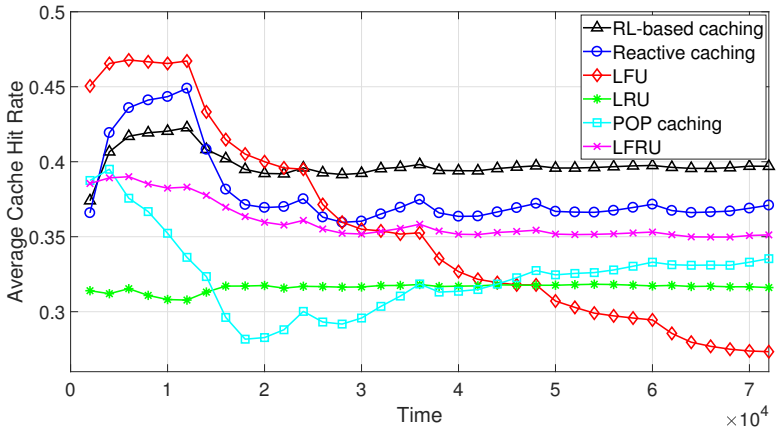

Figure 3.6: CHR over time

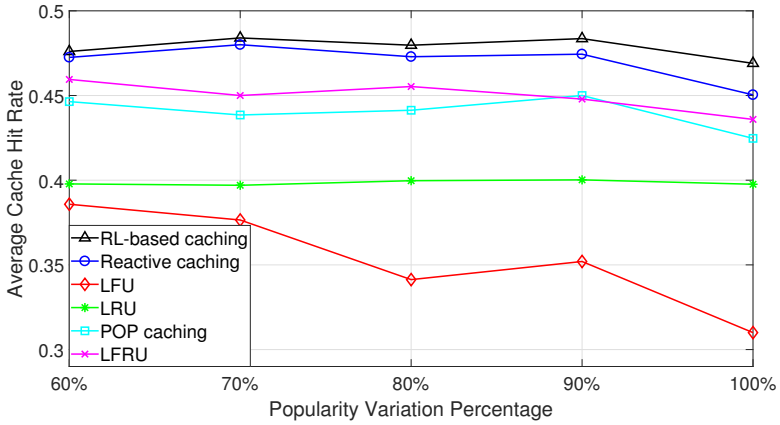

Figure 3.7: CHR versus popularity variation percentage

proposed RL-based caching algorithm reaches the highest CHR and it is very robust against the frequent changes of the file popularity profiles. In addition, the reactive caching scheme also reaches a competitive performance as compared to other benchmark schemes. Finally, the running times of both the proposed caching schemes are measured under this setting. Both the designed algorithms were run on an i5 desktop computer by MATLAB. In order to accurately measure the running time, both schemes are executed for 1000 time slots. The running times for RL-based caching and reactive caching are 1159 seconds and 642 seconds, respectively. According to the measurements, reactive caching runs faster than RL-based caching, which demonstrates that the former scheme requires a lower computational complexity.

In the previous results, the change of the popularity profile is assumed to be

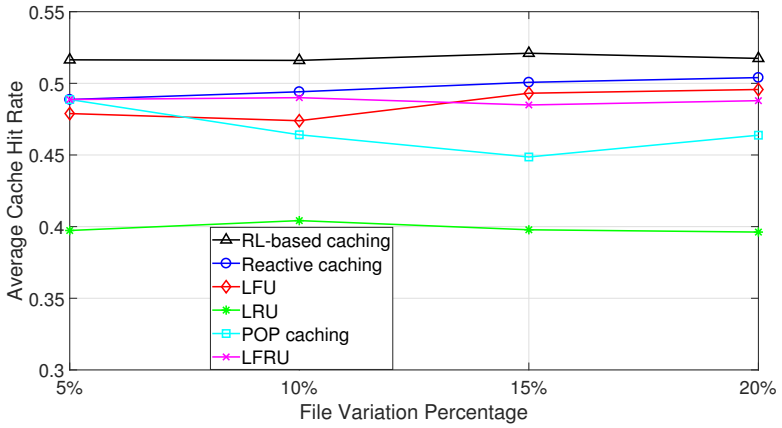

Figure 3.8: CHR versus file variation percentage

independent over time and the popularities of all the files are changed at each variation. Inspired by [86], we further test the caching performance of the proposed algorithms with correlated file popularity, which means only a part of file popularities are changed, so the popularity profile is dependent over time and correlated with previous profile. In Figure 3.7, the variation percentage of popularity profiles ranges from 60% to 100%. According to the results, with the variation percentage increasing, the CHRs of the proposed schemes drop because the popularity profiles are more dynamic. However, the RL-based caching scheme still reaches the best performance and the performance gap between it and the reactive caching becomes larger when the popularities of more files vary. In order to further evaluate the caching performance, a new scenario is simulated. For each variation of popularity profiles, a subset of the files are removed from the file library and a number of new files are included. Besides, a part of the file popularities are still changed periodically. Figure 3.8 presents the caching performance for this dynamic scenario. The RL-based scheme reaches the highest CHR, which demonstrates the robustness of it under a correlated time-varying popularity profiles.

3.6.3 Correlated File Request Process

In this section, we designed two new settings to extend the simulation results, which capture the temporal correlation of file requests in varying ways. The first one is based on the Bernoulli model [86] and the second is based on the SNM [91]. To implement the time-varying popularity profile, a part of the popularity profile is

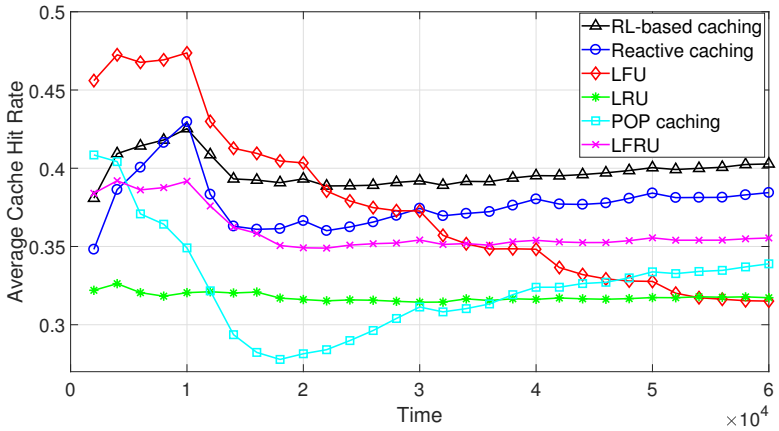

Figure 3.9: CHR over time-Bernoulli request model

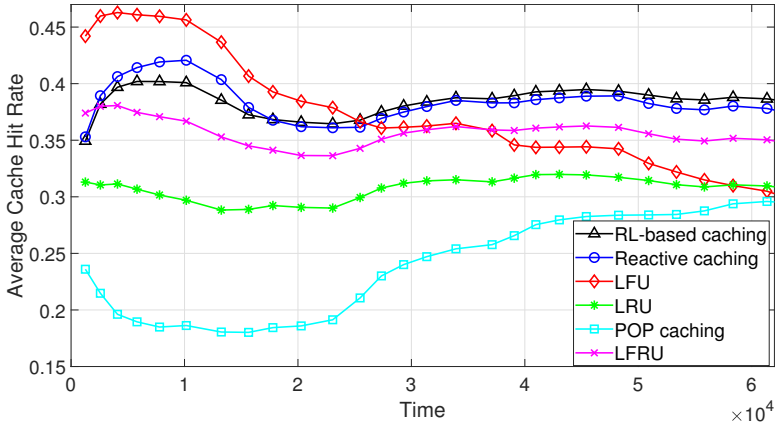

Figure 3.10: CHR over time-SNM

changed periodically consistent with previous results. In Figure 3.9, we study the Bernoulli request model which can capture the request correlation. Assume there are 10 users and each user can make a file request with the probability of 0.1 in each time slot and the file library size is 1000. The result reflects that the proposed algorithms perform well in this new setting and the correlated requests will not affect the caching performance as compared to the results of independent request model (Figure 3.6).

In addition, we also simulate the SNM which is able to capture the popularity evolution and explicitly account for the temporal locality of file requests. Following the setting in [145], the requests for the contents in a window of time slots are generated based on the SNM using the exponential shape [91]. The average number of requests for each content in the considered time window follows the Zipf distribu-

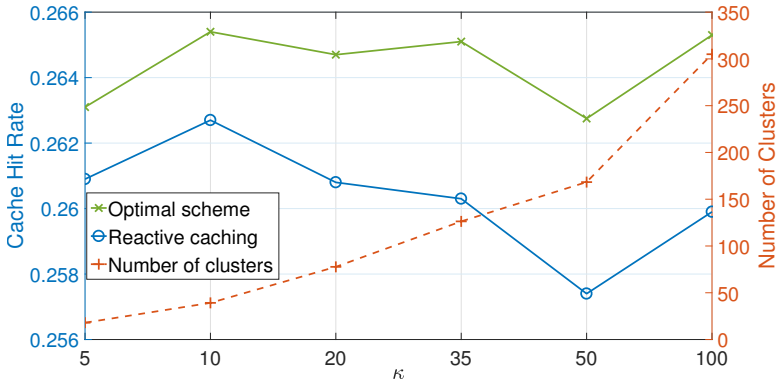

Figure 3.11: CHR versus δ

tion. The result is presented in Figure 3.10. According to the results, we conclude that both the proposed caching schemes outperform all the benchmarks.

3.6.4 Parameter Determination

Here, we investigate the effect of parameters of the proposed context-aware popularity learning algorithm. The popularity profiles are assumed to be unchanged. The optimal caching scheme which is assumed to know the true file popularity is also simulated. For the optimal caching, the server caches the most popular files without any updates.

Figure 3.11 depicts the CHR and the number of clusters versus δ. When δ is relatively small, the CHR gap between the optimal scheme and the reactive caching scheme is very small. However, when $\delta \geq 100$, the gap starts to increase which means the popularity learning scheme gradually loses its learning accuracy. The reason is that when δ is too large, only very few clusters will be created so that the request points with different popularity levels can be grouped into the same cluster. Therefore, there is a high probability that the algorithm cannot identify the difference between the two requests. Moreover, even when $\delta = 100$ and only 16 clusters are created, the reactive caching still performs comparably well, which means our algorithm is not very sensitive to this parameter. In the simulation, to avoid creating too few clusters, δ is set to 5.

Figure 3.12 demonstrates the caching performance versus κ. It is observed that as κ increases, the CHR difference between the optimal scheme and the reactive

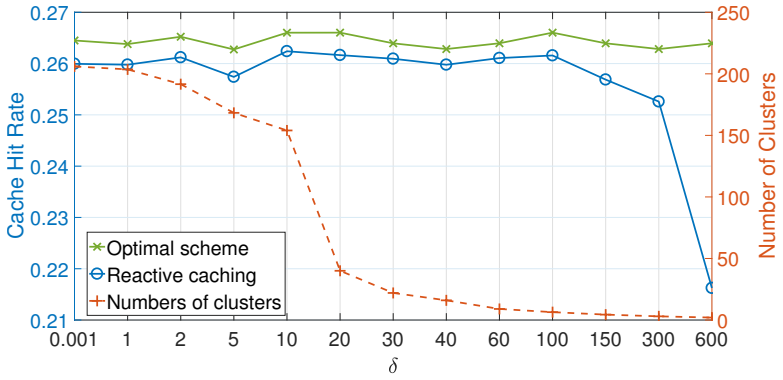

Figure 3.12: CHR versus κ

scheme first increases and then remains stable. This means that creating too many clusters cannot necessarily help to improve the learning accuracy but may lead to confusion of popularity learning. However, we should notice that even the largest performance difference is only about 0.5%, which means the proposed algorithm is reliable with the change of κ. In our simulations, to avoid the algorithm creating too few clusters and thus reducing the CHR, κ is set to 20. We also investigate the effect of ξ and z. The results show that these parameters do not affect the caching performance severely but only affect the speed of creating clusters. Therefore, ξ and z are chosen as 10 and 15 respectively.

3.7 Conclusion

In this chapter, we study a caching decision problem in the network edge server under a dynamic and time-varying file popularity profile. We first design a context-aware popularity learning algorithm to improve the accuracy and speed of tracking the file popularity. With the assistance of this algorithm, an RL-based caching scheme is proposed to make proper caching decisions and to improve the CHR. Furthermore, a reactive caching algorithm is designed to reduce the computational complexity for real time processing. Via the theoretical analysis, we demonstrate the superiority and the efficacy of the proposed algorithms. Finally, through numerical results, we demonstrate that the proposed algorithms are able to achieve a competitive and robust caching performance as compared to various benchmark schemes.

Chapter 4

Edge-Assisted Crowdsourced Live Streaming: Joint Transcoding Task Assignment and Association Control

4.1 Introduction

The rapid development of content delivery networks and cloud computing has facilitated CLSPs that enable people to broadcast live videos which can be watched online by a growing number of viewers. However, to ensure reliable viewer experience, it is important that the viewers should be provided with multiple standard video versions. To achieve this goal, edge-assisted transcoding has become a popular solution to meet the growing demand of computational resources and delay-sensitive services. In [24], the viewers' potential transcoding capabilities and the willingness to help the broadcaster have been demonstrated, and the authors concluded that utilizing the computational resources of the viewer devices can greatly relieve the burden of the CLSP and provide better transcoding service.

However, it is a challenge to dynamically assign transcoding tasks to stable and efficient fog devices since they are not dedicated for transcoding and they could even be offline during transcoding [25]. Therefore, in this chapter[1] we design a edge-assisted transcoding and viewer association algorithm which aims to assign

[1]Part of Chapter 4 has been published in the IEEE Communications Letters [146].

the transcoding tasks to suitable fog devices and optimize the tradeoff between the QoE of viewers and the CLSP's cost for transcoding. More specifically, we first introduce the transcoding success rate of the fog device and define a new QoE metric for the edge-assisted transcoding system. Then, a joint transcoding task assignment and viewer association optimization problem is formulated. The resultant non-convex integer programming is solved by using CGP. Finally, based on trace-driven simulations and by comparing the proposed algorithm with existing benchmark schemes, it is shown that the proposed algorithm outperforms the current cloud transcoding system and can dynamically decide the transcoding schedule.

The rest of this chapter is organized as follows. Section 4.2 describes the system model. The non-convex optimization which aims to maximize the network utility is formulated in Section 4.3 and the proposed joint edge transcoding and viewer association scheme is presented as well. The simulation setup and results are presented in Section 4.4, and the conclusions are drawn in Section 4.5.

4.2 System Model

Assume there is a set of broadcasters, i.e., $\mathcal{I} = \{1, 2, \cdots, I\}$ in the network as illustrated in Figure 4.1. We consider a time-slotted system. The bit rate of the original live video of broadcaster $i \in \mathcal{I}$ in time slot t is defined as B_i^t. Moreover, consider there are J^t viewers in time slot t and $\mathbf{V}_{i,j}^t$ is a binary parameter indicating whether viewer $j \in \mathcal{J}^t = \{1, 2, \cdots, J^t\}$ chooses to watch broadcaster i's live stream in time slot t or not. Denote $y \in \mathcal{Y} = \{1, 2, \cdots, Y\}$ as a video representation which is one of Y possible standard quality levels of a transcoded video. The bit rate of representation y is defined as b_y.

When an original live video is uploaded by a broadcaster i to the regional data centers, the CLSP scheduler will decide which representation should be transcoded by which fog device based on the viewer requirements. Then, all the original live videos will be transmitted to the selected fog devices through those regional data centers for transcoding. After the transcoding process, all the standard transcoded live videos will be transmitted from the fog devices to the associated viewers. Let us

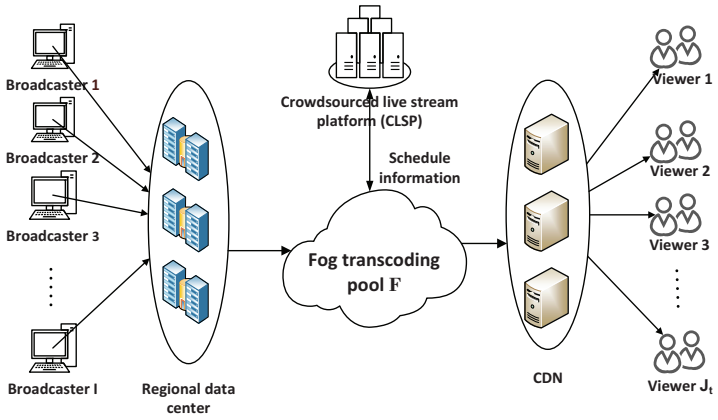

Figure 4.1: System model

assume $\mathcal{F}$ represents the set of all available fog devices. It is assumed that each fog device $f \in \mathcal{F} = \{1, 2, \cdots, F\}$ is able to transcode one broadcaster's live video into only one standard video version because of the limited computational capability of fog devices.

To describe the transcoding task assignment and viewer association problem, two binary variables are defined as $\mathbf{I}^t_{i,y,f}$ and $\mathbf{W}^t_{j,f}$. $\mathbf{I}^t_{i,y,f}$ takes 1 when the fog device f is selected to transcode the original video from broadcaster i into representation y in time slot t and 0 otherwise. In addition, $\mathbf{W}^t_{j,f}$ takes 1 if viewer j is associated with fog device f in time slot t to watch the transcoded live video and 0 otherwise.

4.2.1 Cost Model

To incentivize a fog device to participate in transcoding, CLSP will pay a reward to a fog device if the assigned transcoding task is successfully accomplished. The cost for the task of transcoding broadcaster i video into representation y by fog node f is defined as

$$c^t_{i,y,f} = \mathbf{I}^t_{i,y,f} \cdot \Phi(y) \cdot \alpha_f, \tag{4.1}$$

where $\phi(y)$ is a non-decreasing concave function of y to model the fact that transcoding the same live video to a higher version of representation consumes more computing resources thus causes higher costs. The reward is paid for the

transcoding task only once regardless of how many viewers are watching the same video representation from a broadcaster. Moreover, α_f is defined as the transcoding success rate of device f which reflects the online transcoding stability of f. A higher value of α_f means the probability of being offline during the transcoding is smaller. The total cost related to broadcaster i is defined as

$$c_i^t = \sum_{y \in \mathcal{Y}} \sum_{f \in \mathcal{F}} c_{i,y,f}^t. \tag{4.2}$$

4.2.2 QoE Model

From the perspective of a viewer, the quality of the received video, namely, the received bit rate can greatly determine the viewer's experience [96]. Therefore, we use the term QoE to denote the how good the received video is. The QoE is determined by two factors. First, the acceptable quality levels of the received live videos of different broadcasters vary in terms of their genres (e.g., card game, pixel art game, first shooter game, etc). By categorizing the live videos into a set of genres denoted by $\mathcal{G} = \{1, 2, \ldots, K\}$ and defining $g_i^t \in \mathcal{G}$ as the genre of video from broadcaster i in time slot t, we can define $s_{g_i^t}$ as the basic bit rate that a broadcaster i is suggested to set for the live video in time slot t. Second, it is vital to consider the network capacity of each viewer. Let u_j^t be the highest bit rate that viewer j can receive, which varies due to the viewer network condition. Therefore, the QoE model can be expressed as

$$q_{i,j,y}^t = \log\left(\frac{b_y}{u_j^t} + \frac{b_y}{s_{g_i^t}}\right). \tag{4.3}$$

In (4.3), the QoE model is a non-decreasing concave function of two ratios. The first ratio quantifies the effect of the network condition of viewer j. The higher this ratio is, the better QoE can be achieved. However, this ratio should not exceed one, and a constraint is added to the optimization problem; Otherwise, the viewer capacity is smaller than the bit rate of the transcoded representation and this transcoded representation cannot be smoothly played at the viewer end. The

second ratio quantifies how better the received video quality is compared with the basic genre rate of broadcaster i considering the fact that same representation from different genres of broadcasters can lead to different quality of experience levels. Next, the QoE for a viewer j watching a transcoded video from broadcaster i can be calculated as

$$Q_{i,j}^t = \sum_{y \in \mathcal{Y}} \sum_{f \in \mathcal{F}} \mathbf{V}_{i,j}^t \mathbf{I}_{i,y,f}^t \mathbf{W}_{j,f}^t \alpha_f q_{i,j,y}^t. \tag{4.4}$$

4.2.3 Network Utility

There is a tradeoff between the viewer QoE and the cost imposed on CLSP. On one hand, CLSP prefers to incentivize more fog devices to participate in transcoding and provide ABR service to more viewers. On the other hand, better service will incur more cost. To better express this tradeoff, we define a weighted-difference between the QoE and the cost as

$$U_i^t = \sum_{j \in \mathcal{J}^t} Q_{i,j}^t - \lambda \cdot c_i^t, \tag{4.5}$$

where U_i denotes the network utility in which the parameter λ is used to tune the tradeoff between the two components.

4.3 Edge Transcoding and Viewer Association

4.3.1 Problem Formulation

In this work, we aim to jointly optimize the transcoding task assignment and viewer association by maximizing the total network utility in each time slot over the

whole transcoding system. We therefore, formulate an optimization problem as

$$(\boldsymbol{\mathcal{P}}) \max_{\mathbf{I}^t_{i,y,f}, \mathbf{W}^t_{j,f}} \sum_{i \in \mathcal{I}} U^t_i, \tag{4.6a}$$

$$\textbf{s.t.} \quad \mathbf{C1}: \ \mathbf{V}^t_{i,j} \mathbf{I}^t_{i,y,f} \mathbf{W}^t_{j,f} b_y \leq \min\{u^t_j, B^t_i\} \ , \tag{4.6b}$$

$$\mathbf{C2}: \ \sum_{y \in \mathcal{Y}} \sum_{i \in \mathcal{I}} \mathbf{I}^t_{i,y,f} \leq 1 \ \ \forall f \in \mathcal{F}, \tag{4.6c}$$

$$\mathbf{C3}: \ \sum_{f \in \mathcal{F}} \mathbf{I}^t_{i,y,f} \leq 1 \ \ \forall i \in \mathcal{I}, \forall y \in \mathcal{Y}, \tag{4.6d}$$

$$\mathbf{C4}: \ \sum_{f \in \mathcal{F}} \mathbf{W}^t_{j,f} = 1 \ \ \forall j \in \mathcal{J}^t, \tag{4.6e}$$

$$\mathbf{C5}: \ \mathbf{I}^t_{i,y,f} \in \{0,1\} \ \forall y \in \mathcal{Y}, \forall i \in \mathcal{I}, \forall f \in \mathcal{F}, \tag{4.6f}$$

$$\mathbf{C6}: \ \mathbf{W}^t_{j,f} \in \{0,1\} \ \forall j \in \mathcal{J}^t, \forall f \in \mathcal{F}, \tag{4.6g}$$

where **C1** ensures the received bit rate is lower than both the original video bit rate and the viewer capacity. **C2** ensures that each fog device can at most transcode one specific original live video into one representation. **C3** guarantees that each transcoding task can only be assigned to one fog device. **C4** makes sure that each viewer only receives one video representation in one time slot. **C5** and **C6** guarantee that variable $\mathbf{I}^t_{i,y,f}$ and $\mathbf{W}^t_{j,f}$ are binary.

$\boldsymbol{\mathcal{P}}$ is a non-convex integer programming which is inherently complex to solve. We transform $\boldsymbol{\mathcal{P}}$ by first relaxing the binary variables $\mathbf{I}^t_{i,y,f}$ and $\mathbf{W}^t_{j,f}$ to continuous variables and then introducing an auxiliary variable b such that

$$b \leq \sum_{i \in \mathcal{I}} U^t_i. \tag{4.7}$$

According to (4.1)-(4.5), the formulated objective function is non-convex and maximizing it is challenging. Therefore, we try to maximize b which is the lower bound of the objective function. Additionally, the constraint (4.7) can be rewritten as

$$b \leq \sum_{j \in \mathcal{J}^t} Q^t_{i,j} - \lambda \cdot c^t_i, \tag{4.8}$$

According to the definition of the cost and QoE function in (4.4) and (4.2), we have

$$\frac{b+\lambda\sum_{i\in\mathcal{I}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}c^t_{i,y,f}}{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}^t}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}q^t_{i,j,y,f}}\leq 1, \tag{4.9}$$

Consequently, with the auxiliary variable b and the new constraint in (4.9), the original problem will be transformed and expressed as

$$(\widetilde{\boldsymbol{\mathcal{P}}})\ \max_{\mathbf{I}^t_{i,y,f},\mathbf{W}^t_{j,f},b}\ b, \tag{4.10a}$$

$$\textbf{s.t.}\ \ \textbf{C1}:\ \mathbf{V}^t_{i,j}\mathbf{I}^t_{i,y,f}\mathbf{W}^t_{j,f}b_y\leq\min\{u^t_j,B^t_i\}, \tag{4.10b}$$

$$\textbf{C2}:\ \sum_{y\in\mathcal{Y}}\sum_{i\in\mathcal{I}}\mathbf{I}^t_{i,y,f}\leq 1\ \ \forall f\in\mathcal{F}, \tag{4.10c}$$

$$\textbf{C3}:\ \sum_{f\in\mathcal{F}}\mathbf{I}^t_{i,y,f}\leq 1\ \ \forall i\in\mathcal{I},\forall y\in\mathcal{Y}, \tag{4.10d}$$

$$\textbf{C4}:\ \sum_{f\in\mathcal{F}}\mathbf{W}^t_{j,f}=1\ \ \forall j\in\mathcal{J}^t, \tag{4.10e}$$

$$\widetilde{\textbf{C5}}:\mathbf{I}^t_{i,y,f}\in[0,1]\ \forall y\in\mathcal{Y},\forall i\in\mathcal{I},\forall f\in\mathcal{F}, \tag{4.10f}$$

$$\widetilde{\textbf{C6}}:\mathbf{W}^t_{j,f}\in[0,1]\ \forall j\in\mathcal{J}^t,\forall f\in\mathcal{F}, \tag{4.10g}$$

$$\textbf{C7}:b>0, \tag{4.10h}$$

$$\textbf{C8}:\frac{b+\lambda\sum_{i\in\mathcal{I}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}c^t_{i,y,f}}{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}^t}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}q^t_{i,j,y,f}}\leq 1, \tag{4.10i}$$

where $q^t_{i,j,y,f}=\alpha_f q^t_{i,j,y}\mathbf{V}^t_{i,j}\mathbf{I}^t_{i,y,f}\mathbf{W}^t_{j,f}$. The problem $\widetilde{\boldsymbol{\mathcal{P}}}$ belongs to the category of CGP since the non-convex constraint of **C8** is in the form of a ratio between two posynomials.

In order to solve this problem, we use successive convex approximation method and monomial approximations [147] to transform the problem into a series of GP problems. Applying monomial approximations, the problem can be ex-

pressed as

$$(\widehat{\mathcal{P}}) \max_{\mathbf{I}^t_{i,y,f},\mathbf{W}^t_{j,f},b} \; b, \quad \textbf{s.t.} \quad \textbf{C1}-\textbf{C4}, \widetilde{\textbf{C5}}, \widetilde{\textbf{C6}} \;\&\; \textbf{C7},$$
$$\frac{b+\lambda\sum_{i\in\mathcal{I}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}c^t_{i,y,f}}{\prod_{i\in\mathcal{I}}\prod_{j\in\mathcal{J}^t}\prod_{y\in\mathcal{Y}}\prod_{f\in\mathcal{F}}\left(\frac{q^t_{i,j,y,f}}{\chi_{i,j,y,f}(n)}\right)^{\chi_{i,j,y,f}(n)}} \leq 1,$$

where the parameter $\chi_{i,j,y,f}(n)$ can be obtained by computing

$$\chi_{i,j,y,f}(n) = \frac{q_{i,j,y,f}(n-1)}{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}^t}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}}q_{i,j,y,f}(n-1)}, \quad \forall i\in\mathcal{I}, j\in\mathcal{J}^t, y\in\mathcal{Y}, \forall f\in\mathcal{F}. \tag{4.12}$$

The problem $\widehat{\mathcal{P}}$ can be solved efficiently using CVX [148].

4.3.2 Proposed Algorithm

The proposed algorithm starts with initial values for $\mathbf{I}_0$ and $\mathbf{W}_0$. By solving $\widehat{\mathcal{P}}$ iteratively, we can update $\mathbf{I}$ and $\mathbf{W}$ till all of them converge. Once we have converged values of $\mathbf{I}$ and $\mathbf{W}$, we need to round them into binary values according to the formulated constraints **C2**, **C3**, and **C4**.

4.3.3 Overhead Analysis

In the current CLSPs, real time messaging protocol (RTMP) has been widely used to pull the live stream from the broadcaster and push it to the viewers. RTMP works on top of Transmission Control Protocol, in which the message is divided into small chunks. The required information for the proposed algorithm can be piggybacked within the chunks using RTMP. Such information is only needed at most once in each time slot, which can be expressed with a few bits. Therefore, the communication overhead is trivial compared with the large amount of live video data.

4.4 Numerical Results

4.4.1 Exhaustive Search

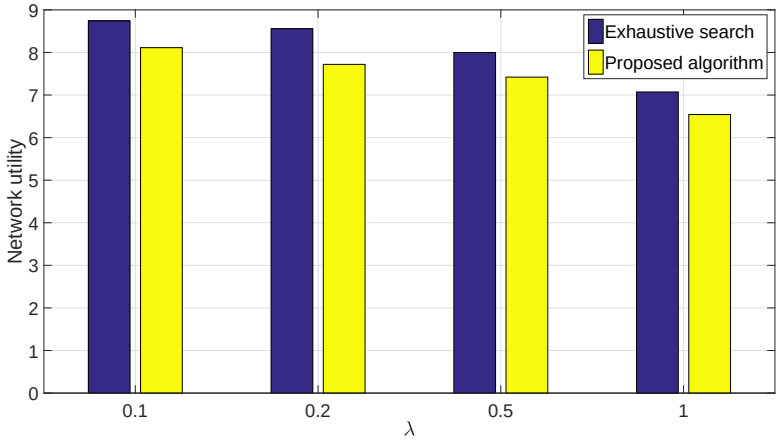

Figure 4.2: Network utility versus λ

To evaluate our algorithm, we first compare the performance of the proposed algorithm with the exhaustive search scheme by setting $I = 2$, $Y = 2$, $F = 3$, and $J^t = 3$. The performance of the proposed scheme for various λ values is collected by running the simulation 10 times and taking the average value of the results. Figure 4.2 depicts the network utility achieved by both schemes versus λ, and confirms that the proposed algorithm closely approaches the globally optimal solution of transcoding task allocation and viewer association.

4.4.2 Trace-Driven Simulation

4.4.2.1 Simulation Setup

In this section, a new simulation is built based on the real-world data of Twitch broadcaster information including the broadcaster IDs, the viewer counts and the original video resolutions [25]. According to the data set, we pick three broadcasters who are consistently online for one hour on 02/01/2015 and dynamically update the viewer counts per five minutes (which is the length of a time slot). In addition, we also find a data set containing the join and leave information of more than 7000 viewers from a chat log online [149]. For each viewer, we calculate the online time duration and estimate the online stability by normalization. Then this information is utilized as the transcoding success rate of each fog device (α_f). 23 viewers are picked to be the fog transcoders with $\alpha_f \in [0.35, 0.63]$. For the representation and the genre rate, we refer to the twitch broadcaster settings [150]. As a result, $Y = 4$

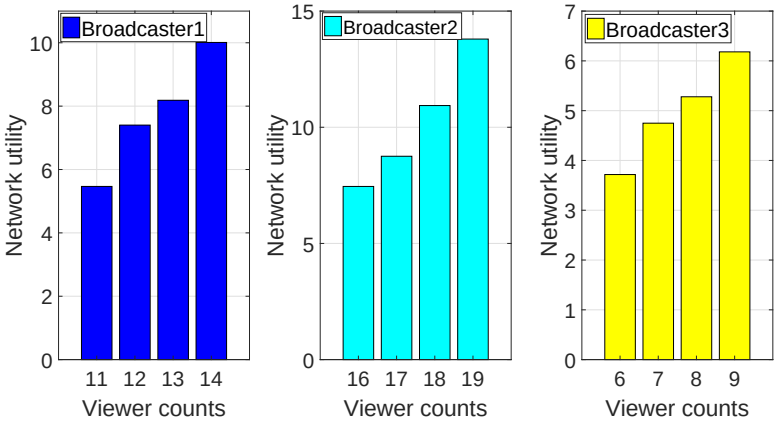

Figure 4.3: Network utility versus viewer counts

and the specific bit rates of the four representations are set to be 400kbps (240P), 1500kbps (480P), 2500kbps (720P), and 5000kbps (1080P). Moreover, we define six viewer categories with the capacity set as $\mathcal{S}$={500kbps, 1000kbps, 2000kbps, 3000kbps, 4500kbps, 6000kbps} and add a bias capacity γ to each viewer capacity $s \in \mathcal{S}$ to explore the effect of the viewer capacities. In addition, we define the function $\phi(y)$ as a logarithmic function of a representation's index, namely $\phi(y) = \log(y)$. Finally, we simulate Top-N which is a currently-running cloud transcoding scheme in Twitch.TV. Top-N offers N premium broadcasters with the ABR service but only the original live video for the rest of broadcasters. To appropriately denote the cost of cloud transcoding, a constant coefficient θ is multiplied to adjust the unit cost of one transcoding task.

4.4.2.2 Performance Evaluation

The results of this trace-driven, data-based simulation are presented in this section. In Figure 4.3, the performance of the proposed scheme is plotted against the viewer counts of each broadcaster with $\lambda = 0.1$. We can find that with the number of viewers increasing, the network utility summed over all broadcasters increases. This is because the cost is paid only for each transcoding task but not for each new viewer. Therefore, when the number of viewers with multiple capacities exceeds a certain threshold, all the representations will be transcoded and the cost will stop increasing.

Figure 4.4 depicts the total cost and QoE values versus λ with different viewer

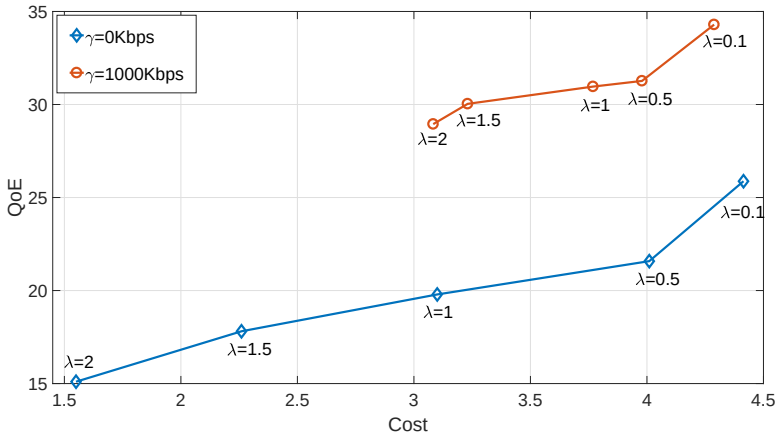

Figure 4.4: Cost and QoE versus λ

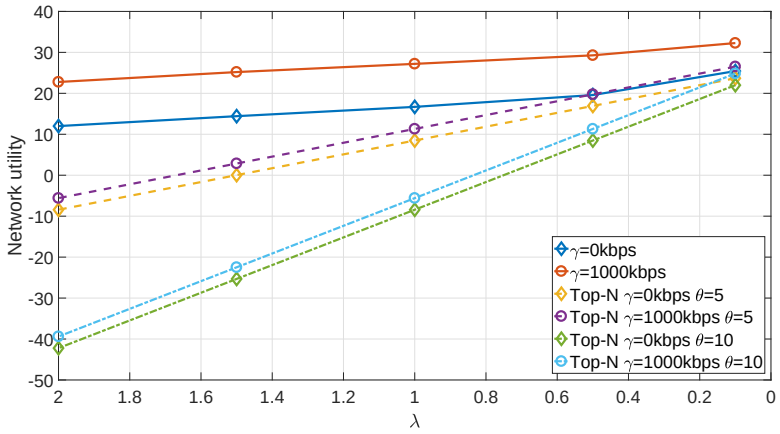

Figure 4.5: Network utility versus λ

capacities. The results demonstrate that with the decrease of λ, both the QoE and the cost tend to increase. Because decreasing λ means that the cost has less impact on network utility thus more representations will be transcoded for the viewers (see Figure 4.5). In addition, the higher viewer capacity will result in better QoE.

In Figs. 4.5 and 4.6, the network utility and the number of transcoded representations are plotted against λ with different viewer capacities. It is evident that both values increase with decreasing λ. The results are consistent with Figure 4.4 and highlight the significance of providing the ABR service to improve the viewer QoE. In addition, Figure 4.5 demonstrates that the proposed algorithm outperforms the Top-N by reaching higher network utility versus λ.

Figure 4.7 describes the relationship between the number of transcoded video representations and the viewer counts. It is shown that with the increase of the

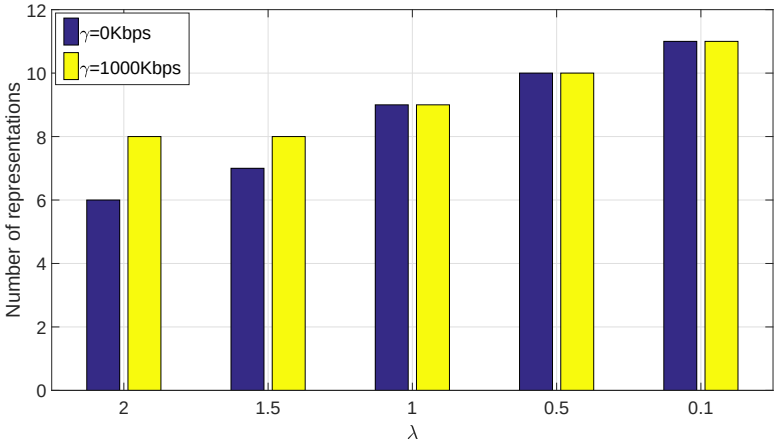

Figure 4.6: Number of representations versus λ

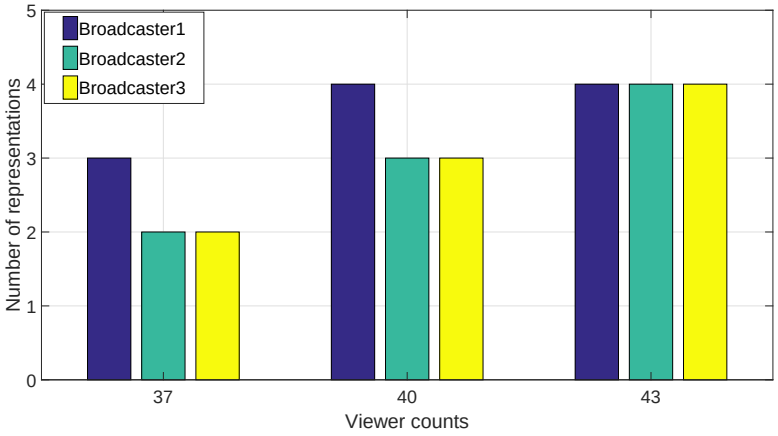

Figure 4.7: Number of representations versus viewer counts

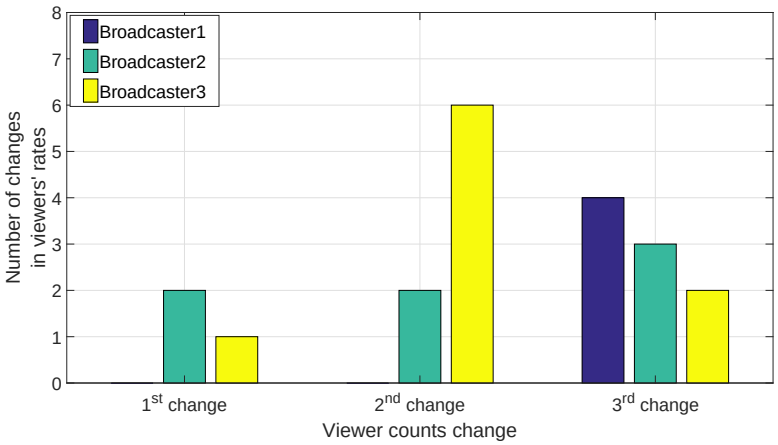

Figure 4.8: Number changes in viewers' rates

viewer counts, the number of representations also increases for each broadcaster, which proves that the proposed algorithm is able to provide viewers with multiple choices of video quality versions according to the viewer capacity.

Figure 4.8 depicts the number of changes in viewers' rates when more viewers join (from 35 to 39, 39 to 40, and 40 to 43). With such changes, around 15 percent of viewers' experienced changes in representations on average. Although new viewers can affect the tradeoff between QoE and cost, most of the viewers receive unchanged representations, which demonstrates the proposed algorithm is able to provide the viewers with robust ABR service.

4.5 Conclusion

In this chapter, a joint edge-assisted transcoding and viewer association algorithm is proposed for the CLSP. In order to maximize the network utility of the system, a non-convex integer problem is formulated and then solved by applying continuous relaxation and monomial approximations. Simulation results demonstrate that the proposed algorithm is able to find the near-optimal solution and is able to provide ABR service to the viewers.

Chapter 5

Risk-Aware Contextual Learning for Edge-Assisted Crowdsourced Live Streaming

5.1 Introduction

In Chapter 4, an edge transcoding system is proposed to leverage the computational resources at the user end. The proposed algorithm relies on solving a non-convex optimization problem in the presence of perfect knowledge of transcoding success rates of viewer devices. However, acquiring such knowledge might not be practically feasible in the live streaming systems. Besides, the notion of the transcoding success rate only involves the effect of the device's online stability which reflects if the fog device can maintain online for transcoding or not. Moreover, the proposed algorithm does not consider the risk of assigning a transcoding task to a device with highly unstable transcoding performance. In this chapter[1], the idea of transcoding success rate is extended by considering both the online stability and the transcoding ability of the fog transcoder, which means we not only consider if the fog device can complete a transcoding task but also consider the performance of the transcoding. Moreover, we also account for the transcoding performance variation during the transcoder selection. In addition, we extend the system model

[1]Part of Chapter 5 has been published in the IEEE Signal Processing Letters [151].

in Section 4.2 by taking the latency experienced by the viewers into account.

To accurately estimate the performance variation of the transcoders, first, we generally study a risk-aware MAB problem based on the MV paradigm. Focusing on the MABs with continuous reward distributions, we first build a finer UCB of the MV with a Gaussian reward assumption and design a Gaussian risk aware-upper confidence bound (GRA-UCB) algorithm to solve the risk-aware MAB problem. We prove that GRA-UCB can reach an $\mathcal{O}(\log(T))$ regret. Next, utilizing the asymptotic distribution of the empirical variance and by extending the GRA-UCB algorithm, a novel asymptotic risk aware-upper confidence bound (ARA-UCB) algorithm is generally designed for sub-Gaussian reward distributions and also proved to achieve $\mathcal{O}(\log(T))$ learning regret. Both proposed algorithms perform numerically well.

Employing the refined confidence bounds and with the aid of the transcoders' context information, a risk-aware contextual learning scheme is designed to learn the transcoding capabilities of fog devices online. Based on the learnt capabilities, a novel joint transcoding task assignment and viewer association algorithm is proposed. The proposed concept of transcoding capability can explicitly reflect the impacts of both the transcoding quality and the performance variation of a transcoder, thus this scheme can determine transcoding task assignment with risk-awareness and avoid assigning transcoding tasks to the transcoders which only perform averagely well. Furthermore, an epoch-based strategy is designed to greatly reduce the switching cost of transcoding task assignments. Numerical results demonstrate that the proposed algorithm achieves significant network utility improvement while keeping the switching cost of task assignment competitively low.

The remainder of this chapter is organized as follows. Section 5.2 describes the system model of the extended edge transcoding system. In Section 5.3, an optimization is formulated to maximize the network utility. In Section 5.4, the risk-aware MAB problem is studied and the proposed algorithms are presented. The designed transcoding task assignment and viewer association algorithm is described in Section 5.5, followed by the simulation results in Section 5.6. The conclusions

are drawn in Section 5.7.

5.2 System Model

Assume there is a set of broadcasters, i.e., $\mathcal{I} = \{1, 2, \cdots, I\}$ in the network as illustrated in Figure 5.1. We consider a time-slotted system. The bit rate of the original live video of broadcaster $i \in \mathcal{I}$ in time slot t is defined as B_i^t. Moreover, consider there are J^t viewers in time slot t and $V_{i,j}^t$ is a binary parameter indicating whether viewer $j \in \mathcal{J}^t = \{1, 2, \cdots, J^t\}$ chooses to watch broadcaster i's live stream in time slot t or not. In addition, denote $y \in \mathcal{Y} = \{1, 2, \cdots, Y\}$ as a video representation which is one of Y possible standard quality levels of a transcoded video and the bit rate of representation y is defined as b_y.

Similar to the system model in Section 4.2, the scheduler of CLSP will decide which representation should be transcoded by which fog device based on viewer requirements, the performance of the fog devices, and system constraints such as the experienced delay by users and the cost for transcoding. In the transcoding process, each fog device $f \in \mathcal{F} = \{1, 2, \cdots, F\}$ is able to transcode the broadcasters' live videos into standard video versions where $\mathcal{F}$ represents the set of all available fog devices. After the transcoding process, all of the standard transcoded live videos will be transmitted from the fog devices to the associated viewers.

To describe the transcoding task assignment and the viewer association, a binary variables is defined as $I_{i,y,f,j}^t$ which takes 1 when fog device f is selected to transcode the original video from broadcaster i requested by viewer j into representation y in time slot t and 0 otherwise.

5.2.1 Cost and QoE Model

To incentivize a fog device to participate in transcoding, similar to the system model in Chapter 4, we define the cost of transcoding one live video from broadcaster i to representation y at fog device f, which is paid by the CLSP for the fog

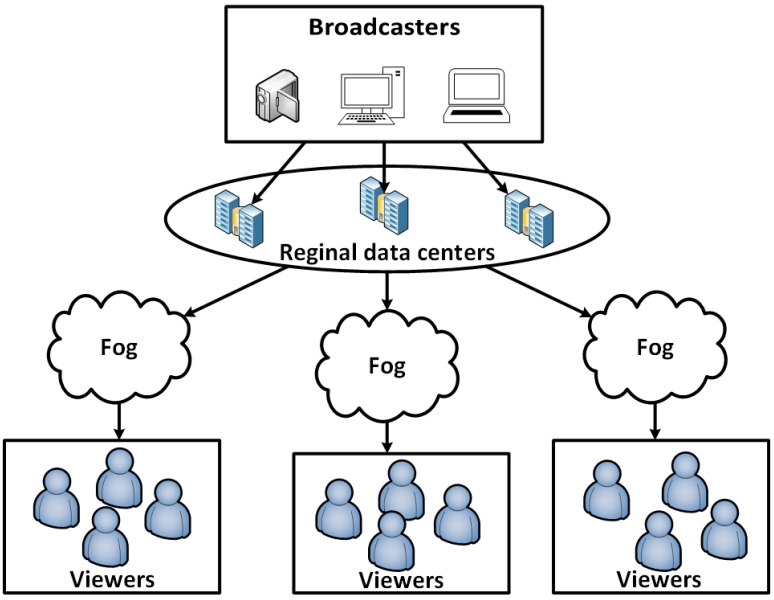

Figure 5.1: System model

device, as $c^t_{i,y,f}$, which is written as

$$c^t_{i,y,f} = \sum_{j \in \mathcal{J}} I^t_{i,y,f,j} \cdot \Phi_y \cdot \omega^t_f \tag{5.1}$$

where Φ_y is a non-decreasing concave function of representation y and ω^t_f is defined as the transcoding capability of device f.

In (5.1), ω^t_f represents the transcoding capability of device f in time slot t. A higher value of ω^t_f means the fog device is more reliable and it can transcode a live stream with higher quality and less delay. To encourage fog devices with higher transcoding capability to join the transcoding candidate pool, $c^t_{i,y,f}$ is linearly increasing with ω^t_f. In this chapter, transcoding capability plays an important role which is not only related to the transcoding quality but also the transcoding performance uncertainty. In a nut shell, a fog device with relatively higher average performance and lower performance fluctuation is more capable for transcoding. The exact definition of transcoding capability will be presented in Section 5.5.

Based on the definition of $c^t_{i,y,f}$, the total cost related to broadcaster i (denoted

by c_i^t) can be defined as

$$c_i^t = \sum_{y \in \mathcal{Y}} \sum_{f \in \mathcal{F}} c_{i,y,f}^t. \tag{5.2}$$

Similar to the definition of the QoE in Chapter 4, we still define the QoE as a function of the representation rate b_y, the source video's genre rate $s_{g_i^t}$, and the viewer network capacity u_j^t, which quantifies both the effect of the viewer's network condition and how much better the received video quality is compared with the basic genre rate of broadcaster. Define the QoE of viewer j watching representation y from broadcaster i as $q_{i,j,y}^t$, which can be written as

$$q_{i,j,y}^t = \log\left(\frac{b_y}{u_j^t} + \frac{b_y}{s_{g_i^t}}\right). \tag{5.3}$$

The QoE for a viewer j watching a transcoded video from broadcaster i can be calculated as

$$Q_{i,j}^t = \sum_{y \in \mathcal{Y}} \sum_{f \in \mathcal{F}} I_{i,y,f,j}^t \omega_f^t q_{i,j,y}^t. \tag{5.4}$$

5.2.2 Delay Model

The latency experienced by the viewers can be categorized into three types, namely, transmission delay, transcoding delay and playout delay. The transmission delay is referred to as the round trip time, including the broadcaster-transcoder delay and the transcoder-viewer delay. In the traditional cloud transcoding system, both the broadcasters and the viewers can be far from the cloud data center, which brings non-negligible latency. The transcoding delay is the processing delay of transcoding a live video to a different quality version. Normally, the transcoding delay can be calculated as the time difference between the input of original video and the output of the transcoded representation. The playout delay is determined by the viewer devices and decoding time, thus it is not involved in this paper. Define ξ_j^t and δ_j^t as the transmission delay and the transcoding delay experienced by viewer j in time

slot t, respectively. The transmission delay can be expressed as

$$\xi_j^t = \sum_{i\in\mathcal{I}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}} \tilde{\tau}_{i,f,j}^t I_{i,y,f,j}^t V_{i,j}^t, \tag{5.5}$$

where $\tilde{\tau}_{i,f,j}^t$ denotes the network delay from broadcaster i to viewer j via fog device f. Next, the transcoding delay can be represented as

$$\delta_j^t = \sum_{i\in\mathcal{I}}\sum_{y\in\mathcal{Y}}\sum_{f\in\mathcal{F}} \tilde{\delta}_{i,y,f} I_{i,y,f,j}^t V_{i,j}^t, \tag{5.6}$$

where $\tilde{\delta}_{i,y,f}$ represents the transcoding delay for fog device f to transcode an original live video with bit rate B_i^t to a representation with bit rate b_y.

Therefore, the overall latency in the proposed transcoding system in time slot t experienced by the viewer j can be represented as

$$D_j^t = \xi_j^t + \delta_j^t. \tag{5.7}$$

5.3 Optimization Problem: Edge Transcoding and Viewer Association

According to the system model formulated in the previous section, there is a tradeoff between the viewer QoE and the cost imposed on CLSP. On one hand, CLSP prefers to incentivize more fog devices to participate in transcoding and provide ABR service to more viewers. On the other hand, better service will incur more cost. To formalize such a tradeoff, we define the weighted-difference between the QoE and cost (which is referred to as the network utility) related to a broadcaster as

$$U_i^t = \sum_{j\in\mathcal{J}^t} V_{i,j}^t Q_{i,j}^t - \lambda \cdot c_i^t, \tag{5.8}$$

where the parameter λ is used to tune the tradeoff between the two components.

Aiming to jointly optimize the transcoding task assignment and viewer asso-

ciation by maximizing the total network utility in each time slot over the whole transcoding system. We therefore, formulate an optimization problem as

$$(\boldsymbol{\mathcal{P}}) \quad \max_{I^t_{i,y,f,j}} \sum_{i\in\mathcal{I}} U^t_i, \tag{5.9a}$$

$$\textbf{s.t.} \quad \textbf{C1}: \; V^t_{i,j} I^t_{i,y,f,j} b_y \leq \min\{u^t_j, B^t_i\}\;, \tag{5.9b}$$

$$\textbf{C2}: \; \sum_{y\in\mathcal{Y}} \sum_{f\in\mathcal{F}} I^t_{i,y,f,j} \leq 1, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}^t, \tag{5.9c}$$

$$\textbf{C3}: \; \sum_{y\in\mathcal{Y}} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}^t} I^t_{i,y,f,j} \leq M, \forall f \in \mathcal{F}, \tag{5.9d}$$

$$\textbf{C4}: \; I^t_{i,y,f,j} \in \{0,1\}, \forall y \in \mathcal{Y}, \forall i \in \mathcal{I}, \forall f \in \mathcal{F}, \forall j \in \mathcal{J}^t, \tag{5.9e}$$

$$\textbf{C5}: \; D^t_j \leq D_{\text{th}}, \forall j \in \mathcal{J} \tag{5.9f}$$

where **C1** makes sure the received bit rate is lower than both the original video bit rate (B^t_i) and the viewer capacity. **C2** ensures that a viewer can only play one representation from one broadcaster. **C3** guarantees that each transcoder can only serve M viewers at most due to the limited bandwidth resource. **C4** guarantees that variable $I^t_{i,y,f,j}$ is binary. **C5** ensures that for every viewer, the experienced delay of each viewer is lower than a predefined threshold D_{th}.

The formulated problem is a linear integer programming that can be efficiently solved by an optimization toolbox called Mosek [152]. Particularly, the transcoding capabilities of transcoders are required to solve $\boldsymbol{\mathcal{P}}$, which is unknown in real live streaming systems. Therefore, an online learning scheme is highly demanded.

5.4 Risk-Aware Learning Algorithm for MABs

In the previous section, to calculate the network utility and to solve the formulated optimization problem, the knowledge of the transcoding capability is indispensable, which needs to be learned online. According to the modelled edge transcoding system, the CLSP controller aims to maximize the cumulative network utility observed in a number of time slots by properly assigning transcoding tasks to multiple transcoders without the prior knowledge of transcoding capability. This decision-making problem can be modelled as an MAB problem, in which the

arms are the transcoders and the agent refers to the CLSP controller. In addition, the unknown transcoding capabilities of fog devices not only reflect the average transcoding performance but also the variation of the performance, so that assigning transcoding tasks to fog devices with large performance variations can be risky. Therefore, in this section, we generally study the risk-aware MAB problem and introduce refined confidence intervals of the variance of arm's reward, which will be used in the following sections to learn the transcoding capability.

5.4.1 Risk-aware MAB Formulation

Consider a risk-aware MAB problem with K arms, i.e., $\mathcal{K} = \{1,2,\ldots,K\}$. Playing each of the arms yields a continuous random reward sampled from an independent distribution. In risk-aware MABs, the risk of receiving a very low reward is considered and the agent prefers playing the arm with a higher mean and lower uncertainty. To measure the risk, in [115], the MV of an arm a is defined as

$$\eta_a = \sigma_a^2 - \rho\mu_a, \tag{5.10}$$

where μ_a and σ_a^2 are the mean and variance of the reward of arm a respectively. In (5.10), $\rho \geq 0$ is a risk-tolerance factor introduced to balance reward-risk tradeoff. As $\rho \to \infty$, the risk-aware MAB problem degenerates to a risk-neutral MAB, and when $\rho = 0$, the problem aims to find the arm with the lowest risk. Defining the arm played at t following the arm-selection policy π as π^t, the observed reward is denoted as $r^t_{\pi^t}$. After playing arms for T rounds, the cumulative MV following the policy π can be calculated as

$$\eta_\pi^T = \mathbb{E}\left[\sum_{t=1}^{T}((r^t_{\pi^t} - \frac{1}{T}\sum_{t=1}^{T} r^t_{\pi^t})^2 - \rho r^t_{\pi^t})\right], \tag{5.11}$$

The objective of the risk-aware MAB problem is to minimize η_π^T under the given risk-tolerance factor ρ.

In order to demonstrate the performance of an arm-selection policy, the optimal policy π^* which has the full knowledge of arms is used as the benchmark to

calculate the performance gap between the optimal policy and a proposed policy π. The performance gap is referred to as the learning regret defined as

$$Reg_{\pi}^{T} = \eta_{\pi}^{T} - \eta_{\pi^*}^{T}. \tag{5.12}$$

In risk-aware MAB, as proved in [118], simply playing the arm with the lowest MV may not be the optimal policy but only a proxy of the optimal policy, that is in general intractable. Therefore, we measure the learning regret by using an approximated policy $\tilde{\pi}^*$ which keeps playing the arm with the smallest MV. Let us define some notations used in the proposed algorithms. Denoting r_a^t as the received reward by playing arm a at round t, the empirical mean $\bar{\mu}_a^t$ and the empirical variance $(s_a^t)^2$ of arm a until round t are defined as

$$\bar{\mu}_a^t = \frac{1}{\tau_a^t} \sum_{d=1}^{\tau_a^t} r_a^{t_a(d)}, \tag{5.13}$$

$$\left(s_a^t\right)^2 = \frac{1}{\tau_a^t - 1} \sum_{d=1}^{\tau_a^t} \left(r_a^{t_a(d)} - \bar{\mu}_a^t\right)^2, \tag{5.14}$$

where $t_a(d)$ represents the round when the d^{th} reward of arm a is observed by the agent and τ_a^t denotes the number of times arm a has been played till round t. Accordingly, the empirical MV can be calculated as

$$\bar{\eta}_a = \left(s_a^t\right)^2 - \rho\, \bar{\mu}_a^t. \tag{5.15}$$

We design two risk-aware bandit algorithms, depending on whether the reward distribution knowledge is available or not. Both algorithms are index-based policies which assign different indexes to each arm. The indexes represent the estimation of the UCB of the MV based on the historical observations, and the arm with the lowest index is played. The key difference is that for GRA-UCB, the reward is assumed to follow Gaussian distribution so the UCB is specifically derived for this case. However, in ARA-UCB, an asymptotic UCB of the variance is derived in the absence of the reward distributions knowledge.

5.4.2 Gaussian Risk aware-Upper Confidence Bound

Suppose that the rewards of arms follow a Gaussian distribution with different means and variances. We utilize the following facts to design an arm-selection algorithm.

Fact 5.1. *(Hoeffding inequality) Let $(X-\mu)$ be a sub-Gaussian random variable. Define the empirical mean over n samples as $\bar{\mu}_n$ and $\mu = \mathbb{E}[X]$, then*

$$\Pr\{\bar{\mu}_n \geq \mu + \kappa\} \leq e^{-2n\kappa^2}, \ \Pr\{\bar{\mu}_n \leq \mu - \kappa\} \leq e^{-2n\kappa^2}. \tag{5.16}$$

Fact 5.1 states that the deviation κ of the empirical mean from the true mean after n samples is bounded by $e^{-2n\kappa^2}$. Here, setting $\kappa = \sqrt{\frac{\log t}{\tau_a^t}}$, then we have

$$\Pr\left\{\bar{\mu}_n \geq \mu + \sqrt{\frac{\log t}{\tau_a^t}}\right\} \leq \frac{1}{t^2}, \ \Pr\left\{\bar{\mu}_n \leq \mu - \sqrt{\frac{\log t}{\tau_a^t}}\right\} \leq \frac{1}{t^2}. \tag{5.17}$$

Fact 5.2. *Let X be a Gaussian random variable with variance σ^2. Define the empirical variance over n samples as s_n^2, based on [153] we have*

$$\Pr\left\{\sigma^2 \leq \frac{(n-1)s_n^2}{\chi^2_{1-\alpha,n-1}}\right\} = 1-\alpha, \tag{5.18}$$

where $\chi^2_{1-\alpha,n-1}$ is the upper 100α percentage points of the chi-square distribution with $(n-1)$ degrees of freedom.

Fact 5.2 gives a definition of the confidence interval of the variance of a random variable when the variable follows the Gaussian distribution. It implies that there is a probability of $100(1-\alpha)\%$ that the constructed confidence interval based on the sample variance will contain the true value of σ^2.

In GRA-UCB algorithm, each arm is assigned with an index which is the UCB of the MV. Based on Facts 5.1 and 5.2, the index of arm a is defined as

$$H_a^t = \frac{(\tau_a^t - 1)\,(s_a^t)^2}{\chi^2_{1-\alpha,\tau_a^t-1}} - \rho\left(\bar{\mu}_a^t + \sqrt{\frac{\log t}{\tau_a^t}}\right), \tag{5.19}$$

where the first and second terms represent the UCBs of the variance and the mean based on (5.18) and (5.17), respectively. In each round, after calculating indexes, the arm with the lowest H_a^t will be played and the number of plays of this arm are updated as $\tau_a(t+1) = \tau_a^t + 1$. Subsequently, the empirical mean and variance are updated based on (5.13) and (5.14).

We derive the learning regret bound of the GRA-UCB algorithm in the following theorem.

Theorem 5.1. *The learning regret of the GRA-UCB algorithm can be upper bounded as*

$$Reg_\pi^T \leq \sum_{a=1}^{K} \left(\left(\frac{4\rho^2}{\Delta_{a,a^*}^2} + C \right) \log T + 3 \right) \left(\Delta_{a,a^*} + \Gamma_{a,a^*}^2 \right) + \sigma_{a^*}^2 + \sigma_{\max}^2 \left(\sum_{a \neq a^*}^{K} \frac{\Gamma_{a,a^*}^2}{\Delta_{a,a^*}} + 1 \right). \tag{5.20}$$

Proof: See Appendix B.

5.4.3 Asymptotic Risk aware-Upper Confidence Bound

GRA-UCB is designed under the assumption that the reward follows an independent Gaussian distribution. However, when the reward distribution is unknown, the confidence interval presented in (5.18) is not pertinent. Following the idea of GRA-UCB, and in this section, we utilize the asymptotic distribution of the empirical variance to drive a confidence interval without any prior assumption of the reward distribution.

Fact 5.3. *Let X be a continuous random variable with mean μ, variance σ^2, and $\mu_4 = \mathbb{E}\left[(X-\mu)^4\right]$. According to [154], the asymptotic distribution of the empirical variance is*

$$\sqrt{n}\left(s_n^2 - \sigma^2\right) \rightarrow \mathcal{N}\left(0, \mu_4 - \sigma^4\right). \tag{5.21}$$

Based on Fact 5.3, in the following lemma, we develop an asymptotic UCB on the variance.

Lemma 5.1. *Applying Fact* 5.3*, for a sufficiently large n, an asymptotic confidence interval of the variance can be derived as*

$$\Pr\left\{\sigma^2 \geq v_n^{\text{upper}}\right\} \leq \alpha, \tag{5.22}$$

where $v_n^{\text{upper}} = \frac{ns_n^2+\sqrt{n\chi_{\alpha,1}^2(\mu_4-s_n^4)+(\chi_{\alpha,1}^2)^2\mu_4}}{n+\chi_{\alpha,1}^2}$ *and* s_n^4 *is the empirical estimate of* σ^4.

Proof: See Appendix C.

Since μ_4 is unknown, an estimate of μ_4 is required. We use $\tilde{\mu}_4 = \frac{1}{n}\sum_{a=1}^{n}(X_a - \bar{\mu})^4$ in the asymptotic UCB provided in Lemma 5.1 to define a refined bound of the MV compared with the GRA-UCB policy. Therefore, an alternative index will be assigned to each arm to determine arm-selection. The index of arm a is defined as

$$N_a^t = v'_{\tau_a^t} - \rho\left(\bar{\mu}_a^t + \sqrt{\frac{\log t}{\tau_a^t}}\right), \tag{5.23}$$

where $v'_{\tau_a^t} = \frac{ns_{\tau_a^t}^2+\sqrt{\tau_a^t\chi_{\alpha,1}^2\left(\tilde{\mu}_4-s_{\tau_a^t}^4\right)+\left(\chi_{\alpha,1}^2\right)^2\tilde{\mu}_4}}{\tau_a^t+\chi_{\alpha,1}^2}$. Similar to GRA-UCB algorithm, the agent chooses the arm with the smallest index to play. We derive the learning regret bound of the ARA-UCB algorithm in the following theorem.

Theorem 5.2. *The learning regret of ARA-UCB algorithm for sub-Gaussian rewards can be upper bounded as*

$$\begin{aligned} Reg_\pi^T \leq \sum_{a=1}^{K}\left((\frac{4\rho^2}{\Delta_{a,a^*}^2}+C)\log T+\beta\log L_a+3\right)\left(\Delta_{a,a^*}+\Gamma_{a,a^*}^2\right)+\sigma_{a^*}^2 \\ +\sigma_{\max}^2(\sum_{a\neq a^*}^{K}\Gamma_{a,a^*}^2/\Delta_{a,a^*}+1). \end{aligned} \tag{5.24}$$

Proof: See Appendix D.

5.4.4 Numerical Results for Risk-aware MABs

In this section, we numerically evaluate the performance of the proposed algorithms and compare them with several benchmarks. We simulate two cases by setting two different reward distributions. In the first simulation, the reward is sam-

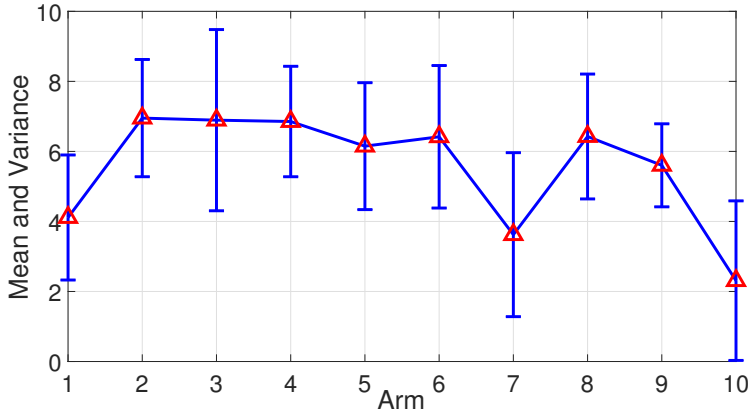

Figure 5.2: The mean and variance of ten arms

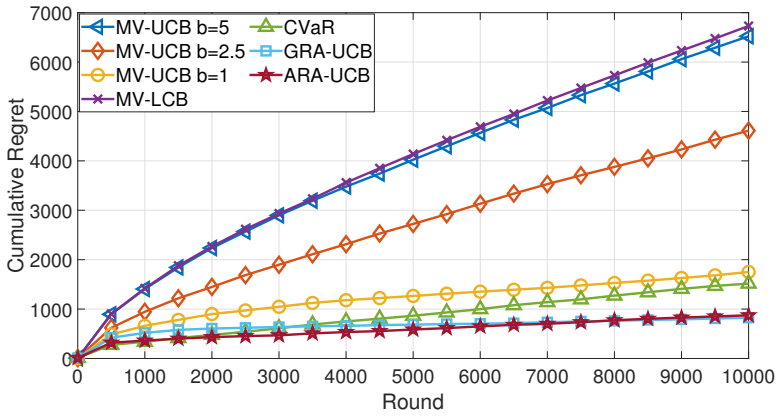

Figure 5.3: The learning regret with Gaussian reward

pled from a Gaussian distribution. The number of arms is set to be $K = 10$ and the risk tolerance $\rho = 1$. The ten pairs of means and variances are uniformly generated (which are illustrated in Figure 5.2). In the second case, the reward is sampled from a truncated Gamma distribution in the range $[0, 10]$. The number of arms is set to be $K = 10$ and the risk tolerance $\rho = 2.5$. The means and variances in the second case are presented in Figure 5.4. Since different reward distributions with different parameters are considered, different risk tolerance factors are used to ensure the optimal arm is risk sensitive with relatively high mean and low variance.

To evaluate the performance of GRA-UCB and ARA-UCB in comparison with the existing schemes, we simulate three algorithms, namely, MV-LCB [115], MV-UCB [118], and CVaR [119]. Finally, we calculate the learning regret by running each simulation for 100 times and taking the average results.

In Figure 5.3, we observe that both proposed algorithms reach low cumulative

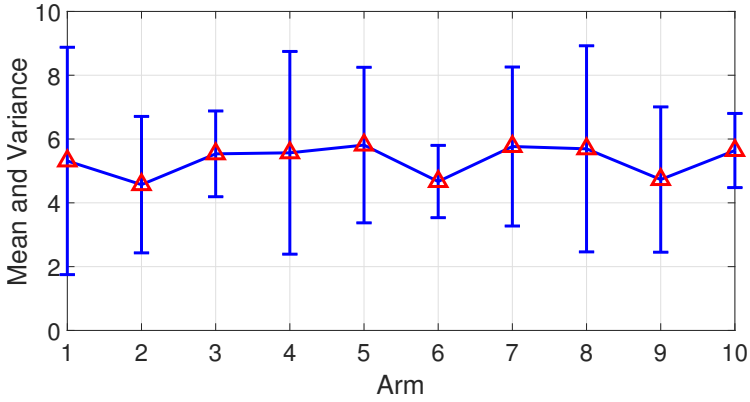

Figure 5.4: The mean and variance of ten arms

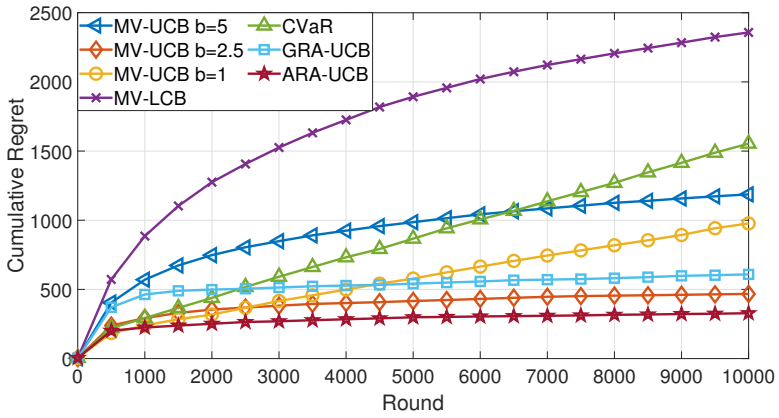

Figure 5.5: The learning regret with truncated Gamma reward

learning regret, because GRA-UCB is designed for the Gaussian reward and ARA-UCB is model agnostic. In addition, although MV-UCB performs relatively well, the performance is sensitive to its parameter b which needs to be adjusted for better performance and no clear hint is discussed in the literature. Moreover, our proposed schemes outperform the MV-LCB because the confidence bound used in MV-LCB is not tight enough.

In Figure 5.5, benefiting from the asymptotic upper confidence bound, ARA-UCB reaches the lowest learning regret. Additionally, we should also note that, because truncated Gamma distribution is used to generate reward while MV-UCB is based on an assumption of symmetric reward distribution, the derived confidence bound of MV-UCB is inaccurate.

5.5 Risk-Aware Contextual Learning for Edge Transcoding

In light of the proposed system model in Section 5.2 and the UCBs of the reward variance in Section 5.4, in this section, a novel risk-aware learning algorithm is designed to learn the transcoding capability ω_f^t online leveraging context information. Then, a novel transcoding task assignment and viewer association algorithm is designed to maximize the network utility of the transcoding system.

Since the aim of the edge-assisted transcoding system is to select a number of fog transcoders per time slot to maximize the network utility, this problem can be modelled as a risk-aware MAB problem, where the arms are the fog devices and the rewards are the transcoding outcomes of selected fog devices. More specifically, we learn the transcoding capabilities of each fog transcoder proposing an index-based MAB algorithm and use the optimization problem formulated in Section 5.3 to determine the transcoder selection while satisfying the constraints.

5.5.1 Transcoding Capability Learning

To solve the optimization problem (5.9), the transcoding capability should be learned since in an edge-assisted transcoding system, transcoders can be the candidate viewer devices whose transcoding capabilities are unknown [25]. Defining a random variable r_f^t as the transcoding outcome when a transcoding task is assigned to transcoder f in time slot t. To describe the average performance of transcoding for a fog device, we define γ_f as the transcoding quality, which is the mean of the transcoding outcome r_f^t of the fog device f. Intuitively, a transcoder with high transcoding quality can still perform poorly if the variation of its transcoding quality is large, which is unaffordable and risky. For instance, choosing a transcoder with high uncertainty can lead to unacceptable transcoding delay and severely deteriorate the viewer experience. Besides, choosing a more risky transcoder can lead to frequent transcoder switches, which means the same transcoding task will be assigned to different fog transcoders and result in high communication overhead and playback latency. These problems reflect the importance of considering the uncertainty

of transcoders. To quantify the variation of the transcoding outcome, we define the variance of the transcoding outcome as σ_f^2.

Based on above definitions and motived by the mean-variance metric [114], to consider both the mean and the variance of the transcoding performance of a fog device, we formally define the transcoding capability ω_f^t as the linear combination of the transcoding quality and the variance of the transcoding outcome, which can be written as

$$\omega_f = \rho\gamma_f - \sigma_f^2. \tag{5.25}$$

where $\rho > 0$ is the risk-tolerance factor introduced to balance the treadoff between a high reward and a low risk.

Since the transcoding capability is assumed to be unknown, an online decision-making scheme is required to learn the transcoding capability while choosing the reliable transcoders. In the following sections, we propose to learn the transcoding quality and the variance of the transcoding outcome.

5.5.1.1 Contextual Learning of Transcoding Quality

In the proposed edge-assisted transcoding system, transcoding results can be determined by some side information (context information). For example, the viewer devices are not specially implemented for video transcoding, and can switch offline during transcoding [24], which will affect how many transcoded videos are available for the viewers. Besides, the computational power and network condition of a fog device can also affect the transcoding outcome by incurring varying latency which can be experienced at the viewer end. However, in the CLSP transcoding system, the instantaneous context information may not accurately reflect the transcoding quality since the transcoding will be executed for some time. Therefore, we choose context information such as average available central processing unit (CPU) cycles and online stability which can reflect the average transcoding performance. Since the transcoding outcome depends on the context information of the transcoder such as available computational resources and transcoder online

stability, the transcoding quality obviously depends on the context information as well. Therefore, by assuming the transcoding quality of a transcoder is the linear combination of its context information and a vector of unknown coefficients θ^*, the transcoding quality can be represented as

$$\gamma_f = \mathbb{E}\left[r_f^t | \mathbf{x}_f\right] = \mathbf{x}_f^\top \theta^*, \tag{5.26}$$

where $\mathbf{x}_f$ represents the z-dimensional context information of fog transcoder f, and θ^* denotes the z-dimensional unknown coefficients which can be treated as the weight of each context information. In addition, the number of the context information types is defined as z.

In order to learn the unknown coefficients θ^*, we realize that given some samples of the transcoding quality and the context information, learning the coefficients belong to a LR problem which can be solved by ridge regression, since ridge regression is a suitable technique to solve linear regression when the number of samples is highly limited.

Define $\mathbf{R}^t$ as the set of transcoding outcomes till time slot t, with the number of transcoding outcomes as m^t. Let W^t be a design matrix of dimension $m^t \times z$ whose row corresponds to the m^t observations till time slot t and column corresponds to the z types of the context information. According to [155], we can acquire the estimated coefficients $\tilde{\theta}^t$ by ridge regression as

$$\tilde{\theta}^t = ({W^t}^\top W^t + I_z)^{-1} {W^t}^\top \mathbf{R}^t, \tag{5.27}$$

where I_z is the z-dimensional identity matrix. Define the estimated transcoding quality as $\tilde{\gamma}_f^t$. Based on the learned knowledge of the unknown coefficients, we can update the estimated transcoding quality using the context information as

$$\tilde{\gamma}_f^t = \mathbf{x}_f^\top \tilde{\theta}^t. \tag{5.28}$$

5.5.1.2 Risk-aware MAB for Transcoding Capability Learning

Since the overall network utility needs to be maximized over time and given unknown transcoding capabilities, the EE dilemma is significant and challenging, which must be carefully addressed in the online decision-making problems. This dilemma means if keep selecting different fog transcoders to do transcoding, we can explore their transcoding capability but this can lead to network utility loss since some transcoders may have poor transcoding capability. However, if we avoid exploring the fog transcoders and only exploit the known knowledge, we may lose the chance of further improving the network utility. Therefore, motivated by the MAB framework [111, 118], to assign transcoding tasks online, we model the transcoder selection problem as a risk-aware MAB problem. According to this model, we study to estimate the UCB of transcoding capability by the learnt knowledge of the transcoding outcome, to solve the EE dilemma. Therefore, the UCBs of both the mean and the variance of the transcoding outcomes of each fog transcoder are derived.

For the UCB of the transcoding quality (γ_f), according to [156], for any $\kappa > 0$, with probability at least $1 - \kappa/T$, the deviation between the estimated transcoding quality and the real transcoding quality can be upper bounded as

$$|\mathbf{x}_f^\top \tilde{\theta}^t - \mathbf{x}_f^\top \theta^*| \leq (\phi + 1)\sqrt{\mathbf{x}_f^\top A^{t^{-1}} \mathbf{x}_f}, \tag{5.29}$$

where $A^t = I_z + W^{t\top} W^t$ and $\phi = \sqrt{\frac{1}{2}\ln\frac{2TF}{\kappa}}$. This UCB can help to estimate the real transcoding quality of each transcoder, which holds with a high probability.

In order to estimate the UCB of the transcoding outcome's variance, we first define the empirical transcoding outcome till t as $\bar{r}_f^t$, the empirical variance $(s_f^t)^2$ of the transcoding outcomes until time slot t can be defined as

$$(s_f^t)^2 = \frac{1}{\tau_f^t - 1} \sum_{d=1}^{\tau_f^t} \left(r_f^{t_f(d)} - \bar{r}_f^t \right)^2, \tag{5.30}$$

where $t_f(d)$ represents the time slot when the d^{th} transcoding outcome of transcoder

f is observed and τ_f^t denotes the number of times that transcoder f has been chosen till t.

Next, according to Fact 5.2 in Section 5.3, a UCB of the variance of a random variable is proposed when the variable follows the Gaussian distribution. Given the UCBs of both the mean and the variance of the transcoding outcomes based on (5.29) and (5.18), the transcoding capability of the fog transcoder f can be written as

$$\tilde{\omega}_f^t = \rho\left(\tilde{\gamma}_f^t + (\phi+1)\sqrt{\mathbf{x}_f^\top A^{t-1}\mathbf{x}_f}\right) - \frac{(\tau_f^t - 1)(s_f^t)^2}{\chi^2_{1-a,\tau_f^t-1}}, \tag{5.31}$$

where the first term in the RHS of (5.31) represents the UCB of the transcoding quality and the second term reflects the variance of the transcoding outcome. In (5.31), τ_f^t represents the number of transcoding tasks which is assigned to transcoder f till time slot t and ${s_f^t}^2$ denotes the empirical variance of the transcoding outcome of transcoder f at time slot t.

Similarly, when the reward distribution information is unknown, according to Lemma 5.1, we can build a new transcoding capability estimation, which can be written as

$$\tilde{\omega}_f^t = \rho\left(\tilde{\gamma}_f^t + (\phi+1)\sqrt{\mathbf{x}_f^\top A^{t-1}\mathbf{x}_f}\right) - v'_{\tau_f^t}, \tag{5.32}$$

5.5.2 Switching Cost

Based on the developed learning algorithm for transcoding capability, we can assign the transcoding tasks to the fog transcoders which are supposed to return relatively high reward and are less risky. However, in order to learn the transcoding capability, simply assigning one transcoding task to different fog transcoders in different time slots is not efficient, because assigning a transcoding task to different transcoders frequently can lead to unaffordable task-switching costs and further increase the communication overheads.

To deal with this problem, according to (5.27), we noticed that whichever

transcoder is selected can contribute in collecting information about the coefficients θ^*, thus can further guide the learning process of the transcoding capability. Therefore, instead of determining the task assignment and viewer association in each time slot, we investigate an epoch-based sampling strategy which means a transcoding is consistently assigned to the same fog device for a finite number of time slots (which is referred as an epoch) and the task reassignments are only proceeded once at the beginning of each epoch. With this strategy, we can greatly reduce the switching cost while keep learning the transcoding capability.

The effectiveness of the epoch-based sampling depends on a well-designed epoch length [157], which is supposed to increase as time continues. Given τ_f^t as the task assignment counter of a fog device f, let $F^t = \left\{ f : \tau_f^t \geq t^\zeta \log t \right\}$ be the set of fog devices whose assigned task numbers are more than $t^\zeta \log t$ till time slot t and assume $\zeta > 0$. Define the smallest counter as

$$\tau_{\min}^t = \min_{f \in F^t} \tau_f^t, \tag{5.33}$$

the length of an epoch till time slot t can be calculated as

$$E^t = \lceil (1+\varepsilon)^{\tau_{\min}^t} \rceil, \tag{5.34}$$

where $\varepsilon > 0$.

5.5.3 Transcoder Selection Algorithm

With the learnt transcoding capability of each fog transcoder, we can exploit the learned knowledge to solve the optimization problem $\mathcal{P}$ in (5.9) and update the estimations based on the observed transcoding outcomes. Since the time slots have been divided into epochs, we only need to solve the optimization problem per epoch. Thus, the computational cost can be greatly reduced.

In addition, according to the optimization problem $\mathcal{P}$, multiple transcoding tasks can be assigned to the same transcoder. However, the computational resources of the transcoders are limited and performing excessive transcoding tasks on one transcoder simultaneously can lead to soaring transcoding delay and exhaust the

Algorithm 5.1 Risk-aware contextual transcoding task assignment and viewer association algorithm

Require: γ, ζ, ρ, $\mathbf{x}_f^t$, ϕ
  for $t \leftarrow 1$ **to** T **do**
    if Current epoch ends **then**
      Update τ_f^t and $\tau_{\min}^t$
      Calculate the epoch length E^t based on (5.34)
      for $f \leftarrow 1$ **to** F **do**
        Calculate the estimated transcoding capability $\tilde{\omega}_f^t$ based on (5.31) or (5.32)
      end for
      Solve optimization problem $\mathcal{P}$ to get $I_{i,y,f,j}^t$
      Execute self-inspection and modify $I_{i,y,f,j}^t$ accordingly
    else
      Maintain the same assignment, $I_{i,y,f,j}^t \leftarrow I_{i,y,f,j}^{t-1}$
    end if
    Observe the transcoding outcomes
    Update $\tilde{\theta}^t$ based on (5.27)
  end for

bandwidth resource. Therefore, to avoid overwhelming the transcoders, every time the optimization variable $I_{i,y,f,j}^t$ is calculated, a self-inspection is executed at every selected transcoder and any transcoder assigned with excessive tasks will offload these tasks to the cloud data center for transcoding.

The detailed risk-aware contextual transcoding task assignment and viewer association algorithm is described in Algorithm 5.1.

5.6 Simulation Results

5.6.1 Simulation Setup

We test the proposed algorithm with a synthetic dataset, which is based on the real-world setting. We assume a live stream transcoding system with 4 broadcasters, 50 viewers, 15 fog devices and 4 representations. The viewer count of each broadcaster live stream is decided by its popularity. The popularity is modelled by Zipf distribution which is normally used for video content popularity modelling (e.g., [52]).

We set the original live video rate and the representation rates according to the

twitch broadcaster settings [150]. The original rates for four broadcasters are set as 4000kbps, 2500kbps, 1500kbps and 500kbps. The specific bit rates of the four representations are set to be 400kbps (240P), 1200kbps (480P), 2000kbps (720P), and 3500kbps (1080P). Moreover, we randomly set the viewer capacity in the range of $[500, 4000]$ kbps.

Based on the system model, a transcoding task can be assigned to multiple fog devices to serve different viewers. Since the transcoders are at network edge which is close to the viewers, the fog devices and the viewers are assumed to be distributed in a region of one square kilometre, and their locations are randomly determined following a uniform distribution.

According to [24], the transcoding capability can be affected by transcoder's computational power. Besides, since the candidate viewers can also undertake transcoding tasks and the viewers with low stability can be offline during transcoding, the online stability of the transcoders should be considered as a factor of transcoding capability. Based on [25], the online stability is generated by sampling the Pareto distribution. As a result, we choose the CPU mark, the average CPU usage, the average RAM usage and the online stability as the context information of fog transcoders.

As discussed in [104], the transcoding delay is calculated based on the required computational resources of a task and the available CPU cycles (determined by the CPU mark and usage) of the transcoder. For the transmission delay, it can be divided into two parts as discussed in Section 5.2.2. The broadcaster-transcoder delay is randomly set in the range of $[200, 300]$ ms according to [96], and the transcoder-viewer delay is set in the range of $[0, 100]$ ms depending on the distance between a viewer and a fog device. To demonstrate the risk-awareness of the designed algorithm, the variance of an fog transcoder follows a uniform distribution with the range of $[0, 0.8]$.

Finally, to test the performance of the designed algorithm, both Gaussian and Gamma distributions are simulated to generate the transcoding outcomes. The parameters are uniformly selected. The LinUCB algorithm [126] which utilizes the

	Risk sensitivity	Context awareness
Proposed scheme	Yes	Yes
LinUCB	No	Yes
MV-UCB	Yes	No

Table 5.1: Comparison with benchmarks on risk sensitivity and context awareness

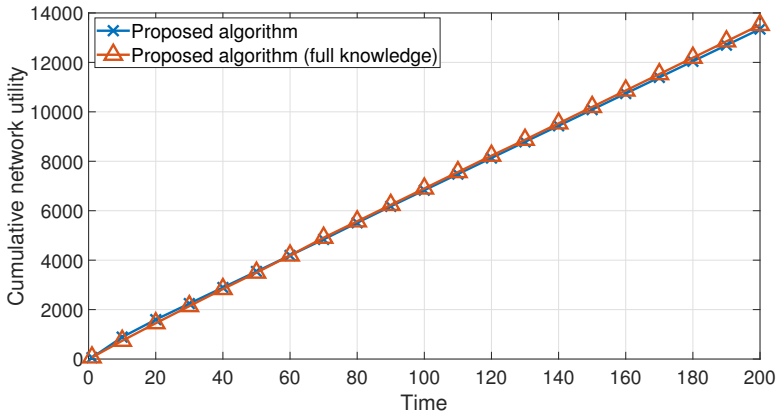

Figure 5.6: The Gaussian reward case

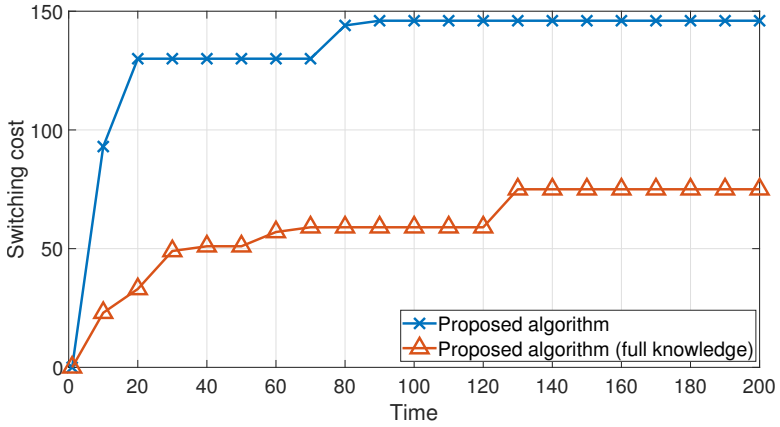

Figure 5.7: The Gaussian reward case

context information to estimate the reward is simulated as the benchmark. In addition, the MV-UCB algorithm [118] which is a risk-aware MAB is also simulated for comparison. Table 5.1 describes the comparison between the proposed algorithm and the benchmarks.

5.6.2 Numerical Results

To demonstrate the performance of the proposed scheme, we first present the transcoding performances with and without the knowledge of the transcoding capa-

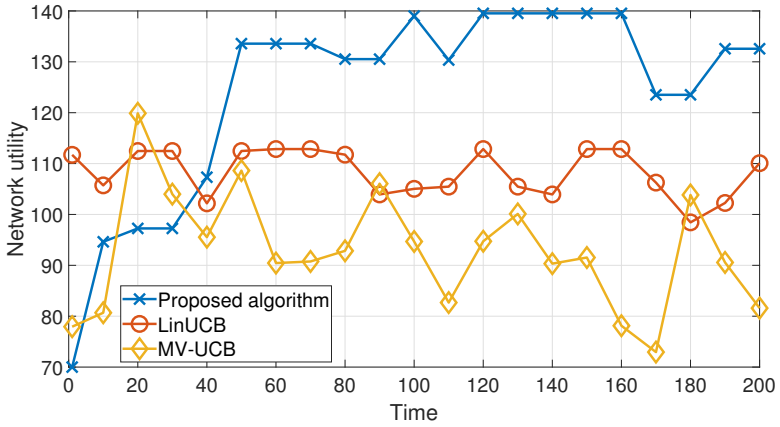

Figure 5.8: The Gaussian reward case

bilities of edge transcoders. In Figures 5.6 and 5.7, the cumulative network utilities and the cumulative switching costs are presented using the Gaussian reward distribution, respectively. According to the figures, we can find that the proposed scheme can learn the transcoding capability quickly and achieve a highly competitively network utility as compared to the case when the transcoding capability is known. Additionally, without the transcoding capability knowledge, Solving the optimization problem (5.9) certainly will cause higher switching costs. This is due to the fact that different estimated values of transcoding capabilities will be passed to the problem and can lead to different solutions. Furthermore, it should be noted that given the full knowledge of the transcoding capabilities, there are still switching costs. This is because we are assuming a time-varying CPU usage which will affect the delay performance and further affect the optimization problem due to the constraint C5 in (5.9).

We first evaluate the proposed algorithm under the Gaussian distribution scenario. In Figure 5.8, the network utilities per time slot achieved by all three algorithms are presented. It is shown that the proposed algorithm outperforms the LinUCB and MV-UCB because it not only utilizes the context information to learn the transcoding quality but also considers the uncertainty of the transcoding outcome. In addition, the cumulative network utilities are depicted in Figure 5.9. Moreover, in Figure 5.10, the cumulative switching costs are presented and the proposed algorithm reaches the lowest cumulative switching cost. This is because the proposed

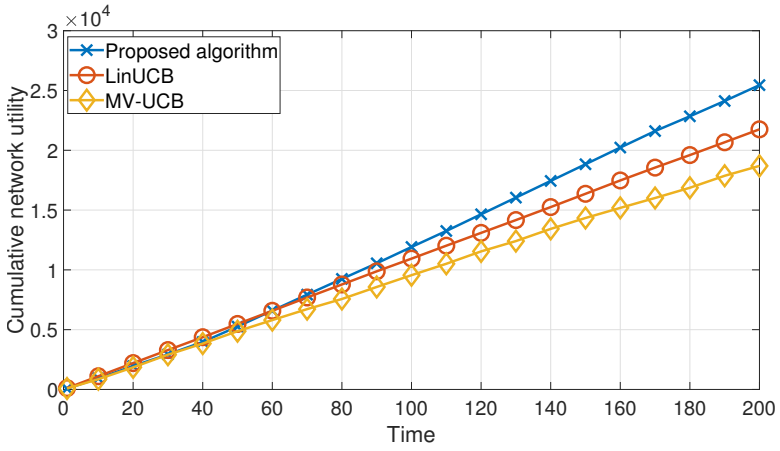

Figure 5.9: The Gaussian reward case

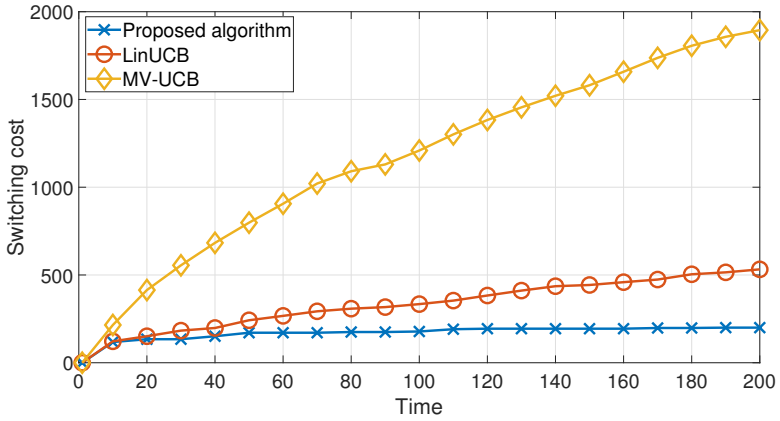

Figure 5.10: The Gaussian reward case

algorithm does not solve the optimization problem $\mathcal{P}$ per time slot so that the task assignment and viewer association will not change frequently.

In Figures 5.11, 5.12, and 5.13, Gamma distribution is used to generate the transcoding outcome. The results demonstrate that the proposed algorithm achieves a higher network utility while maintaining low switching cost. The experienced average latencies per viewer are presented in Table 5.2. According to the results, we can find that the average delays of the proposed algorithm are higher in both scenarios. This is because the proposed algorithm can better quantify the transcoding capabilities of the transcoders and utilize cloud computing in a more efficient way to ensure high network utility. Moreover, the delay thresholds in both simulations are set to 1.3 seconds, which further demonstrates that the proposed algorithm can improve the network utility of the transcoding system while satisfying the delay

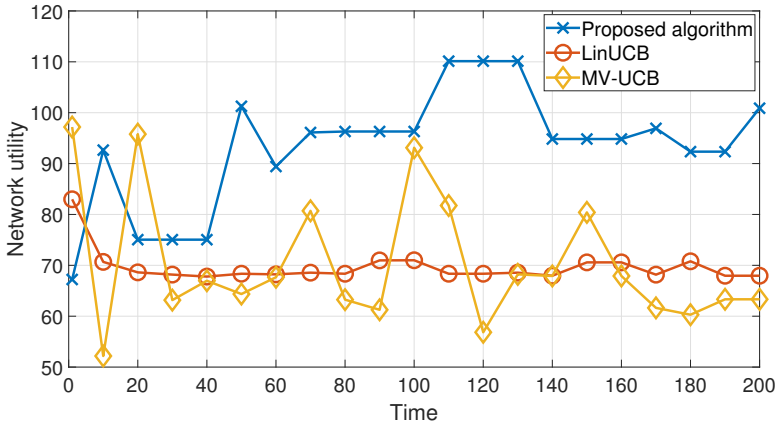

Figure 5.11: The Gaussian reward case

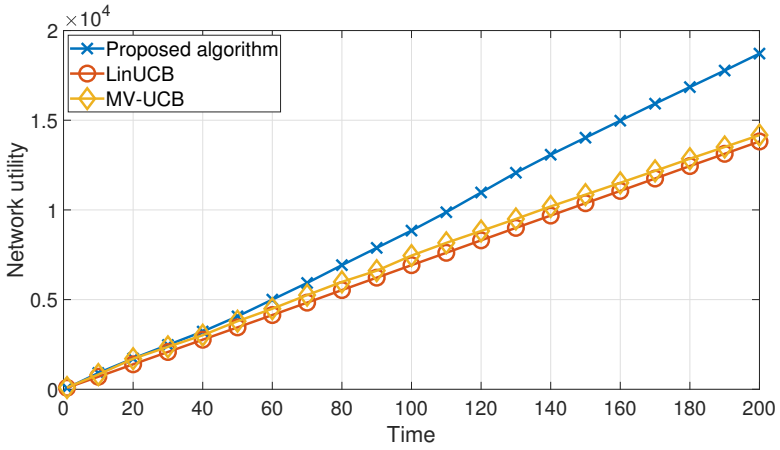

Figure 5.12: The Gaussian reward case

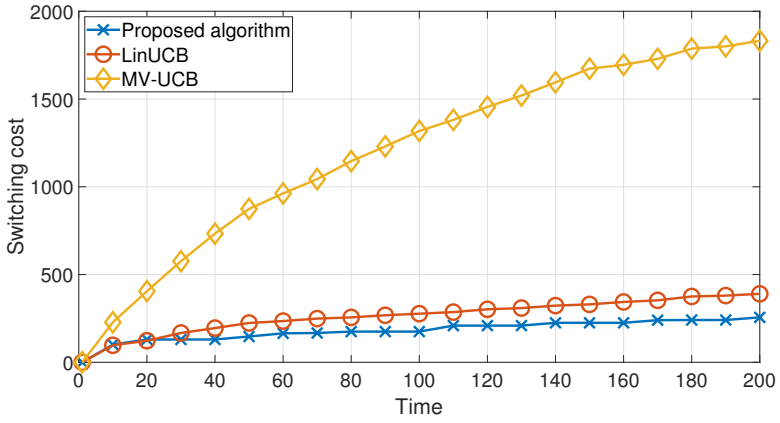

Figure 5.13: The Gaussian reward case

constraint.

	Gaussian distribution case	Gamma distribution case
Proposed algorithm	1.09	1.24
LinUCB	0.46	0.50
MV-UCB	0.44	0.56

Table 5.2: Averaged delay

5.7 Conclusion

In this chapter, a more challenging transcoding task assignment and viewer association problem is formulated, which assumes the transcoding capability of the transcoder is unknown and considers the risk of choosing highly unstable transcoders. To solve the problem, we first design two risk-aware MAB algorithms to minimize the cumulative MV with refined UCBs of the mean and the variance. In light of the UCBs and context information, a risk-aware contextual online learning algorithm is designed to learn the transcoding capability. Moreover, an epoch-based task assignment strategy is designed to reduce the task-switching cost. As a result, a transcoding task assignment and viewer association algorithm is designed. Numerical results demonstrate that the proposed algorithm achieves a significant improvement in the network utility while reducing the switching cost as compared to the benchmarks.

Chapter 6

Transcoder Selection Problem: Structured Bandits

6.1 Introduction

In Chapter 5, we study the transcoding task assignment problem which is modelled as a risk-aware contextual MAB problem. To capture the transcoding performance correlation among different fog transcoders, context information of the fog transcoder is collected to estimate the transcoding capability. In this chapter [1], we still focus on solving the transcoding task assignment but consider a different scenario when the context information cannot be shared with the central controller of the CLSP due to reasons such as possible large communication overhead and privacy concern from the viewers [83]. To solve the transcoding task assignment problem under this circumstance, we study the structured MAB to model the correlation among transcoders and determine transcoder selection online. In this problem, the expected reward function of an arm is a function of some parameters which are shared by all the arms.

As compared to the existing structured bandit problem, an extended model is considered in this chapter, which removes the restrictions on the properties of the reward function and the number of unknown parameters. In order to solve the structured bandit problem, we first build a confidence set which is a set of parameters

[1] Part of Chapter 6 has been published in the 2021 International Conference on AI in Information and Communication (ICAIIC) [158].

whose expected reward is close to the empirical mean rewards of each arm, and then a novel technique is designed to estimate the true value of the unknown parameter based on the established confidence set. On top of that, an enhanced TS-based algorithm is designed to sequentially select fog transcoders while handling the EE dilemma.

Simulation results demonstrate that the proposed GI-TS algorithm can solve the structured bandit problem with a noteworthy improvement of the learning regret compared with the existing benchmarks. In addition, the proposed algorithm is applied to solve the fog transcoder selection problem in live video transcoding system. The results proved that the proposed algorithm can solve this problem with a performance improvement as compared to the benchmark schemes.

The remainder of this chapter is organized as follows. Section 6.2 describes the system model of the structured bandit problem. The proposed GI-TS algorithm is described in Section 6.3. The simulation results are presented in Section 6.4, followed by the conclusions drawn in Section 6.5.

6.2 System Model

Consider a structured bandit problem with K arms, i.e., arm $k \in \mathcal{K} = \{1, 2, \ldots, K\}$. At each round, one of the K arms must be selected by an agent to play, and the reward of the played arm can be observed. Define the arm played at round t as k_t where $k_t \in \mathcal{K}$, the observed reward of playing arm k_t can be defined as R_{k_t}, which is a random variable following an unknown probability distribution. As a structured bandit problem, the expectation of R_{k_t} can be defined as

$$\mu_k(\boldsymbol{\theta}^*) = \mathbb{E}\left[R_{k_t} \mid \boldsymbol{\theta}^*\right], \tag{6.1}$$

where $\boldsymbol{\theta}^* = [\theta_1, \theta_2, \cdots \theta_N]$ is a fixed but unknown parameter vector, which belongs to a parameter set Θ and is shared by all the arms. In addition, the expected reward function of each arm which is known by the agent can be of any form. The observed rewards are assumed to be sub-Gaussian with variance proxy σ^2, which is known to the agent. This assumption is common in MAB [125].

The goal of the agent is to select arm k_t at each round t to maximize the total reward in any T rounds. Assume there is an oracle policy that knows the true value of $\boldsymbol{\theta}^*$ and always selects the optimal arm which is defined as $k^* = \text{argmax}_{k\in\mathcal{K}} \mu_k(\boldsymbol{\theta}^*)$. The cumulative reward following this policy can be represented as $\sum_{t=1}^{T} \mu_{k^*}(\boldsymbol{\theta}^*)$. If the agent misses choosing the optimal arm at round t, a reward loss is incurred, which is defined as $\mu_{k^*}(\boldsymbol{\theta}^*) - \mu_{k_t}(\boldsymbol{\theta}^*)$. Therefore, the expected cumulative learning regret can be defined as

$$\text{Reg}(T) := \mathbb{E}\left[\sum_{t=1}^{T} \mu_{k^*}(\boldsymbol{\theta}^*) - \mu_{k_t}(\boldsymbol{\theta}^*)\right]. \tag{6.2}$$

It is noticed that minimizing the cumulative learning regret equals to maximizing the cumulative reward.

6.3 Proposed Algorithm

In structured bandit, the agent knows the forms of reward functions which are functions of $\boldsymbol{\theta}^*$. Therefore, if the unknown parameter vector can be accurately estimated, the optimal arm can be determined surely. Moreover, since all arms share $\boldsymbol{\theta}^*$, no matter which arm is played, the observed reward can be used to refine the estimation of $\boldsymbol{\theta}^*$. In this regard, enlightened by [125], a TS-based algorithm called GI-TS is designed to utilize the feature of the setting to solve the structured bandit problem.

6.3.1 Confidence Set Construction

Define the empirical mean reward of arm k at round t as $\bar{\mu}_k(t)$ and define the number of times that arm k has been played till round t as $n_k(t)$. GI-TS starts to estimate the unknown parameter vector by constructing a confidence set based on the following fact:

Fact 6.1. *(Hoeffding inequality) Let Z be a random variable which follows a σ^2-Gaussian distribution with mean $\mu(\boldsymbol{\theta}^*)$, then*

$$Pr(|\mu(\boldsymbol{\theta}^*) - \bar{\mu}| \geq \delta) \leq 2\exp\left(-\frac{\delta^2 n}{2\sigma^2}\right), \tag{6.3}$$

where $\bar{\mu}$ represents the empirical mean of Z, and n denotes the number of samples. When $\delta = \sqrt{\frac{2\alpha\sigma^2 \log t}{n}}$, Hoeffding inequality can be bounded as

$$\Pr\left(|\mu(\boldsymbol{\theta}^*) - \bar{\mu}| \geq \sqrt{\frac{2\alpha\sigma^2 \log t}{n}}\right) \leq 2t^{-\alpha}, \tag{6.4}$$

which implies that the gap between the empirical mean and the real mean being larger than a threshold is increasing unlikely to happen with t increasing. α is introduced to control the variation rate of the bound.

According to Fact 6.1, corresponding to each arm, define a set of parameters as

$$A_k(\boldsymbol{\theta}) = \left\{\boldsymbol{\theta} : |\mu_k(\boldsymbol{\theta}) - \bar{\mu}_k(t)| < \sqrt{\frac{2\alpha\sigma^2 \log t}{n_k(t)}}\right\}. \tag{6.5}$$

The confidence set $\tilde{\Theta}_t$ can be calculated as the intersection of $A_k(\boldsymbol{\theta})$'s of all the arms, which can be written as

$$\tilde{\Theta}_t = \bigcap_{k \in \mathcal{K}} A_k(\boldsymbol{\theta}). \tag{6.6}$$

An example of confidence set construction is illustrated in Figure 6.1. The marked curves represent the reward functions of the two arms. For each arm, a set of parameters can be found, so the gap between $\bar{\mu}_k$ and $\mu_k(\boldsymbol{\theta})$ is smaller than $\sqrt{\frac{2\alpha\sigma^2 \log t}{n_k(t)}}$. By taking the intersection of the two sets, the confidence set is established. In this case, we can find that $\boldsymbol{\theta}^*$ falls into the confidence set.

Based on (6.5), define $A_k^{\mathrm{C}}(\boldsymbol{\theta})$ as the complement of $A_k(\boldsymbol{\theta})$. Subsequently, the probability that $\boldsymbol{\theta}^*$ does not belong to the confidence set at round $t+1$ can be written

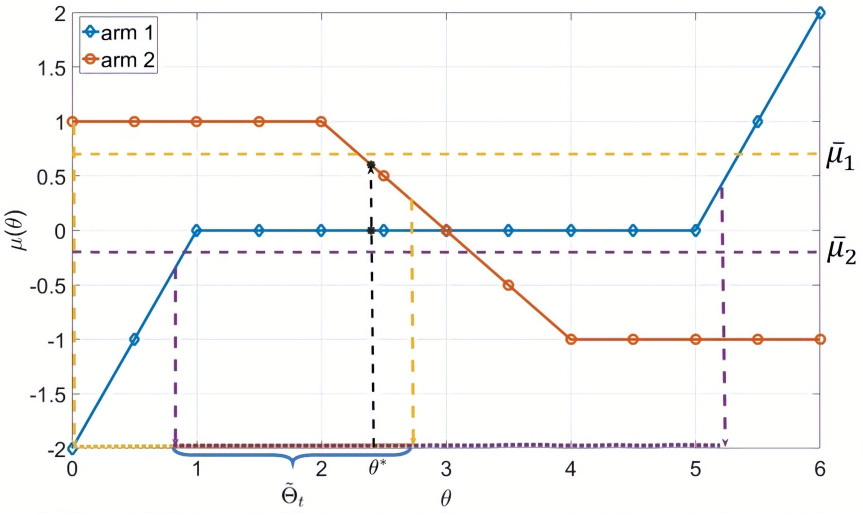

Figure 6.1: Confidence set construction

as

$$\begin{aligned}
\Pr\left(\boldsymbol{\theta}^* \notin \tilde{\Theta}_t\right) =& \Pr\left(\bigcup_{k\in\mathcal{K}} A_k^{\mathrm{C}}(\boldsymbol{\theta})\right), \\
\leq& \sum_{k=1}^{K} \Pr\left(|\mu_k(\boldsymbol{\theta}^*) - \bar{\mu}_k(t)| \geq \sqrt{\frac{2\alpha\sigma^2 \log t}{n_k(t)}}\right), \\
\leq& \sum_{k=1}^{K} \sum_{n_k(t)=1}^{t} \Pr\left(|\mu_k(\boldsymbol{\theta}^*) - \bar{\mu}_k(t)| \geq \sqrt{\frac{2\alpha\sigma^2 \log t}{n_k(t)}}\right), \\
\leq& K \sum_{n_k(t)=1}^{t} 2t^{-\alpha} = 2Kt^{1-\alpha}. \qquad (6.7)
\end{aligned}$$

Therefore, $\Pr\left(\boldsymbol{\theta}^* \in \tilde{\Theta}_t\right) \geq 1 - 2Kt^{1-\alpha}$, which means as t increases, the probability of $\boldsymbol{\theta}^*$ falling into the confidence set $\tilde{\Theta}_t$ superlinearly converges to 1 with $\alpha > 2$.

6.3.2 Unknown Parameter Estimation

After establishing the confidence set, we need to estimate the unknown parameter vector $\boldsymbol{\theta}^*$. Since the confidence set maintains a group of parameters whose corresponding expected rewards are close to the empirical mean rewards of each arm, the confidence set can be used to estimate $\boldsymbol{\theta}^*$. This is because the more times an agent plays, the higher is the probability that the true value of $\boldsymbol{\theta}^*$ falls into this set as it is presented in (6.7). Moreover, in order to utilize the knowledge of the

reward functions, we do not simply estimate $\boldsymbol{\theta}^*$ by directly solving the equation $\bar{\mu}_k(t) = \mu_k(\boldsymbol{\theta})$, which can incur high computation complexity and hardly guarantee a sufficiently accurate solution. Because the reward function can be non-monotonic in our problem, solving the equation may lead to multiple solutions.

In GI-TS, define a parameter in the confidence set as $\boldsymbol{\theta_n}$. To estimate $\boldsymbol{\theta}^*$, $\boldsymbol{\theta_n}$ is uniformly selected within the confidence set. Then, inspired by [130], the estimated parameter $\hat{\boldsymbol{\theta}}_t$ can be calculated as

$$\hat{\boldsymbol{\theta}}_t = \sum_{\boldsymbol{\theta_n} \in \tilde{\Theta}_t} \boldsymbol{\theta_n} w_{\boldsymbol{\theta_n}}(t), \tag{6.8}$$

where $w_{\boldsymbol{\theta_n}}(t)$ represents the weight of $\boldsymbol{\theta_n}$ at round t. The weight can be calculate as

$$w_{\boldsymbol{\theta_n}}(t) = \frac{G_{\boldsymbol{\theta_n}}(t)^{-1}}{\sum_{\boldsymbol{\theta_n} \in \tilde{\Theta}_t} G_{\boldsymbol{\theta_n}}(t)^{-1}}, \tag{6.9}$$

where $G_{\boldsymbol{\theta_n}}(t) = \sum_{k \in \mathcal{K}} |\mu_k(\boldsymbol{\theta_n}) - \bar{\mu}_k(t)|$. The logic behind is that, instead of calculating the mean value of the confidence set, the parameter whose expected reward is closer to the empirical mean reward has a greater impact on the estimated unknown parameter vector. Therefore, the estimation of $\boldsymbol{\theta}^*$ is accomplished by calculating the weighted summation of the parameters inside the confidence set.

6.3.3 Arm Selection

After calculating $\hat{\boldsymbol{\theta}}_t$, we need to utilize it for arm selection to minimize the learning regret. However, we still need to balance the tradeoff between exploring the reward of each arm and exploiting the learned knowledge to play the best arm, because the empirical mean reward can be inaccurate especially at the early stage. Therefore, a Thompson sampling-based scheme namely GI-TS is designed to solve the EE dilemma.

Consider (6.5), (6.6), and (6.8), if $\boldsymbol{\theta}^* \in \tilde{\Theta}_t$, given $\hat{\boldsymbol{\theta}}_t \in \tilde{\Theta}_t$, for each arm, it can be inferred that $|\mu_k(\boldsymbol{\theta}^*) - \mu_k(\hat{\boldsymbol{\theta}}_t)| \leq \sqrt{\frac{2\alpha\sigma^2 \log t}{n_k(t)}}$. Since the confidence set is the intersection among $A_k(\boldsymbol{\theta})$ of different arms, the worst case occurs if $n_k(t) = t/K$, which leads to the possibility of having the largest gaps between $\mu_k(\boldsymbol{\theta}^*)$ and $\mu_k(\hat{\boldsymbol{\theta}}_t)$.

Algorithm 6.1 GI-TS

Require: Reward functions $\{\mu_1, \mu_2, \cdots, \mu_k\}$, $n_k \leftarrow 0$, t_h
  for $t \leftarrow 1$ **to** T **do**
    if $t < t_\text{h}$ **then**
      Randomly play an arm $k_t \in \mathcal{K}$
      Observe R_{k_t}
      $n_{k_t}(t) = n_{k_t}(t-1) + 1$
      $\bar{\mu}_{k_t}(t) = \frac{\bar{\mu}_{k_t}(t-1) \cdot n_{k_t}(t-1) + R_{k_t}}{n_{k_t}(t)}$
    else
      Construct confidence set $\tilde{\Theta}_t$ based on (6.5) and (6.6)
      Calculate $\hat{\boldsymbol{\theta}}_t$ based on (6.8), and (6.9)
      for $k \in \mathcal{K}$ **do**
        Sample $\tilde{R}_k(t) \sim \mathcal{N}\left(\mu_k(\hat{\boldsymbol{\theta}}_t), \frac{\sigma^2}{t}\right)$
      end for
      Play arm $k_t = \text{argmax}_{k \in \mathcal{K}} \tilde{R}_k$, observe R_{k_t}
      $n_{k_t}(t) = n_{k_t}(t-1) + 1$
      $\bar{\mu}_{k_t}(t) = \frac{\bar{\mu}_{k_t}(t-1) \cdot n_{k_t}(t-1) + R_{k_t}}{n_{k_t}(t)}$
    end if
  end for

Therefore, in the worst case, the gap between $\mu_k(\hat{\boldsymbol{\theta}}_t)$ and $\mu_k(\boldsymbol{\theta}^*)$ can be upper bounded as

$$|\mu_k(\boldsymbol{\theta}^*) - \mu_k(\hat{\boldsymbol{\theta}}_t)| \leq \sqrt{\frac{2K\alpha\sigma^2 \log t}{t}}, \tag{6.10}$$

which means the estimated reward $\mu_k(\hat{\boldsymbol{\theta}}_t)$ will converge to its real reward as t increases.

Assuming that the likelihood of the reward is given by the probability density function of Gaussian distribution. we propose an improvement to compute the posterior distribution for *structured* bandits considering that any play is informative for all arms and thus will reduce the variance as σ^2/t. Therefore, we will generate an independent sample $\tilde{R}_k(t)$ from the distribution $\mathcal{N}\left(\mu_k(\hat{\boldsymbol{\theta}}_t), \frac{\sigma^2}{t}\right)$ for each arm. The agent will then play the arm with the highest $\tilde{R}_k(t)$. The pseudocode of GI-TS is presented in Algorithm 6.1.

	Number of arms	Length of $\boldsymbol{\theta}^*$
First scenario	3	1
Second scenario	10	1
Third scenario	3	2

Table 6.1: Simulation setting

6.4 Simulation Results

6.4.1 Simulation Setup

In this section, the performance of the GI-TS is evaluated. We first generate a range of different reward functions to demonstrate the robustness and performance of the proposed algorithm, and three scenarios are set by changing the number of arms and the length of the unknown parameter vector. The settings are depicted in Table 6.1. In addition, we study a transcoder selection problem in live streaming transcoding application which can be modelled as a structured bandit problem.

In the simulations, α is set to be 3. We choose the variance of each arm in all simulations as $\sigma^2 = 4$, and R_k is drawn from $\mathcal{N}(\mu_k(\boldsymbol{\theta}^*), 4)$. t_{h} is set to be 10, which is the number of initialization rounds for arm selection. In addition, the simulations run for 10000 rounds each time and the learning regret is presented as the average of 100 independent experiments.

6.4.2 Benchmarks

Several existing algorithms are simulated as benchmarks. First, the classical scheme UCB and TS are simulated. In addition, we also simulate the UCBc and TSc schemes which are specially designed for the structured bandit problem [159]. In these two schemes, first a confidence set of parameters is built with the knowledge of reward functions, and some of the arms are identified to be the competitive arms with the help of the confidence set. After that, either UCB or TS is applied to select only one of the competitive arms.

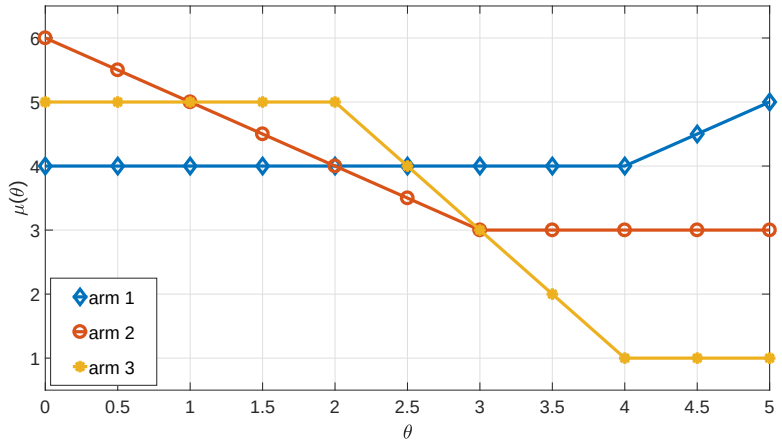

Figure 6.2: Reward function versus $\boldsymbol{\theta}$

6.4.3 Numerical Results

6.4.3.1 First Scenario

In Figure 6.2, the reward functions of arms versus $\boldsymbol{\theta}$ are presented. We can find that arm 1 is optimal when $\boldsymbol{\theta}^* \in [2.5, 5]$, arm 2 is optimal when $\boldsymbol{\theta}^* \in [0, 1]$, and arm 3 is optimal when $\boldsymbol{\theta}^* \in [1, 2.5]$.

With this setting, we test the proposed algorithm and the benchmarks with varying θ^*. In Figure 6.3, it is apparent that GI-TS and TSc outperform classical schemes by considering the correlation among the arms so that the learning regret is greatly reduced. This is because no matter which arm is played, the observed reward can help to find the shared unknown parameter θ^* by refining the confidence set each round and improve the learning speed. In addition, our proposed GI-TS algorithm reaches the lowest regret compared with all the benchmarks. This is because GI-TS directly estimates the value of θ^* to find the optimal arm with the help of reward function information. Noted that when $\theta^* = 4.8$, TSc also performs well as compared to GI-TS because when the unknown parameter locates at the edge of Θ, taking the weighted summation of the confidence cannot estimate the unknown parameter very accurately. However, the results are still comparable and GI-TS is more robust with varying θ^*.

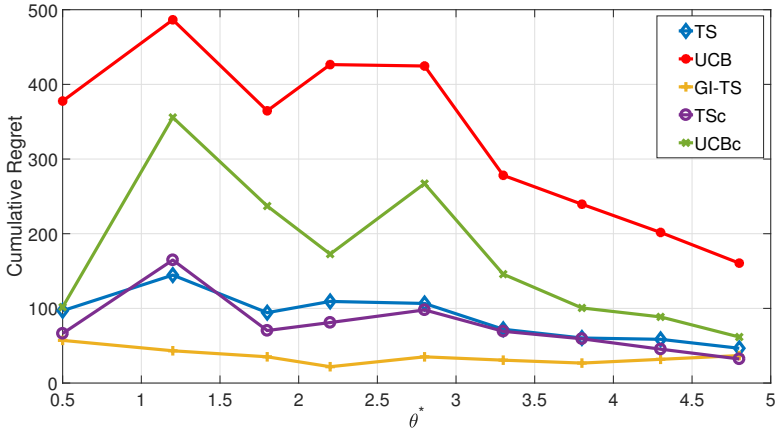

Figure 6.3: Cumulative regret versus $\boldsymbol{\theta}^*$

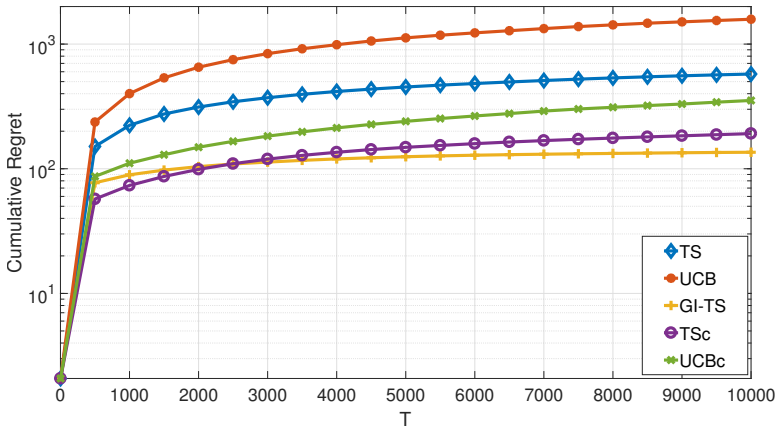

Figure 6.4: Cumulative regret when $\boldsymbol{\theta}^* = 1.5$

6.4.3.2 Second Scenario

In this section, we further test the proposed algorithm in a more complex setting with ten arms whose reward functions are similar to the functions in the first scenario.

From Figures 6.4, 6.5, 6.6, and 6.7, we can find that GI-TS outperforms all the benchmarks under different values of θ^*. When $\theta^* = 3.8$ or 4.4, the learning regret of GI-TS is only one fifth of the regret incurred by the TSc scheme. Compared with the first scenario, GI-TS scales well when more arms are added into the system. This is because, instead of using the number of plays for one arm to estimate the variation of the reward distribution of that arm, which is used in the normal TS scheme, the total number of rounds is utilized based on the fact that the confidence set is updated in each round no matter which arm is played. This novel method

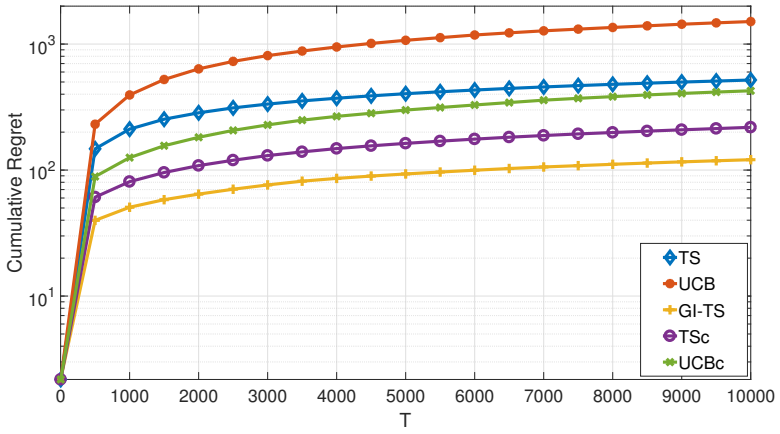

Figure 6.5: Cumulative regret when $\boldsymbol{\theta}^* = 2.7$

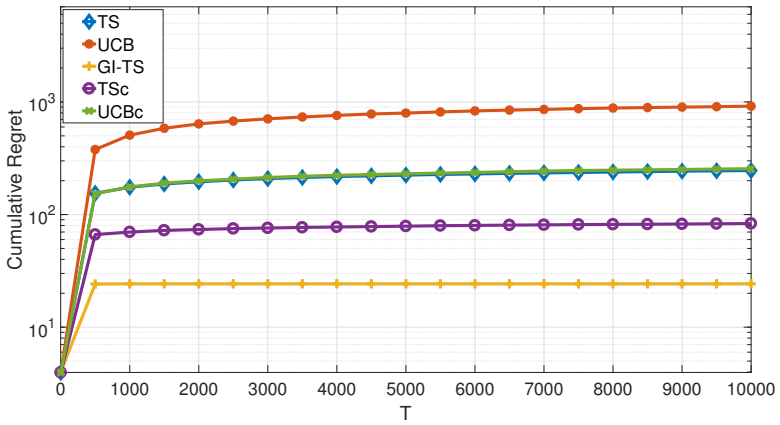

Figure 6.6: Cumulative regret when $\boldsymbol{\theta}^* = 3.8$

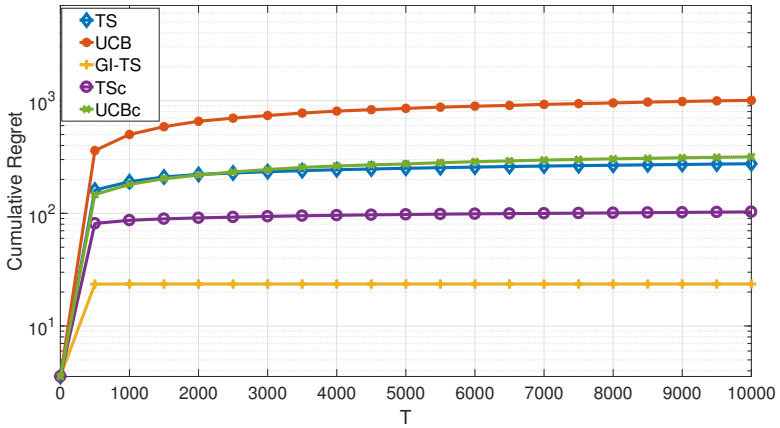

Figure 6.7: Cumulative regret when $\boldsymbol{\theta}^* = 4.4$

improves the convergence rate and helps to reach a competitive learning regret.

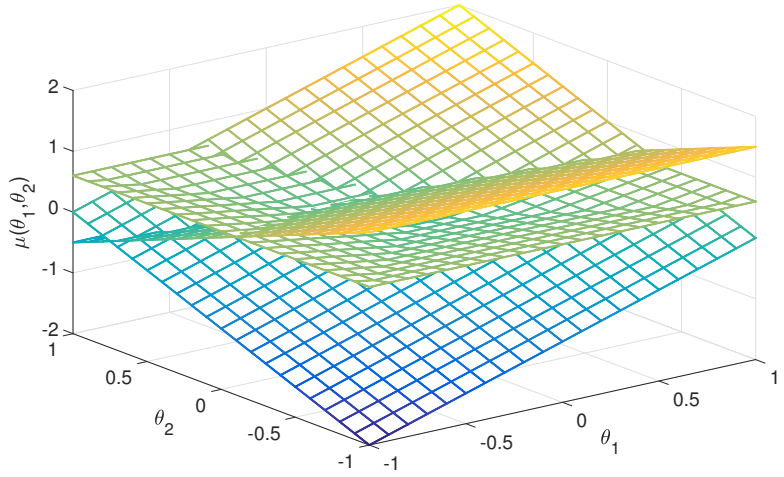

Figure 6.8: Reward function versus θ_1 and θ_2

6.4.3.3 Third Scenario

In this section, we test the proposed algorithm when the unknown parameter $\boldsymbol{\theta}^*$ is a 2-dimensional vector. Noted that this setting cannot be considered when linear or generalized linear models are used. The reward functions of the three arms are $\mu_1(\boldsymbol{\theta}) = \theta_1 + \theta_2$, $\mu_2(\boldsymbol{\theta}) = {\theta_1}^2 - \theta_2 + 0.5$, and $\mu_3(\boldsymbol{\theta}) = 0.5\max\{\theta_1, \theta_2\} + 0.1$, which are presented in Figure 6.8.

In Figures 6.9, Figure 6.10, and Figure 6.11, we choose three different pairs of θ_1 and θ_2, so that arm 1, arm 2, and arm 3 are optimal in each case. The results demonstrate that the proposed algorithm scales well when one more parameter is added compared with the results in the first scenario, and GI-TS still outperforms all the benchmarks by reaching a higher learning speed compared with TSc and UCBc.

6.4.4 Transcoder Selection Problem

In this section, we focus on solving a decision-making problem of transcoder selection for crowdsourced live streaming platforms by GI-TS, which can be modelled as a structured bandit problem.

In crowdsourced live streaming platforms, the central controller needs to meet the massive transcoding demands of thousands of broadcasters. In order to provide sufficient computational resource, [24] proposed to take the advantage of edge computing by incentivizing end viewers' devices to be candidate transcoders. However,

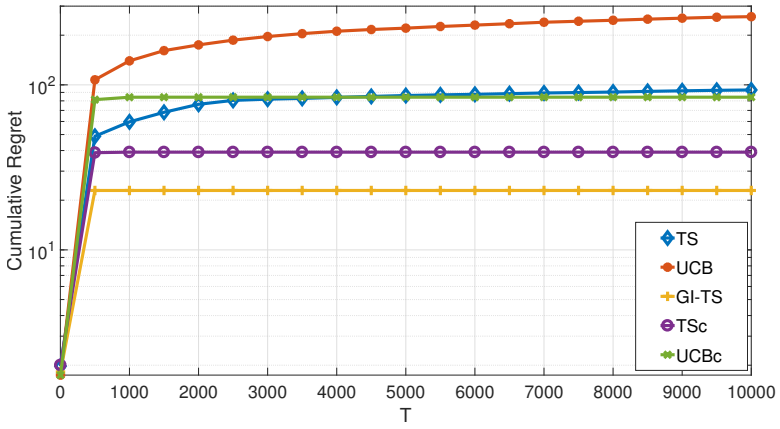

Figure 6.9: Cumulative regret when $\theta_1 = 0.8$ and $\theta_2 = 0.8$

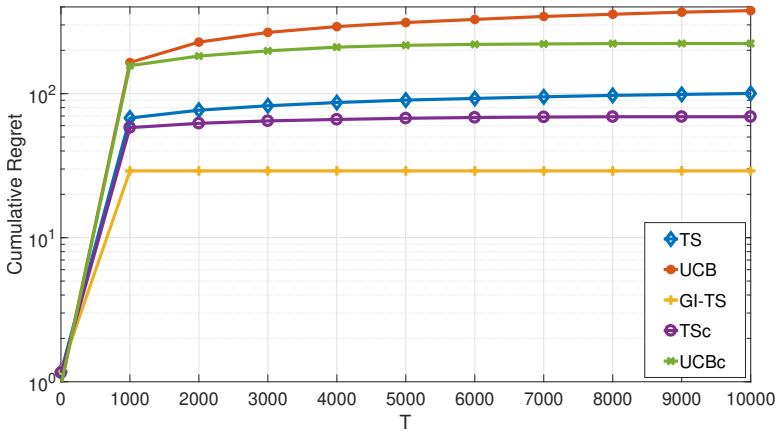

Figure 6.10: Cumulative regret when $\theta_1 = 0.8$ and $\theta_2 = 0.4$

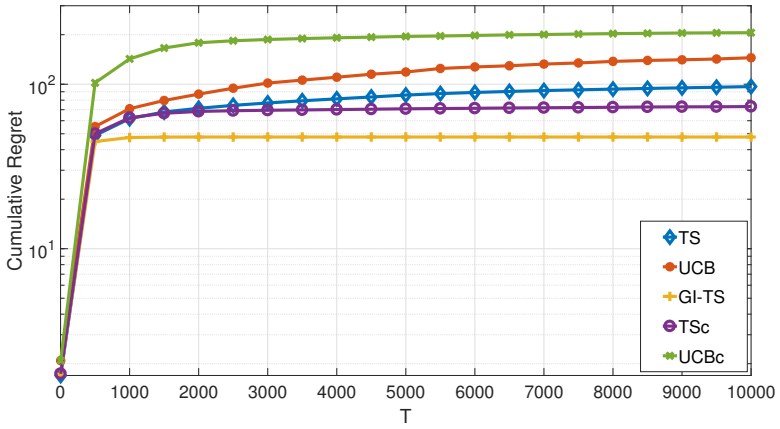

Figure 6.11: Cumulative regret when $\theta_1 = -0.9$ and $\theta_2 = -0.2$

the candidates can be offline during transcoding which can severely deteriorate the transcoding performance, so the stable devices are preferred to be selected. According to [160], the distribution of online durations of candidate transcoders follows the

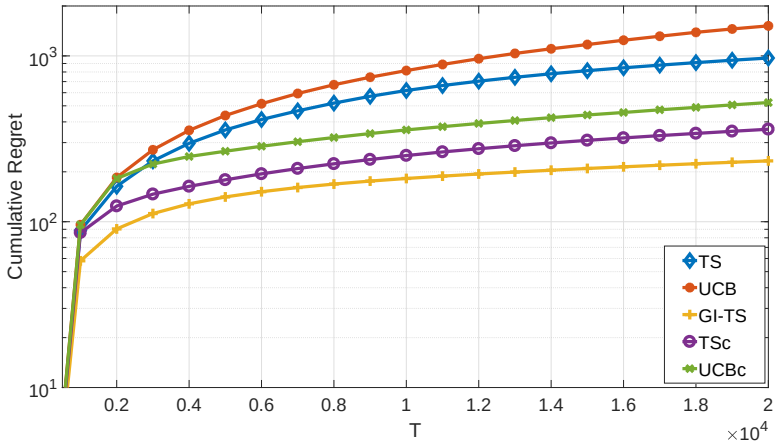

Figure 6.12: Cumulative regret when $\boldsymbol{\theta}^* = 22.5$

Pareto distribution, which implies that the longer a device is online, the more likely this device will continue to be online. The cumulative probability function of a Pareto random variable X with parameters β and x_m is

$$F_X(x) = \begin{cases} 1 - \left(\frac{x_m}{x}\right)^{\beta}. & x \geq x_m \\ 0, & x < x_m \end{cases} \tag{6.11}$$

For simplicity, assume $x \geq x_m$ always holds, define $S_f(x_m) = 1 - \left(\frac{x_m}{x_f}\right)^{\beta_f}$ as the transcoding stability of fog device f to show the probability that the transcoder will be online after the selection. In addition, x_f and β_f are predefined parameters which vary due to the heterogeneity of transcoders. Based on the setting, each fog device can be regarded as an arm, and the expected reward function of arm f can be written as $\mu_f = S_f(x_m)$. Moreover, the form of each reward function is known with $\boldsymbol{\theta}^* = x_m$, which is shared by all the arms. Therefore, this transcoder selection problem can be modelled as a structured bandit problem, which can be solved by the proposed GI-TS algorithm. We perform a simulation with 10 fog transcoders. The result is depicted in Figure 6.12. We can find that the proposed GI-TS algorithm outperforms all the benchmarks.

6.5 Conclusions

In this chapter, a new type of MAB problem called structured bandit is studied which assumes the arms are correlated by sharing the same unknown parameter in their reward functions. In order to maximize the cumulative reward in this new bandit problem, we design an algorithm which can estimate the true value of the unknown parameter with the knowledge of the reward functions and the historical observation of rewards. Simulation results demonstrate that the proposed GI-TS algorithm outperforms all the existing benchmarks and achieves the lowest cumulative learning regret in transcoder selection problem and other simulation scenarios. The GI-TS algorithm is robust and scales well when more arms are added or multi-dimensional unknown parameter is considered in the structured bandit problem.

Chapter 7

Conclusions

7.1 Summary of the Thesis

With the evolution of content demand characteristics and video sharing systems, Internet-based video sharing systems have gradually superseded the traditional television-based systems. Moreover, emerging video representations (QHD, 4K, 5K, etc.) have increased the bandwidth consumption per user request. Besides, the requirements of the ABR services for both on-demand video and live video streaming systems have further increased the burden of the bandwidth consumption. These challenges stress the importance of providing efficient caching and transcoding services. Moreover, new services such as crowdsourced live streaming, AR, and VR have come to the fore with even more stringent latency and computing requirement than traditional video streaming system. These services aim to provide the users with extensive personalized sensory experience and hologram depictions in real-time, and hence have to rely on edge caching and computing. In addition, the proliferation of social networks and social media is enabling the users to be content publisher thus disrupting the traditional server-client content delivery model. This trend makes the network traffic and user request to become highly dynamic.

In this thesis, we study RL to solve online decision-making problems in content caching and video transcoding systems at the network edge leveraging the context information. The main contributions and novelties are summarized as follows.

In Chapter 3, we study the caching replacement problems at an edge server,

assuming the unknown and time-varying content popularity profile. The caching decision problem is modelled as a non-stationary MDP with varying states and transition probabilities. In the formulated problem, since the immediate reward is a function of the file popularity which is unknown, a context-aware popularity learning algorithm is designed to learn the time-varying file popularities. Specifically, an incremental clustering scheme is leveraged to cluster requests into different groups according to their context information. This helps to extract the similarities among requests, which enhances the popularity learning rate and accuracy. With the learned knowledge, an RL-based content caching scheme is designed via SARSA and linear function approximation. A specific caching decision will be made based on the proposed scheme each time a request is received. Next, illuminated by the RL-based caching scheme, a reactive caching algorithm is proposed to reduce the computational complexity by directly comparing the popularities between the requested file and the cached files for cache replacement decision.

Both the theoretical analyses and the numerical results are performed to demonstrate the effectiveness of this work. The theoretical analysis proves that the popularity learning error achieved by the proposed popularity learning scheme only grows sublinearly with the increasing of requests, and the reactive caching scheme converges to the optimal caching scheme which is fed with true file popularity knowledge. Additionally, the computational complexities of both proposed algorithms are proved to be competitively low, which enhances the feasibilities of the algorithms for real-world applications. Extensive numerical results driven by different file requests models and varying system parameters have illustrated the superiority of the proposed algorithms. As compared to multiple benchmarks, the proposed algorithms can better track the variation of the time-varying popularity profiles and achieve robust caching performance.

In Chapter 4, an edge-assisted transcoding system is proposed for crowdsourced live streaming service, and a new QoE metric is defined which considers the influences from both the quality and the genre of the received live video. The transcoding task assignment and viewer association problem is formulated as a non-

convex integer optimization problem, which is then solved by the computationally attractive CGP. Trace-driven simulations demonstrate that the proposed CGP-based scheme reaches close performance as compared to the exhaustive search scheme and outperforms existing cloud-based transcoding schemes in terms of overall network utility.

In Chapter 5, a more practical edge transcoding system is proposed considering both the network delay and transcoding delay experienced at the UE, together with the network utility. The main purpose is to learn the transcoding capability which can be unknown in practical transcoding system. To identify the risk of choosing highly unstable transcoders while learning, given the risk-sensitivity of this formulated online decision-making problem, two risk-aware bandit algorithms are designed to balance the mean-variance tradeoff with refined UCB of the arms' variances. Based on the UCBs, a risk-aware contextual learning algorithm is designed to decide which transcoders are more stable and more efficient. In addition, an epoch-based transcoding task assignment and viewer association algorithm is proposed to maximize the network utility and maintain low switching costs of transcoding tasks.

Theoretical analyses of the proposed risk-aware bandit algorithms are presented which prove that both schemes achieve logarithmic learning regrets. Numerical results demonstrate that the proposed transcoding task assignment and viewer algorithm achieves a significant improvement in the network utility while reducing the switching cost as compared to the benchmarks.

In Chapter 6, a structured bandit problem is studied to solve the transcoding task assignment from a different perspective. Here, we assume that there are performance correlations among multiple transcoders but the context information which used to build the correlations is not available for some reason. In order to solve the structured bandit problem which assumes the arm rewards are functions of globally shared parameters, we first build a confidence set which is a set of parameters whose expected reward is close to the empirical mean rewards of each arm, and then a novel technique is designed to estimate the true value of the unknown pa-

rameter according to the built confidence set. With the estimation, an enhanced TS-based algorithm is designed to sequentially select edge transcoders while handling the exploration-exploitation dilemma. Comprehensive simulation results demonstrate that the proposed GI-TS scheme can achieve a noteworthy improvement of the learning regret compared with the existing benchmarks.

7.2 Future Work

At present, the video sharing system is still in the path of rapid development and thus the edge caching and computing still facing many challenges. We notice that efficient utilization of the context information to build temporal and spatial locality has the potential to improve existing video streaming systems. In addition, more practical system models which involve the system dynamics and the spatial-temporal correlations should be considered in the algorithm design. In this section, we provide recommendations for future work, based upon the current works in this thesis.

Above all, there are some immediate works. The proposed caching algorithms can be applied to the scenario where multiple cache-enabled edge servers are available, by exchanging the learned popularity knowledge for caching decision. Besides, it is worth studying the cooperation between edge servers, which has the potential to further unblock the bottleneck of popularity learning rate at the network edges. In the live video transcoding system, our work can be further extended to consider the asynchronous nature of the transcoding task arrivals, which can help to reduce the time complexity of large-scale optimization.

We also identify some related fundamental research topics in the long run:

First, the catalogue of the contents is also time-varying and hence should be taken into account when designing caching algorithms. In addition, since multiple versions of the same content are generated for varying user requests with different popularities, this difference can also be involved which we believe will provide finer granularity in the caching scheme design and improve the caching efficiency.

Second, in the edge-assisted transcoding systems, it is necessary to include the

massive broadcasters and transcoders in the optimization problem which, however, can be quite challenging to solve given the large scale of the problem. Hence, a less complex transcoding scheduling scheme is worth studying. Moreover, to cope with the stringent latency requirement of the emerging live AR and VR streaming systems, we suggest that through learning from historical traces of both the broadcaster and viewers, the service provider may predict the patterns and assign computational resources in a proactive way, which can improve the streaming process accordingly.

Finally, in edge caching and transcoding systems, we realize that the effects of the uncertainty and risk during online decision-making have not been well explored hence requires further investigation. Besides the risk-aware MAB, other safe RL-based schemes may also be worth exploring to solve the risk-aware decision-making problems in content caching and video crowdsourcing applications.

Appendix A

Mathematical Induction

In order to verify the transition from (3.17) to (3.18), we need to prove the following equation

$$\mathbb{E}\Big[\sum_{i=1}^{u}\left(A_i-\frac{{A_i}^2}{B}\right)\Big]=\sum_{i=1}^{u'}\Big[(A_i-\frac{{A_i}^2}{B})\sum_{i'=i}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\Big],$$
$$\leq\sum_{i=1}^{u'}(A_i-\frac{{A_i}^2}{B})\prod_{j=1}^{i}(1-\frac{A_j}{B}). \tag{A.1}$$

This can be proved via the mathematical induction. First, let us rewrite (A.1) and prove,

$$\sum_{i'=i}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\leq\prod_{j=1}^{i}(1-\frac{A_j}{B}). \tag{A.2}$$

For $i=1$, (A.2) is equal to

$$\sum_{i'=1}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\leq(1-\frac{A_1}{B}). \tag{A.3}$$

For $i=X$, we assume the following equation holds,

$$\sum_{i'=X}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\leq\prod_{j=1}^{X}(1-\frac{A_j}{B}). \tag{A.4}$$

Then, when $i=X+1$, we need to prove that

$$\sum_{i'=X+1}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]\leq\prod_{j=1}^{X+1}(1-\frac{A_j}{B}). \tag{A.5}$$

The left-hand side of (A.5) can be written as

$$\begin{aligned}&\textstyle\sum_{i'=X+1}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big],\\ =&\textstyle\sum_{i'=X}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]-\frac{A_{X+1}}{B}\prod_{j=1}^{X}(1-\frac{A_j}{B}).\end{aligned}\tag{A.6}$$

According to the assumption in (A.5) and (A.6), we have

$$\begin{aligned}\textstyle\sum_{i'=X+1}^{u'}\Big[\frac{A_{i'+1}}{B}\prod_{j=1}^{i'}(1-\frac{A_j}{B})\Big]&\leq\textstyle\prod_{j=1}^{X}(1-\frac{A_j}{B})-\frac{A_{X+1}}{B}\prod_{j=1}^{X}(1-\frac{A_j}{B}),\\ &=\textstyle\prod_{j=1}^{X}(1-\frac{A_j}{B})(1-\frac{A_{X+1}}{B}),\\ &=\textstyle\prod_{j=1}^{X+1}(1-\frac{A_j}{B}).\end{aligned}\tag{A.7}$$

Therefore, we complete the proof of the transition from (3.17) to (3.18).

Appendix B

Proof of Theorem 5.1

Lemma B.1. *(Lemma 2 of [118]) The learning regret of a policy π with respect to the approximated policy $\tilde{\pi}^*$ is bounded as*

$$\widetilde{Reg}_{\pi}^{T} \leq \sum_{a=1}^{K} \mathbb{E}\left[\tau_a\right]\left(\Delta_{a,a^*} + \Gamma_{a,a^*}^2\right) + \sigma_{a^*}^2,$$

where $a^* = \operatorname{argmin}_a \eta_a$, $\Delta_{a,a^*} = \eta_a - \eta_{a^*}$ and $\Gamma_{a,a^*} = \mu_a - \mu_{a^*}$.

Lemma B.2. *(Theorem 1 of [118]) With $\sigma_{\max}^2 = \max_{a\in\mathcal{K}} \sigma_a^2$, the learning regret difference can be bounded as*

$$Reg_{\pi}^{T} - \widetilde{Reg}_{\pi}^{T} \leq \sigma_{\max}^2 \left(\sum_{a\neq a^*}^{K} \frac{\Gamma_{a,a^*}^2}{\Delta_{a,a^*}} + 1 \right), \tag{B.1}$$

Lemma B.3. *The expected number of plays of any suboptimal arm $a \neq a^*$ in GRA-UCB can be upper bounded by*

$$\mathbb{E}\left[\tau_a^T\right] \leq \left(\frac{4\rho^2}{\Delta_{a,a^*}^2} + C \right) \log T + 3. \tag{B.2}$$

Proof: First, we upper bound τ_a^T as

$$\tau_a^T = \sum_{t=1}^{T} \left\{ I^t = a \right\} = l_a + \sum_{t=l_a+1}^{T} \left\{ I^t = a, \tau_a^t \geq l_a \right\} \tag{B.3}$$

where I^t represents the index of the arm played at round t, $\{A\}$ is an indicator

function which is equal one if A is true and zero otherwise, and l_a is an arbitrary positive integer. Thus,

$$\tau_a^T \leq l_a + \sum_{t=l_a+1}^{T} \{H_a^t - H_{a^*}^t \leq 0, \tau_a^t \geq l_a\}. \tag{B.4}$$

Based on (5.19), we can write $H_a^t - H_{a^*}^t$ as

$$\begin{aligned} H_a^t - H_{a^*}^t = (\hat{s}_{t,\tau_a^t}^2 - \sigma_a^2 - \rho(\bar{\mu}_a^t - \mu_a - c^t)) - (\hat{s}_{t,\tau_{a^*}^t}^2 - \rho(\bar{\mu}_{a^*}^t - \mu_{a^*} + c_*^t)) \\ + (\Delta_{a,a^*} - 2\rho c^t + \sigma_{a^*}^2), \end{aligned} \tag{B.5}$$

where $c^t = \sqrt{\frac{\log t}{\tau_a^t}}$, $c_*^t = \sqrt{\frac{\log t}{\tau_{a^*}^t}}$, and $\hat{s}_{t,\tau_a^t}^2 = \frac{(\tau_a^t - 1)(s_a^t)^2}{\chi_{1-\alpha,\tau_a^t-1}}$.

For $\tau_a^t \geq l_a = \left\lceil \frac{\left(4\rho^2 \log T\right)}{\Delta_{a,a^*}^2} \right\rceil$, the last term of (B.5) will be always positive since $\Delta_{a,a^*} - 2\rho c^t \geq 0$. Consequently, to ensure $H_a^t - H_{a^*}^t \leq 0$, the first term of (B.5) needs to be negative and the second term of (B.5) needs to be positive. Therefore, continuing from (B.4) and (B.5), we have

$$\begin{aligned} \tau_a^T < l_a + \sum_{t=l_a+1}^{T} \{\hat{s}_{t,\tau_a^t}^2 \leq \sigma_a^2\} + \sum_{t=l_a+1}^{T} \{\bar{\mu}_a^t \geq \mu_a + c^t\} \\ + \sum_{t=l_a+1}^{T} \{\bar{\mu}_{a^*}^t \leq \mu_{a^*} - c_*^t\}. \end{aligned} \tag{B.6}$$

Applying Facts 5.1 and 5.2, $\mathbb{E}[\tau_a^T]$ can be written as

$$\begin{aligned} \mathbb{E}[\tau_a^T] &\leq l_a + \sum_{t=l_a+1}^{T} \alpha + 2 \sum_{t=l_a+1}^{T} t^{-2} \\ &\leq \frac{4\rho^2 \log T}{\Delta_{a,a^*}^2} + 1 + \int_{l_a}^{T} \alpha dt + 2 \int_{l_a}^{T} t^{-2} dt. \end{aligned} \tag{B.7}$$

By setting $\alpha = \frac{C}{t}$, $C > 0$, $\mathbb{E}[\tau_a^T]$ is upped bounded as

$$\begin{aligned}\mathbb{E}[\tau_a^T] &\leq \frac{4\rho^2 \log T}{\Delta_{a,a^*}^2} + 1 + C\log T + 2l_a^{-1} \\ &\leq \left(\frac{4\rho^2}{\Delta_{a,a^*}^2} + C\right)\log T + 3, \end{aligned} \tag{B.8}$$

which completes the proof.

Combining Lemmas B.1, B.2, and B.3, the proof of Theorem 5.1 can be completed.

Appendix C

Proof of Lemma 5.1

The distribution presented in Fact 5.3 can be reformulated as a standard Gaussian distribution as

$$\frac{\sqrt{n}\left(s_n^2-\sigma^2\right)}{\sqrt{\mu_4-\sigma^4}} \to \mathcal{N}(0,1). \tag{C.1}$$

Therefore we have $\frac{n\left(s_n^2-\sigma^2\right)^2}{\mu_4-\sigma^4} \to \chi_1^2$. Consequently, the one-sided confidence interval is defined as

$$\Pr\{\frac{n\left(s_n^2-\sigma^2\right)^2}{\mu_4-\sigma^4} \leq \chi_{\alpha,1}^2\} = 1-\alpha. \tag{C.2}$$

Consequently, the $(1-\alpha)$ asymptotic confidence interval of the variance σ^2 is established as

$$\Pr\left\{v_n^{\text{lower}} \leq \sigma^2 \leq v_n^{\text{upper}}\right\} = 1-\alpha, \tag{C.3}$$

where $v_n^{\text{lower}} = \frac{ns_n^2-\sqrt{n\chi_{\alpha,1}^2\left(\mu_4-s_n^4\right)+\left(\chi_{\alpha,1}^2\right)^2\mu_4}}{n+\chi_{\alpha,1}^2}$. From (C.3), obviously we have $\Pr\left\{\sigma^2 \geq v_n^{\text{upper}}\right\} \leq \alpha$, which completes the proof.

Appendix D

Proof of Theorem 5.2

In ARA-UCB, similar to the analysis of GRA-UCB, we need to bound the number of plays for sub-optimal arms. The following lemma provides an upper bound on $\mathbb{E}\left[\tau_a^T\right]$.

Lemma D.1. *The expected number of plays of any suboptimal arm $a \neq a^*$ in ARA-UCB for sub-Gaussian rewards can be upper bounded by*

$$\mathbb{E}\left[\tau_a^T\right] \leq \left(\frac{4\rho^2}{\Delta_{a,a^*}^2} + C\right)\log T + \beta \log L_a + 3. \tag{D.1}$$

Proof: According to (5.23) and (B.4), we have

$$\begin{aligned} N_a^t - N_{a^*}^t = (v'_{\tau_a^t} - \sigma_a^2 - \rho(\bar{\mu}_a^t - \mu_a - c^t)) - (v'_{\tau_{a^*}^t} - \rho(\bar{\mu}_{a^*}^t - \mu_{a^*} + c_*^t)) \\ + (\Delta_{a,a^*} - 2\rho c^t + \sigma_{a^*}^2). \end{aligned} \tag{D.2}$$

For $\tau_a^t \geq l_a = \left\lceil \frac{\left(4\rho^2 \log T\right)}{\Delta_{a,a^*}^2} \right\rceil$, $\Delta_{a,a^*} - 2\rho c^t \geq 0$, and hence, the last term of (D.2) will always be positive.

To ensure $N_a^t - N_{a^*}^t \leq 0$, we need the first term of (D.2) to be negative and the second term of (D.2) to be positive. Consequently, continuing from (D.2), we have

$$\begin{aligned} \tau_a^T \leq l_a + \sum_{t=l_a+1}^{T} \{v'_{\tau_a^t} \leq \sigma_a^2\} + \sum_{t=l_a+1}^{T} \{\bar{\mu}_a^t \geq \mu_a + c^t\} \\ + \sum_{t=l_a+1}^{T} \{\bar{\mu}_{a^*}^t \leq \mu_{a^*} - c_*^t\}. \end{aligned} \tag{D.3}$$

According to Lemma 5.1, we can conclude that

$$\Pr\{v'_{\tau_a^t} \leq \sigma_a^2\} \leq \beta\alpha, \tag{D.4}$$

where $\beta > 1$ is a scalar. Given a sufficiently large number of samples (L_a) of arm a, Lemma 5.1 holds when $\beta = 1$. However, when $\tau_a^t < L_a$ which means τ_a^t is not sufficiently large, the actual confidence bound in (C.3) is smaller than the expected bound. Since the reward is sub-Gaussian, β cannot go to infinity and will decrease when more samples are collected.

Applying Fact 5.1 and (D.4), by setting $\alpha = Ct^{-1}$ with $C > 0$, the expected value of τ_a^T in (D.3) can be written as

$$\begin{aligned}
\mathbb{E}\left[\tau_a^T\right] &\leq l_a + \sum_{t=l_a+1}^{L_a} (\beta-1)\alpha + \sum_{t=l_a+1}^{T} \alpha + 2\sum_{t=l_a+1}^{T} t^{-2}, \\
&\leq \left\lceil \frac{4\rho^2 \log T}{\Delta_{a,i^*}^2} \right\rceil + \int_{l_a}^{L_a} \beta \frac{C}{t} dt + \int_{l_a}^{T} \beta \frac{C}{t} dt + 2\int_{l_a}^{T} t^{-2} dt, \\
&\leq \left(\frac{4\rho^2}{\Delta_{i,i^*}^2} + C \right) \log T + \beta \log L_a + 3,
\end{aligned} \tag{D.5}$$

which completes the proof.

Combining Lemmas B.1, B.2, and D.1, the proof of Theorem 5.2 can be completed.

Bibliography

[1] S. Li, J. Xu, M. van der Schaar, and W. Li. Popularity-driven content caching. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016.

[2] S. Li, J. Xu, M. van der Schaar, and W. Li. Trend-aware video caching through online learning. *IEEE Transactions on Multimedia*, 18(12):2503–2516, 2016.

[3] 2020 global networking trends report:. Technical report, Cisco, 2020.

[4] L. Tang, Q. Huang, A. Puntambekar, Y. Vigfusson, W. Lloyd, and K. Li. Popularity prediction of facebook videos for higher quality streaming. In *USENIX Annual Technical Conference*, pages 111–123, Santa Clara, 2017.

[5] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang. Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 20(1):7–38, 2018.

[6] A. Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1 – 32, 2012.

[7] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys Tutorials*, 16(2):1024–1049, 2014.

[8] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[9] D. Perino and M. Varvello. A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, pages 44–49, Toronto, 2011.

[10] V. A. Siris, X. Vasilakos, and G. C. Polyzos. Efficient proactive caching for supporting seamless mobility. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 2014.

[11] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire. The role of caching in future communication systems and networks. *IEEE Journal on Selected Areas in Communications*, 36(6):1111–1125, 2018.

[12] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung. Cache in the air: exploiting content caching and delivery techniques for 5g systems. *IEEE Communications Magazine*, 52(2):131–139, 2014.

[13] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang. Cooperative content caching in 5g networks with mobile edge computing. *IEEE Wireless Communications*, 25(3):80–87, 2018.

[14] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications*, volume 3, pages 1587–1596, Anchorage, 2001.

[15] T. Kelly and D. Reeves. Optimal web cache sizing: scalable methods for exact solutions. *Computer Communications*, 24(2):163–173, 2001.

[16] A. Tatar, M.D. De Amorim, S. Fdida, and P. Antoniadis. A survey on predicting the popularity of web content. *Journal of Internet Services and Applications*, 5(1):8, 2014.

[17] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili. Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. In *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Jackson, 2017.

[18] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu. Understanding performance of edge content caching for mobile video streaming. *IEEE Journal on Selected Areas in Communications*, 35(5):1076–1089, 2017.

[19] C. Zhang and J. Liu. On crowdsourced interactive live streaming: A twitch.tv-based measurement study. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, page 55–60, Portland, 2015.

[20] K. Bilal and A. Erbad. Impact of multiple video representations in live streaming: A cost, bandwidth, and qoe analysis. In *2017 IEEE International Conference on Cloud Engineering (IC2E)*, pages 88–94, 2017.

[21] Mansoor Iqbal. Twitch revenue and usage statistics (2020). `https://www.businessofapps.com/data/twitch-statistics/`.

[22] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the 6th ACM Multimedia Systems Conference*, page 49–60, Portland, 2015.

[23] K. Bilal, O. Khalid, A. Erbad, and U.K. Samee. Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. *Computer Networks*, 130:94 – 120, 2018.

[24] Q. He, C. Zhang, X. Ma, and J. Liu. Fog-based transcoding for crowdsourced video livecast. *IEEE Communications Magazine*, 55(4):28–33, 2017.

[25] Q. He, C. Zhang, and J. Liu. Crowdtranscoding: Online video transcoding with massive viewers. *IEEE Transactions on Multimedia*, 19(6):1365–1375, 2017.

[26] Y. Zhu, Q. He, J. Liu, B. Li, and Y. Hu. When crowd meets big video data: Cloud-edge collaborative transcoding for personal livecast. *IEEE Transactions on Network Science and Engineering*, 7(1):42–53, 2020.

[27] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

[28] Y. Shi, J. Zhang, B. O'Donoghue, and K. B. Letaief. Large-scale convex optimization for dense wireless cooperative networks. *IEEE Transactions on Signal Processing*, 63(18):4729–4743, 2015.

[29] Z. Tian, J. Wang, J. Wang, and J. Song. Distributed noma-based multi-armed bandit approach for channel access in cognitive radio networks. *IEEE Wireless Communications Letters*, 8(4):1112–1115, 2019.

[30] N. Gulati and K. R. Dandekar. Learning state selection for reconfigurable antennas: A multi-armed bandit approach. *IEEE Transactions on Antennas and Propagation*, 62(3):1027–1038, 2014.

[31] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.

[32] Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz, and G. Y. Grabarnik. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1569–1580, 2019.

[33] S. Henri, C. Vlachou, and P. Thiran. Multi-armed bandit in action: Optimizing performance in dynamic hybrid networks. *IEEE/ACM Transactions on Networking*, 26(4):1879–1892, 2018.

[34] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[35] W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[36] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[37] X. Wang, S. Hoi, C.H. Steven, C. Liu, and M. Ester. Interactive social recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 357–366, Singapore, 2017.

[38] S. Asur and B. A. Huberman. Predicting the future with social media. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 492–499, 2010.

[39] P. Blasco and D. Gündüz. Multi-access communications with energy harvesting: A multi-armed bandit model and the optimality of the myopic policy. *IEEE Journal on Selected Areas in Communications*, 33(3):585–597, 2015.

[40] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis. Optimal and scalable caching for 5g using reinforcement learning of space-time popularities. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):180–190, 2018.

[41] E. Baştuğ, M. Bennis, and M. Debbah. A transfer learning approach for cache-enabled wireless networks. In *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 161–166, 2015.

[42] Y. Zhu, J. Liu, Z. Wang, and C. Zhang. Cloud meets uncertain crowd: An auction approach for crowdsourced livecast transcoding. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1372–1380, Mountain View, 2017.

[43] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, page 13–16, Helsinki, 2012.

[44] K. Poularakis and L. Tassiulas. Cooperation and information replication in wireless networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2062):20150123, 2016.

[45] I. Baev, R. Rajaraman, and C. Swamy. Approximation algorithms for data placement problems. *SIAM Journal on Computing*, 38(4):1411–1429, 2008.

[46] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[47] K. Poularakis and L. Tassiulas. Optimal cooperative content placement algorithms in hierarchical cache topologies. In *2012 46th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2012.

[48] A. Ioannou and S. Weber. A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Communications Surveys Tutorials*, 18(4):2847–2886, 2016.

[49] X. Peng, J. Shen, J. Zhang, and K. B. Letaief. Backhaul-aware caching placement for wireless networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.

[50] R. Wang, X. Peng, J. Zhang, and K. B. Letaief. Mobility-aware caching for content-centric wireless networks: modeling and methodology. *IEEE Communications Magazine*, 54(8):77–83, 2016.

[51] 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*, 34(5):1222–1234, 2016.

[52] Zheng Chen, Jemin Lee, Tony QS Quek, and Marios Kountouris. Cooperative caching and transmission design in cluster-centric small cell networks. *IEEE Transactions on Wireless Communications*, 16(5):3401–3415, 2017.

[53] D. Malak, M. Al-Shalash, and J. G. Andrews. Optimizing content caching to maximize the density of successful receptions in device-to-device networking. *IEEE Transactions on Communications*, 64(10):4365–4380, 2016.

[54] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief. Mobility-aware caching in d2d networks. *IEEE Transactions on Wireless Communications*, 16(8):5001–5015, 2017.

[55] C. Zhan and Z. Wen. Content cache placement for scalable video in heterogeneous wireless network. *IEEE Communications Letters*, 21(12):2714–2717, 2017.

[56] S. Ioannidis and E. Yeh. Jointly optimal routing and caching for arbitrary network topologies. *IEEE Journal on Selected Areas in Communications*, 36(6):1258–1275, 2018.

[57] P. Cheng, C. Ma, M. Ding, Y. Hu, Z. Lin, Y. Li, and B. Vucetic. Localized small cell caching: A machine learning approach based on rating data. *IEEE Transactions on Communications*, 67(2):1663–1676, 2019.

[58] Kuikui Li, Chenchen Yang, Zhiyong Chen, and Meixia Tao. Optimization and analysis of probabilistic caching in *n*-tier heterogeneous networks. *IEEE Transactions on Wireless Communications*, 17(2):1283–1297, 2018.

[59] Xianzhe Xu and Meixia Tao. Analysis and optimization of probabilistic caching in multi-antenna small-cell networks. In *IEEE GLOBECOM*, Singapore, 2017.

[60] Z. Chen, N. Pappas, and M. Kountouris. Probabilistic caching in wireless d2d networks: Cache hit optimal versus throughput optimal. *IEEE Communications Letters*, 21(3):584–587, 2017.

[61] M. A. Maddah-Ali and U. Niesen. Fundamental limits of caching. *IEEE Transactions on Information Theory*, 60(5):2856–2867, 2014.

[62] Xuejian Xu and Meixia Tao. Modeling, analysis, and optimization of coded caching in small-cell networks. *IEEE Transactions on Communications*, 65(8):3415–3428, 2017.

[63] Jinbei Zhang, Xiaojun Lin, and Xinbing Wang. Coded caching under arbitrary popularity distributions. *IEEE Transactions on Information Theory*, 64(1):349–366, 2018.

[64] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[65] D.J.C. MacKay. Fountain codes. *IEE Proceedings - Communications*, 152:1062–1068(6), 2005.

[66] M. Ji, G. Caire, and A. F. Molisch. Fundamental limits of caching in wireless d2d networks. *IEEE Transactions on Information Theory*, 62(2):849–869, 2016.

[67] M. Mohammadi Amiri and D. Gündüz. Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff. *IEEE Transactions on Communications*, 65(2):806–815, 2017.

[68] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi. Learn to cache: Machine learning for network edge caching in the big data era. *IEEE Wireless Communications*, 25(3):28–35, 2018.

[69] B. N. Bharath, K. G. Nagananda, and H. V. Poor. A learning-based approach to caching in heterogenous small cell networks. *IEEE Transactions on Communications*, 64(4):1674–1686, 2016.

[70] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang. Learning-based content caching and sharing for wireless networks. *IEEE Transactions on Communications*, 65(10):4309–4324, 2017.

[71] SM Shahrear Tanzil, William Hoiles, and Vikram Krishnamurthy. Adaptive scheme for caching youtube content in a cellular network: Machine learning approach. *IEEE Access*, 5:5870–5881, 2017.

[72] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas. A deep learning approach for optimizing content delivering in cache-enabled hetnet. In *2017 International Symposium on Wireless Communication Systems (ISWCS)*, pages 449–453, 2017.

[73] J. Dilley and M. Arlitt. Improving proxy cache performance: analysis of three replacement policies. *IEEE Internet Computing*, 3(6):44–50, 1999.

[74] M. Bilal and S. Kang. A cache management scheme for efficient content eviction and replication in cache networks. *IEEE Access*, 5:1692–1701, 2017.

[75] Jie Xu, Mihaela van der Schaar, Jiangchuan Liu, and Haitao Li. Forecasting popularity of videos using social media. *IEEE Journal of Selected Topics in Signal Processing*, 9(2):330–343, 2015.

[76] X. Zhao, P. Yuan, H. li, and S. Tang. Collaborative edge caching in context-aware device-to-device networks. *IEEE Transactions on Vehicular Technology*, 67(10):9583–9596, 2018.

[77] Kaiyang Guo, Chenyang Yang, and Tingting Liu. Caching in base station with recommendation via q-learning. In *IEEE WCNC*, San Francisco, 2017.

[78] S. O. Somuyiwa, A. György, and D. Gündüz. A reinforcement-learning approach to proactive caching in wireless networks. *IEEE Journal on Selected Areas in Communications*, 36(6):1331–1344, 2018.

[79] L. Dong, D. Niyato, D. I. Kim, and D. T. Hoang. A joint scheduling and content caching scheme for energy harvesting access points with multicast. In *IEEE GLOBECOM*, Singapore, 2017.

[80] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao. Multi-agent reinforcement learning for efficient content caching in mobile d2d networks. *IEEE Transactions on Wireless Communications*, 18(3):1610–1622, 2019.

[81] L. Li, Y. Xu, J. Yin, W. Liang, X. Li, W. Chen, and Z. Han. Deep reinforcement learning approaches for content caching in cache-enabled d2d networks. *IEEE Internet of Things Journal*, 7(1):544–557, 2020.

[82] S. Müller, O. Atan, M. van der Schaar, and A. Klein. Smart caching in wireless small cell networks via contextual multi-armed bandits. In *2016 IEEE International Conference on Communications (ICC)*, 2016.

[83] 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*, 16(2):1024–1036, 2017.

[84] H. Gerhard, N. Konstantinos, H. Frank, and H. Oliver. Performance evaluation for new web caching strategies combining lru with score based object selection. *Computer Networks*, 125:172–186, 2017.

[85] E.J. Coffman and P.J. Denning. *Operating systems theory*, volume 973. 1973.

[86] B. N. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor. Caching with time-varying popularity profiles: A learning-theoretic perspective. *IEEE Transactions on Communications*, 66(9):3837–3847, 2018.

[87] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings.*, volume 1, pages 126–134, New York, 1999.

[88] A. Dabirmoghaddam, M.M. Barijough, and J.J. Garcia-Luna-Aceves. Understanding optimal caching and opportunistic caching at "the edge" of information-centric networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, page 47–56, Paris, 2014.

[89] H. Kim, J. Park, M. Bennis, S. Kim, and M. Debbah. Ultra-dense edge caching under spatio-temporal demand and network dynamics. In *IEEE International Conference on Communications (ICC)*, Paris, 2017.

[90] F. Mériaux, S. Lasaulce, and H. Tembine. Stochastic differential games and energy-efficient power control. *Dynamic Games and Applications*, 3(1):3–23, 2013.

[91] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. Temporal locality in today's content caching: Why it matters and how to model it. *SIGCOMM Comput. Commun. Rev.*, 43(5):5–12, 2013.

[92] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas. Placing dynamic content in caches with small population. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, San Francisco, 2016.

[93] M. Garetto, E. Leonardi, and S. Traverso. Efficient analysis of caching strategies under dynamic content popularity. In *IEEE INFOCOM*, pages 2263–2271, 2015.

[94] J. Yan, Y. Jiang, F. Zheng, F. R. Yu, X. Gao, and X. You. Distributed edge caching with content recommendation in fog-rans via deep reinforcement learning. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, Dublin, 2020.

[95] Qiyun He, Jiangchuan Liu, Chonggang Wang, and Bo Li. Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach. *IEEE Transactions on Multimedia*, 18(5):916–928, 2016.

[96] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen, and G. Zhang. Online cloud transcoding and distribution for crowdsourced live game video stream-

ing. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(8):1777–1789, 2017.

[97] K. Bilal, A. Erbad, and M. Hefeeda. Crowdsourced multi-view live video streaming using cloud computing. *IEEE Access*, 5:12635–12647, 2017.

[98] C. Zhang, J. Liu, and H. Wang. Cloud-assisted crowdsourced livecast. *ACM Transactions on Multimedia Comput. Commun. Appl.*, 13(3s):46:1–46:22, 2017.

[99] G. Gao, Y. Wen, and C. Westphal. Dynamic priority-based resource provisioning for video transcoding with heterogeneous qos. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5):1515–1529, 2019.

[100] C. Dong, Y. Jia, H. Peng, X. Yang, and W. Wen. A novel distribution service policy for crowdsourced live streaming in cloud platform. *IEEE Transactions on Network and Service Management*, 15(2):679–692, 2018.

[101] W.Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[102] F. Wang, C. Zhang, F. wang, J. Liu, Y. Zhu, H. Pang, and L. Sun. Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 910–918, Paris, 2019.

[103] K. Bilal, E. Baccour, A. Erbad, A. Mohamed, and M. Guizani. Collaborative joint caching and transcoding in mobile edge networks. *Journal of Network and Computer Applications*, 136:86–99, 2019.

[104] Z. Zhang, R. Wang, F. R. Yu, F. Fu, and Q. Yan. Qos aware transcoding for live streaming in edge-clouds aided hetnets: An enhanced actor-critic approach. *IEEE Transactions on Vehicular Technology*, 68(11):11295–11308, 2019.

[105] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia. A survey of general-purpose crowdsourcing techniques. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2246–2266, 2016.

[106] J. Ren, Y. Zhang, K. Zhang, and X. Shen. Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Communications Magazine*, 53(3):98–105, 2015.

[107] X. Tao and W. Song. Location-dependent task allocation for mobile crowdsensing with clustering effect. *IEEE Internet of Things Journal*, 6(1):1029–1045, 2019.

[108] K. Han, C. Zhang, and J. Luo. Taming the uncertainty: Budget limited robust crowdsensing through online learning. *IEEE/ACM Transactions on Networking (TON)*, 24(3):1462–1475, 2016.

[109] U. U. Hassan and E. Curry. A multi-armed bandit approach to online spatial task assignment. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 212–219, 2014.

[110] P. Yang, N. Zhang, S. Zhang, K. Yang, L. Yu, and X. Shen. Identifying the most valuable workers in fog-assisted spatial crowdsourcing. *IEEE Internet of Things Journal*, 4(5):1193–1203, 2017.

[111] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[112] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24, pages 2249–2257, 2011.

[113] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. *A Tutorial on Thompson Sampling*, volume 11. 2018.

[114] H. Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[115] A. Sani, A. Lazaric, and R. Munos. Risk-aversion in multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 3275–3283, 2012.

[116] S. Vakili and Q. Zhao. Mean-variance and value at risk in multi-armed bandit problems. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1330–1335, 2015.

[117] S. Vakili and Q. Zhao. Risk-averse online learning under mean-variance measures. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1911–1915, 2015.

[118] S. Vakili and Q. Zhao. Risk-averse multi-armed bandit problems under mean-variance measure. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1093–1111, 2016.

[119] N. Galichet, M. Sebag, and O. Teytaud. Exploration vs exploitation vs safety: Risk-aware multi-armed bandits. In *Asian Conference on Machine Learning*, pages 245–260, 2013.

[120] Y. David, B. Szörényi, M. Ghavamzadeh, S. Mannor, and N. Shimkin. Pac bandits with risk constraints. In *ISAIM*, 2018.

[121] L.A. Prashanth, K. Jagannathan, and R.K. Kolla. Concentration bounds for cvar estimation: The cases of light-tailed and heavy-tailed distributions. *arXiv preprint arXiv:1901.00997*, 2019.

[122] A. Kagrecha, J. Nair, and K. Jagannathan. Distribution oblivious, risk-aware algorithms for multi-armed bandits with unbounded rewards. In *Advances in Neural Information Processing Systems 32*, pages 11272–11281. 2019.

[123] E. Even-Dar, M. Kearns, and J. Wortman. Risk-sensitive online learning. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2006.

[124] A. Yekkehkhany, E. Arian, M. Hajiesmaili, and R. Nagi. Risk-averse explore-then-commit algorithms for finite-time bandits. *arXiv preprint arXiv:1904.13387*, 2019.

[125] T. Lattimore and R. Munos. Bounded regret for finite-armed structured bandits. In *Advances in Neural Information Processing Systems 27*, pages 550–558. 2014.

[126] L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, Raleigh, 2010.

[127] A. Shipra and G. Navin. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 127–135, 2013.

[128] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 1015–1022, Haifa, 2010.

[129] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 539–548, Pisa, 2016.

[130] S. Gupta, G. Joshi, and O. Yağan. Correlated multi-armed bandits with a latent random source. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3572–3576, 2020.

[131] J. Huang. Demand functions in decision modeling: A comprehensive survey and research directions. *Decision Sciences*, 44(3):557–609, 2013.

[132] O. Atan, C. Tekin, and M. van der Schaar. Global bandits. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):5798–5811, 2018.

[133] C. Shen, R. Zhou, C. Tekin, and M. van der Schaar. Generalized global bandit and its application in cellular coverage optimization. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):218–232, 2018.

[134] Z. Wang, R. Zhou, and C. Shen. Regional multi-armed bandits with partial informativeness. *IEEE Transactions on Signal Processing*, 66(21):5705–5717, 2018.

[135] S. Gupta, S. Chaudhari, S. Mukherjee, G. Joshi, and O. Yağan. A unified approach to translate classical bandit algorithms to the structured bandit setting. *arXiv preprint arXiv:1810.08164*, 2018.

[136] Xingchi Liu, Mahsa Derakhshani, and Sangarapillai Lambotharan. Contextual learning for content caching with unknown time-varying popularity profiles via incremental clustering. *IEEE Transactions on Communications*, 69(5):3011–3024, 2021.

[137] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[138] J. Yu and S. Mannor. Online learning in markov decision processes with arbitrarily changing rewards and transitions. In *2009 International Conference on Game Theory for Networks*, pages 314–322, Istanbul, 2009.

[139] E. Liberty, R. Sriharsha, and M. Sviridenko. An algorithm for online k-means clustering. pages 81–89. SIAM, 2016.

[140] G.A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[141] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Approximate reinforcement learning: An overview. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Paris, 2011.

[142] N. K. Jong and P. Stone. Model-based function approximation in reinforcement learning. In *ACM International Joint Conference on Autonomous Agents and Multiagent Systems*, Honolulu, 2007.

[143] F. S. Melo, S. P. Meyn., and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *ACM International Conference on Machine Learning*, Helsinki, 2008.

[144] A. Slivkins. Contextual bandits with similarity information. *Journal of Machine Learning Research*, 15:2533–2568, 2014.

[145] J. Gao, S. Zhang, L. Zhao, and X. S. Shen. The design of dynamic probabilistic caching with time-varying content popularity. *IEEE Transactions on Mobile Computing*, 2020.

[146] X. Liu, M. Derakhshani, and S. Lambotharan. Joint transcoding task assignment and association control for fog-assisted crowdsourced live streaming. *IEEE Communications Letters*, 23(11):2036–2040, 2019.

[147] M. Derakhshani, X. Wang, D. Tweed, T. Le-Ngoc, and A. Leon-Garcia. Apsta association control for throughput maximization in virtualized wifi networks. *IEEE Access*, 6:45034–45050, 2018.

[148] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, March 2014.

[149] Codonbyte. Codonbyte's pastebin. `https://pastebin.com/u/Codonbyte`.

[150] Broadcasting guidelines. `https://stream.twitch.tv/encoding/`.

[151] X. Liu, M. Derakhshani, S. Lambotharan, and M. van der Schaar. Risk-aware multi-armed bandits with refined upper confidence bounds. *IEEE Signal Processing Letters*, 28:269–273, 2021.

[152] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[153] D. C. Montgomery and G. C. Runger. *Applied statistics and probability for engineers*. John Wiley and Sons, 2014.

[154] A. DasGupta. *Asymptotic theory of statistics and probability*. Springer Science & Business Media, 2008.

[155] Arthur E. Hoerl, Robert W. Kannard, and Kent F. Baldwin. Ridge regression:some simulations. *Communications in Statistics*, 4(2):105–123, 1975.

[156] W. Chu, L. Li, L. Reyzin, and S. Robert. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pages 208–214, 2011.

[157] T. Le, C. Szepesvári, and R. Zheng. Sequential learning for multi-channel wireless network monitoring with channel switching costs. *IEEE Transactions on Signal Processing*, 62(22):5919–5929, 2014.

[158] X. Liu, M. Derakhshani, Z. Zhu, and S. Lambotharan. Globally informative thompson sampling for structured bandit problems with application to crowdtranscoding. In *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 210–215, 2021.

[159] S. Gupta, G. Joshi, and O. Yağan. Exploiting correlation in finite-armed structured bandits. *arXiv preprint arXiv:1810.08164*, 2018.

[160] Q. He, C. Zhang, and J. Liu. Crowdtranscoding: Online video transcoding with massive viewers. *IEEE Transactions on Multimedia*, 19(6):1365–1375, 2017.