Mobility-Prediction based Proactive Edge Caching in Vehicular Networks

Qiao Wang

PhD

University of York

Electronic Engineering

July 2022

Abstract

This thesis studies and designs machine intelligence based mobility-prediction algorithms to address the proactive edge caching problem in vehicular networks. In particular, the thesis focuses on predicting the next road-side unit (RSU) along a vehicle's route as the node for proactively caching content and meanwhile investigates approaches to improve such prediction accuracy.

Firstly, the thesis presents an offline sequence prediction based proactive caching (SPPC) system that employs Compact Prediction Tree+ (CPT+) algorithm by modelling RSUs as symbols of sequences. The proposed system explores the feasibility and the performance of applying the next RSU prediction to proactive caching. Moreover, to achieve better adaptability of learning approaches, the thesis then proposes an online bandit learning approach by designing two novel multi-armed bandit (MAB) based proactive caching systems. They are proven to have better prediction accuracy than other comparative systems, and hence better network performance in terms of average network delay. In addition, the MAB-based learning systems are also evaluated with an extended uncertainty analysis framework, Subjective Logic, using entropy, and they demonstrate improved uncertainty reduction during learning, which shows analytical evidence of their better prediction accuracy. Furthermore, with the aim of fully exploring the potential of cMAB learning, the thesis proposes a Hybrid cMAB Proactive Caching (HCPC) system which implements Dual-context cMAB and Single-context cMAB algorithms and is further developed into two system variants: Vehicle-Centric and RSU-Centric. They allow RSUs to adaptively finalise their predictions by a specific switching mechanism and improve the prediction accuracy to a new level.

Lastly, to verify the applicability and adaptability of the proposed algorithms and systems in this thesis, traffic simulation with Simulator of Urban MObility (SUMO) in two major cities, Las Vegas, USA and Manchester, UK, with different road layouts, is performed and various traffic scenarios are tested and assessed.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I still remember the interview I had with my supervisor, Prof. David Grace, four years ago. That was the very start of my PhD research journey. It was a wonderful interview and more importantly, that was when I gained the confidence to commence my research career because Prof. Grace said to me that I was research material from his long supervision experience. So, I took his compliment and encouragement and decided to join his research group as soon as I received the offer. Therefore, first and foremost, I would like to express my deepest gratitude to my supervisor for his support for my PhD application, for his professional and careful guidance throughout my entire study, and for his trust and encouragement all the time. I have also been deeply influenced by his research enthusiasm and rigorous working attitude. Needless to say, I am extremely lucky to have such a supportive supervisor.

I would also like to thank my thesis advisor, Prof. Alister G. Burr for the motivating in-depth discussions, valuable comments, and suggestions during the periodic research assessments that helped in shaping my research work. I have benefited a lot from the discussions and insightful conversations with him.

I am also thankful to the members of the Communication Technologies Research Group and the Department of Electronic Engineering for creating a wonderful academic atmosphere. In particular, my sincere thanks to all my previous and current colleagues in the research group for their countless discussions and sharing and valuable advice. Thanks to Dr. Hamed Ahmadi for giving me the opportunity to join the MACBOOK project so that I have enhanced my knowledge in relevant subjects. Thanks to my postgraduate supervisor, Dr. Dmitri Moltchanov, who was so supportive of my PhD applications at the very beginning.

Finally, I would like to express my gratitude from the bottom of my heart to my dear family. Thanks to my parents and my sister who have been supportive of all my decisions throughout my life. I am deeply grateful to my wife, Hanna, for many years of support, company, and dedication. Thanks to my precious daughters, Emily and Joanna, for the happiness and joy they have brought to my life.

Declaration

All work presented in this thesis is original to the best knowledge of the author. This work has not previously been presented for an award at this or any other institution. All sources are acknowledged as References. The research presented in this thesis features in a number of the author's publications listed below.

Journal Articles

- Q. Wang and D. Grace, "Proactive Edge Caching in Vehicular Networks: An Online Bandit Learning Approach", *accepted by IEEE Access for publication*, available as preprint [6] on *TechRxiv*
- Q. Wang and D. Grace, "A Hybrid Proactive Caching System in Vehicular Networks based on Contextual Multi-armed Bandit Learning", *Manuscript under review in IEEE Transactions on Vehicular Technology*, available as preprint [7] on *TechRxiv*

Conference Paper

- Q. Wang and D. Grace, "Sequence prediction-based proactive caching in vehicular content networks," in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*, 2020, pp. 1–6

Qiao Wang

July 2022

Chapter 1

Introduction

1.1 Overview

The past decades have witnessed rapid growth of the automobile industry and its economic and societal influence continues to expand. The industry has been making road vehicles more and more intelligent over the past decade, thanks to the developments in electronics and communication technologies. Vehicles, embedded with onboard units (OBUs), are able to communicate with road infrastructures e.g., roadside units (RSUs), and even with other vehicles. These form a large communication network, i.e., vehicular network [9, 10]. It is considered to be one of the most important enabling technologies of the next-generation intelligent transportation system [9]. In addition, the upcoming era of autonomous driving means that vehicles not only will act as a simple means of transportation but also become moving entertainment centres where drivers and passengers are able to entertain themselves while travelling in the car [10, 11].

However, such a revolution also poses unprecedented challenges to conventional vehicular networks from the perspective of content transmission. Currently, tremendous data demands from vehicular users are satisfied by the remote content provider through network infrastructure such as RSUs. This inevitably causes problems such as high network latency and poor quality of experience (QoE) for the users, given the limitation of link capacity and bandwidth resources [12]. In addition to this, as fast-moving objects, vehicles may experience frequent intermittent connections with RSUs, which results in a rapidly changing vehicular environment. High-speed mobility causes frequent link re-connections and fast fading of vehicle-to-RSU channels, which means that a content transmission between a vehicular user and an RSU may not be completed within the coverage of the RSU and the user has to re-request the remaining content after reconnecting to a new RSU at a dramatically reduced data rate [11]. This is another cause for the downgraded QoE.

Recently, thanks to the development of mobile edge intelligence, *mobile edge computing* (MEC) units can be installed on RSUs, which enables them to perform both storage and computation functionalities [13, 14]. This is the key enabler of *edge caching* techniques

[15]. These techniques bring content closer to end users and allow them to reduce the frequency of accessing content from content providers by directly accessing it through caches in the RSUs, and hence it is an effective approach to resolve the challenge of network latency and backbone network congestion due to a massive amount of remote requests to the content provider [16, 17]. Nevertheless, the challenge of frequent intermittent connectivity due to vehicles' high-speed mobility means that vehicles may not be able to finish a content transmission before leaving the currently connected RSU, as a result of which, they have to re-establish the connection to the remote server for the remaining parts at a drastically reduced data rate [12, 18]. This inevitably causes the user experience to be downgraded. *Proactive edge caching* [8, 19, 20] has been recognised as a promising solution to the above issue. It not only makes content close to the vehicular users but also predicts where they may need content in advance through prediction algorithms. Proactively caching the desired content at the future RSU(s) allows vehicles to continue their earlier incomplete content transmissions immediately after accessing the new RSU without having to request the content again from the remote server. Such a proactive caching technique is referred to as *mobility-prediction based*. With regard to mobility prediction, the computation capability in the MEC unit is again the key enabler.

Proactive caching at an accurate future RSU relies on effective prediction. For prediction purposes, the rapid development of machine learning (ML) has played an important role. ML algorithms, ranging from sequence prediction algorithms such as Compact Prediction Tree (CPT) and CPT+ [4, 5] to Deep Learning e.g., Long Short-term Memory (LSTM) [21], have been applied to relevant works in [8, 19, 20]. On the other hand, predicting the next RSU as a proactive caching node is a direct application of reinforcement learning (RL) because every prediction of a proactive caching node is a decision to make. The goal of any RL problem is to map perceived *states* to *actions* by learning a *policy* function. Nevertheless, in systems that do not have to be represented by states, the learning problems become *stateless* decision problems and the learning agent becomes stateless, which significantly reduces the number of trials needed to

learn a mature strategy and speed up the learning process [22]. This is of great help in a dynamically changing vehicular environment. *Multi-armed bandit* (MAB) problems [23, 24] and the extension *contextual* MAB (cMAB) are basic instances of RL problems or to be specific, single state model-free RL problems, where a learning agent does not have to build up a model of the environment. This feature makes it efficient in dealing with the variable vehicular environment. It has also attracted significant attention in various applications, from recommendation systems and information retrieval to healthcare and finance, thanks to its excellent performance combined with certain attractive properties, such as learning from less feedback citebouneffouf2019survey. In a bandit problem, the agent, i.e., the bandit, takes an action to achieve an immediate reward without states being involved, aiming to maximise the total amount of rewards.

The purpose of the work presented in this thesis is to apply machine learning techniques for mobility prediction to achieve proactive caching in vehicular networks. Specifically, it aims to predict the next RSU that a vehicle will connect to, such that the earlier unfinished content can be proactively cached at the predicted RSU and continue to be transmitted immediately when a new connection is established, without having to request from the remote content providers. More accurate prediction means a higher chance of a *cache hit*. To this end, this thesis will design intelligent vehicular systems based on sequence prediction algorithm and more importantly, MAB learning, and apply the designed systems in various traffic scenarios of modern cities. The ultimate goal is to prove the effectiveness and applicability of mobility-prediction methods to proactive edge caching in different realistic scenarios whilst persistently improving prediction accuracy. Given the above objectives, this thesis bases the study on the following assumptions and there are other more specific assumptions regarding the work chapters later, which will be pointed out in relevant places (mainly in the introduction of network architecture in Chapter 4):

- The vehicular network considered in this thesis is MEC-enabled. This is because the network edge RSUs in this thesis are capable of machine learning and content

caching. This means that the RSUs need to be equipped with storage and computation capabilities. Therefore, as a technique that can enable both caching and computing, the MEC architecture is adopted. However, this thesis does not investigate how computational and caching resources are consumed.

- The content is abstracted and assumed to be fragment/chunk based type, rather than a particular type such as high-resolution map or videos and vehicular users in the network only have one type of content under transmission. The size of the content is designed to be sufficiently big so that proactive caching can take effect. The legitimacy of this assumption will be elaborated in Chapter 4.

1.2 Hypothesis

The following hypothesis has guided the research work presented in this thesis:

"*Mobility-prediction techniques can effectively enable proactive edge caching, and the degree of improvement depends on the prediction accuracy.*"

The high-mobility peculiarity of vehicular networks has posed the challenge of frequent intermittent connection. Such frequent link reconnection means that vehicles may not receive the entire content being transmitted before leaving the connected RSU, and re-establishing the connection to the remote content server is inevitable, resulting in a drastically reduced data rate. Mobility-prediction based proactive edge caching can be a potential technique to address this challenge. Accurate predictions of the network edge node for vehicular users allow the network to proactively cache the desired content at precise edge locations such that seamless transmission can be achieved. This thesis, therefore, focuses on developing mobility-prediction algorithms to predict the network edge node i.e., the next RSU along the vehicle path, and then establishes corresponding proactive edge caching systems and verifies their effectiveness through metrics such as network delay. In addition, this thesis continuously improves the prediction accuracy of the developed algorithms and adapts these algorithms to a variety of challenging

vehicular environments and traffic scenarios, because higher prediction accuracy can result in better proactive caching performance.

1.3 Research Contributions

The contributions of the thesis are summarised and listed as follows in descending order of their importance:

- Mobility prediction algorithms based on online multi-armed bandit (MAB) learning and contextual MAB (cMAB) to solve the proactive caching problem in vehicular networks are proposed in Chapter 5, where the problem is formulated as an independent multi-agent MAB problem by modelling RSUs as learning agents. Compared to traditional reinforcement learning techniques, MAB is more computationally efficient and is applied for the first time in the context of mobility prediction and proactive caching. The advantage and distinction of the proposed approaches to other work in [19, 20] are that they do not require massive offline training and hence, are highly adaptable to fast-changing vehicular environments. Another main contribution of Chapter 5 is the specifically extended subjective logic framework for uncertainty analysis with entropy, which provides an insight into the uncertainty behind the learning-based proactive caching systems including uncertainty variation and correlation with prediction accuracy. As far as is known, no work in the literature has done this.

- Chapter 6 presents a novel *Hybrid cMAB Proactive Caching System* (HCPC) with a specially designed switching mechanism that allows RSUs to adaptively finalise their predictions between the *Dual-context* (two-dimensional) and *Single-context* (one-dimensional) cMAB algorithms. In particular, the dual-context cMAB is creatively designed to use the vehicle ID and the previous RSU as combined context, since the dual context using the previous RSU and the penultimate RSU discussed in Section 5.6 has limited accuracy improvement due to the limited

number of cases that can benefit from it. Moreover, the novel hybrid mechanism designed in this chapter relies on the real-time online learning performance of the underlying algorithms compared to the hybrid scheme in [12], which relies on the quality of the offline dataset to determine the first- or second-order Markov chain model. In addition, the hybrid system is further developed into two variants: a Vehicle-Centric system that implements vehicle-level switching and an RSU-Centric system with RSU-level switching for comprehensive performance comparisons, so that the potential of Single- and Dual-context can be fully exploited.

- The SPPC system presented in Chapter 4 is an original system that utilises a sequence prediction algorithm to solve mobility-based proactive caching. It demonstrates for the first time the feasibility and superiority of sequence prediction for solving such problems in vehicular networks. The advantage of using sequence prediction to predict the next RSU is that it can be more convenient and straightforward compared to the works in [19, 20] which applied deep learning for direction prediction and then inferred the next RSU.

- The systems proposed in Chapter 5, as well as Chapter 6 implement MAB and cMAB techniques on individual RSUs in a distributed way to enable instant learning and prediction, whilst previous similar works e.g., [12, 19, 20] were based on centralised approaches and offline training. The agent RSUs are capable of learning and making decisions independently and form an independent Multi-agent reinforcement learning (MARL) system.

- This thesis has created a variety of traffic scenarios simulated by SUMO in two modern cities with significantly different characteristics and layouts, Las Vegas and Manchester, from USA and UK respectively. The aim is to test the applicability and adaptability of proactive caching systems. In particular, Chapter 6 designs three more realistic traffic scenarios: *Commuting traffic, Random traffic, Mixed traffic* in the two cities to evaluate the system performance in a more

comprehensive way. By contrast, the closest previous work in [20] only considered a highway and a single intersection scenario simulated in SUMO. The work in this thesis has provided more comprehensive traffic scenarios for evaluating the proposed algorithms and systems.

1.4 Thesis Outline

The rest of the thesis is structured as follows:

- Chapter 2 first presents a general review of edge caching in mobile networks, followed by introducing edge caching in vehicular networks with a brief overview of vehicular networks. An overview of proactive caching in general mobile networks is presented. Furthermore, prediction algorithms for proactive caching are also discussed.

- Chapter 3 discusses the underpinning techniques applied in this thesis. It first presents an introduction to vehicular traffic simulation in *Simulator of Urban MObility* (SUMO) and the event-driven network simulation method used in this thesis. Next, it discusses the technical and theoretical details of machine learning and prediction techniques including sequence prediction algorithm, reinforcement learning technique as well as the *multi-armed bandit* (MAB) problem.

- Chapter 4 introduces the work that designs a proactive caching system based on sequence prediction algorithm, named *SPPC* system. It first describes the architecture of the MEC-enabled vehicular network that the entire thesis is based on, and then introduces the technical design details of the SPPC system. Traffic simulation scenarios and areas, as well as system performance results, are also presented.

- Chapter 5 concentrates on online-learning proactive caching system based on MAB and contextual MAB (cMAB). This is a fundamental work that first

conducts MAB learning in regard to mobility prediction. The chapter discusses in detail the design of the two proposed MAB-based algorithms based on the theoretical background presented in Chapter 3. Moreover, it also introduces the Subjective Logic uncertainty analysis framework and elaborates on how it is applied to the analysis of proactive caching systems. The simulation results as well as theoretical analysis, time complexity, and convergence of the proposed algorithms are also covered in this chapter. Furthermore, the end of the chapter discusses an extended work on two-dimensional context cMAB.

- Chapter 6 proposes a Hybrid cMAB Proactive Caching System that implements both single-context and dual-context cMAB. This is a further exploration of the potential of cMAB algorithm, based on the work in Chapter 5. This chapter covers the elaboration of the proposed hybrid cMAB system as well as the two parallel cMAB-based prediction algorithms. In addition, it designs modified simulation environments and traffic scenarios in comparison to Chapter 5. The relevant simulation results are demonstrated in detail. In the end, the chapter further investigates and discusses the limitations of cMAB algorithm.

- Chapter 7 presents the conclusions of this thesis, summarises its original contributions, and discusses a number of recommendations for future work.

Chapter 2

Literature Review

2.1 Introduction

Significant research has been done on various aspects of vehicular networks, ranging from spectrum allocation to smart cities. This literature review chapter aims to provide a comprehensive review of the studies that are highly related to the subject of the thesis. First of all, a general review of edge caching in mobile networks is presented. Edge caching in vehicular networks is then introduced, starting with a brief overview of vehicular networks. The following section focuses on proactive caching in general mobile networks, and meanwhile, prediction algorithms applied to proactive caching will also be elaborated on in great detail.

2.2 Content Caching in Mobile Networks

This section focuses on content caching technologies in general mobile networks. An overview of mobile edge caching will be provided first, following which is the discussion of caching in vehicular networks. Finally, the research on proactive caching in the literature will be reviewed.

2.2.1 Mobile Edge Caching

Current wireless networks are facing challenges on network capacity and backhaul links posed by the increasing demand for massive multimedia services. Mobile Internet is the main driver of the exponential traffic growth since it takes over a considerable amount of traffic from conventional Internet [16]. The majority of such traffic is non-real-time and it has been demonstrated in [25] that video on demand (VoD) will generate more than 69% of mobile data traffic by the end of 2019.

The emerging mobile edge caching techniques are considered as promising solutions to cope with the above situation. The reason that network congestion and network resource wastage occurs is because in traditional mobile network architectures, users'

content requests are usually satisfied by Internet content providers [1]. Inevitably, mobile networks suffer a significant amount of duplicate traffic. In contrast, caching popular content at the network edge such as at base stations (BSs), RSUs and user equipment (UEs) is an effective way to avoid transmitting the same content repeatedly, thereby improving users' QoE because of reduced service delay. In addition, the recent advances in the storage industry make it possible for the network to exploit a large amount of storage resources anywhere in the network. Figure 2.1 illustrates a generic architecture of edge caching in the general mobile network. However, employing caching techniques is easier said than done and many issues should be studied. The remainder of this subsection will present related work based on two problems: where to cache and what to cache.

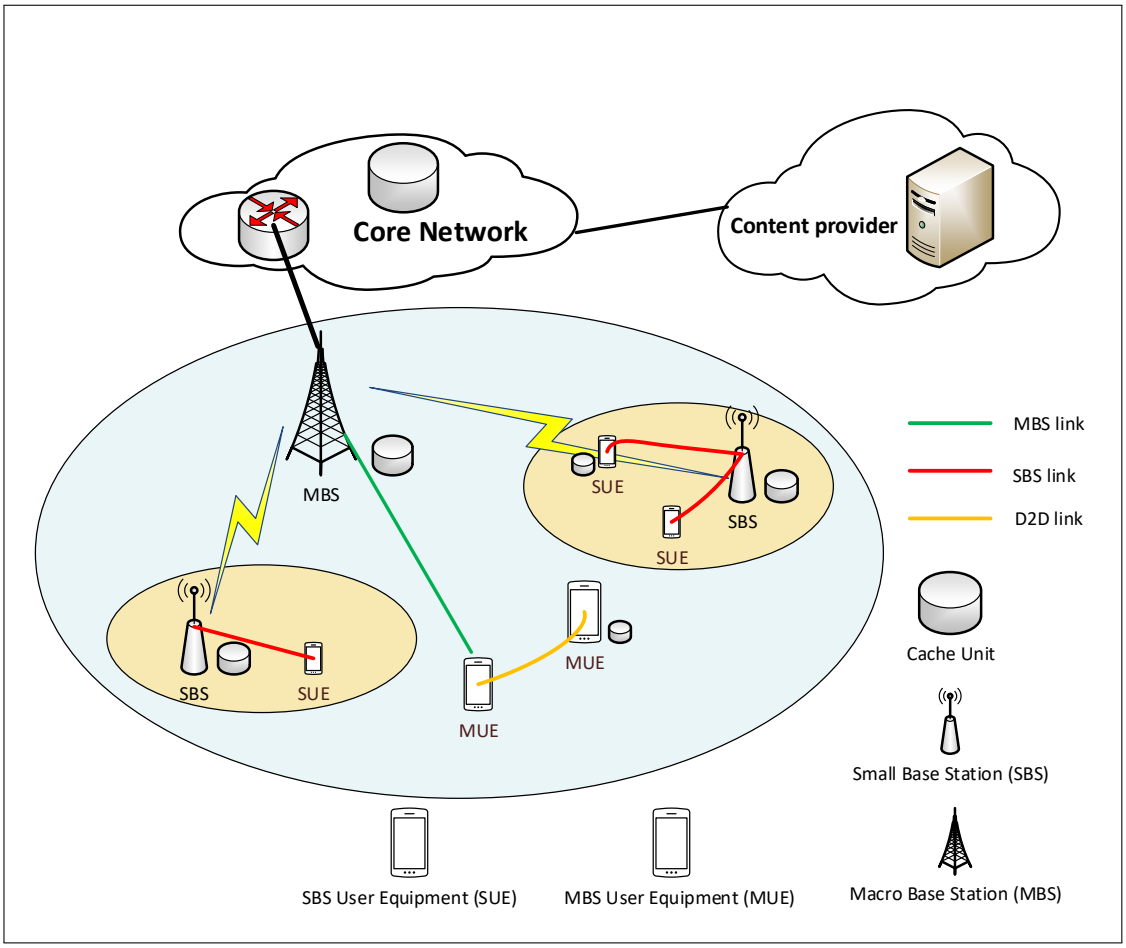


Figure 2.1 A general architecture of mobile edge caching, redrawn from [1]

1. *Where to Cache*

   A number of caches in wireless networks have been deployed in the core network e.g., EPC in LTE since it is technically easier than deploying in the radio access network (RAN). On the other hand, places for caching in the mobile edge basically include two types: infrastructure caching and infrastructure-less caching.

   - *Infrastructure caching* As shown in Figure 2.1, macro BSs (MBSs) and even small BSs (SBSs) are expected to be equipped with cache units. Caches deployed at BS level is called infrastructure caching. In contrast to caching in the core network, edge caching at BSs dramatically alleviates backhaul congestion [26]. From the point of view of caching hit probability, caching at MBSs in heterogeneous networks (HetNets) obtains a better hit rate because of the wider coverage range of MBSs. The work in [27] jointly considered the problem of optimal content placement at BSs and user-BS association problem to achieve an optimal trade-off between load balancing and backhaul saving. An efficient algorithm was proposed iterating between cache-aware user association and association-aware content placement and the result showed that a significant amount of backhaul traffic reduction was obtained. Despite the positive achievement, the authors considered the user association as static and only single connection between users and BSs, which is not realistic in heterogeneous networks. In order to improve the video capacity in mobile networks, Ahlehagh et al. [25] studied the performance of reactive and proactive caching at MBS (eNodeB in their work) utilising user preference profiles. In conjunction with edge caching, they proposed video-aware backhaul and wireless channel scheduling techniques. Video capacity was improved by 300% compared to that without RAN caches. It is worth mentioning that the popularity of videos was also well-studied in this paper. Additionally, as ultra-dense heterogeneous networks become a popular trend in the next-generation wireless networks, deploying caches at SBSs is also an excellent choice and has been studied in the literature,

because SBSs are usually much closer to end users with higher data rates. For example, Bastug et al. [28] considered cache-enabled small cell networks with the stochastic distribution. Users could be served by SBSs through backhaul links or local caches. The authors concluded that increasing SBSs and overall storage size could achieve a reasonable outage probability.

- *Infrastructure-less caching* Infrastructure-less caching means caching at the user level. Smart devices today such as smartphones, tablets and vehicles have already possessed or are able to carry large storage. Therefore, the storage of smart devices should be utilized efficiently. It is obvious that by infrastructure-less caching both the traffic of BSs and the backhaul network can be further alleviated and the QoE of users can also be greatly improved. The network may push content to users according to known or predicated content popularity or user demand, namely content push, self-caching or pre-fetching [26]. However, in reality, this highly depends on a user's willingness due to energy consumption for caching. So, it is an interesting research topic on how to stimulate users to cache popular content. On the other hand, device-to-device (D2D) communication can be established for content sharing when the content is not cached in a user's local storage but in other users in proximity, namely D2D caching. In such a case, only those UEs that are willing to share their cache resources will be considered caching "helpers". Social relations among users and their common interests were considered in [29] to propose a caching-based D2D communication. Similarly, an opportunistic cooperative D2D communication scheme was proposed in [30] to achieve high throughput. Interference among D2D links was carefully controlled by exploiting caching capability at the devices.

2. *What to Cache*

There are two necessary elements to consider when deciding what to cache in the network: content popularity and user preference. By making full use of these

two factors, a maximum hit probability can be achieved, i.e., the chance that the content requested by users is cached in local caches.

- *Content Popularity* Content popularity defined in [16] means "the ratio of the number of requests for a particular content to the total number of requests from all the users". The *Zipf* distribution which is a type of power law distribution has been used to represent the popularity of content [31] with two parameters, content catalog size N_f and a skewness parameter β. This popularity model is the most popular one in current works assuming the content popularity is static or at least constant for a period of time. Unlike the Zipf distribution which is a static model, the authors [32] proposed a dynamic popularity model namely the Shot Noise Model which uses two parameters to model each content: the duration reflecting the lifetime of the content and the height for instantaneous popularity. Besides, due to the benefits to many applications, predicting content popularity has become a hot topic in the research field. One of the frequently used prediction algorithms is cumulative view statistics. However, it is challenging to predict content popularity in a wireless environment because of the dynamic number of users in the coverage of a BS and the restricted number of cumulative requests within the period of popular content existence [16].
- *User Preference* Owing to the fact that a user generally has a strong preference towards particular content categories, the user preference profile consists of the probability that a particular content is requested by a specific user during a certain amount of time and varies among users [16]. Similarly, with big data analytics, user preference is also able to be predicted by machine learning methods, e.g., collaborative filtering based on the content request record of users and the similarity among them [33]. Motivated by the effectiveness of collaborative filtering and to fill the gap in the literature where user preference has not been effectively distinguished from content popularity, Chen et al. [34] studied optimal caching policy for cache-

enabled D2D communications by leveraging the user preference learned by collaborative filtering. Simulation results showed that the offloading gain can be remarkably improved by using the proposed caching policy when the user preferences are less correlated.

Since users are demanding higher data rates and lower latencies for mobile networks, the network architecture is evolving from BS-centric to device-centric and content-centric. The concept of mobile edge networks has arisen recently, with the underlying idea to move network functions nearer to end users by utilising software-defined networks (SDN) and network function virtualisation (NFV) [1]. Wang et al. [1] defined mobile edge networks as "A mobile network architecture that deploys and utilises flexible computing and storage resources at the mobile network edge, including the radio access network, edge routers, gateways and mobile devices, etc., with the help of SDN and NFV technologies". Evolving from mobile cloud computing (MCC), MEC enables network edge to possess cloud computing capabilities and has been recognized as a potential technique to compensate for the drawbacks of MCC (e.g., long latency and high bandwidth requirement).

As two fundamental components of mobile edge networks, MEC and mobile edge caching should be utilised cooperatively to achieve better network performance and higher user experience. For example, in augmented reality applications, MEC can help extract key features from originally captured videos so that caching and transmission resources could be saved. However, many studies treat storage and computing resources separately. Therefore, Zhang et al. [35] proposed a mobility-aware cooperative content caching framework by leveraging MEC for edge caching enhancement. Specifically, they explored both storage and computing capabilities of caching nodes and utilised MEC resources to reduce the size of content files so that the caching ability of the node can be improved. Moreover, they raised the concept of a vehicular caching cloud and proposed a vehicle cloud-aided cooperative caching scheme where caching and computing resources at the network edge were jointly scheduled. With this scheme,

both the MEC resources on BSs and the caching capacity of smart vehicles were fully utilised, and compared to the no vehicle-aided scenario, the average downloading latency could be further reduced especially when the number of content types exceeded 10. However, the computing resources were only used for compressing content files and no investigation was conducted in prediction tasks with MEC resources. Jiao et al. [36] studied communication, computing, and cache (3C) trade-off problems for proactive content caching in cellular networks for vehicular media applications. The 3C resources were measured by transmission hops, the scale and accuracy of user demand prediction, and the number of cached copies, respectively. Their results have revealed that to complete a content delivery task, communication resources could benefit from investments in computing and caching, with linear and logarithmical reduction achieved respectively.

2.2.2 Edge Caching in Vehicular networks

Overview of Vehicular Networks

Network Architecture Vehicular networks can be recognised as a special type of wireless mobile network. In some of the literature, vehicular networks are also referred to as vehicular ad hoc networks (VANETs), a subclass of mobile ad hoc networks (MANETs). Regardless of the naming differences, the vehicular network being discussed in this thesis is a network where it is assumed that all nodes are vehicles moving at various speeds, with the aim of enabling communication and data exchange between vehicles on the road and between vehicles and roadside infrastructures. For this purpose, on-board units (OBUs) and roadside units (RSUs) must be placed on vehicles and the road, respectively.

A general architecture of a vehicular network is illustrated in Figure 2.2. Communication scenarios in vehicular networks can be classified as Vehicle-to-Vehicle (V2V), Vehicle-to-Roadside units (V2R), Vehicle-to-Infrastructure (V2I), Vehicle-to-Cloud, and Vehicle-

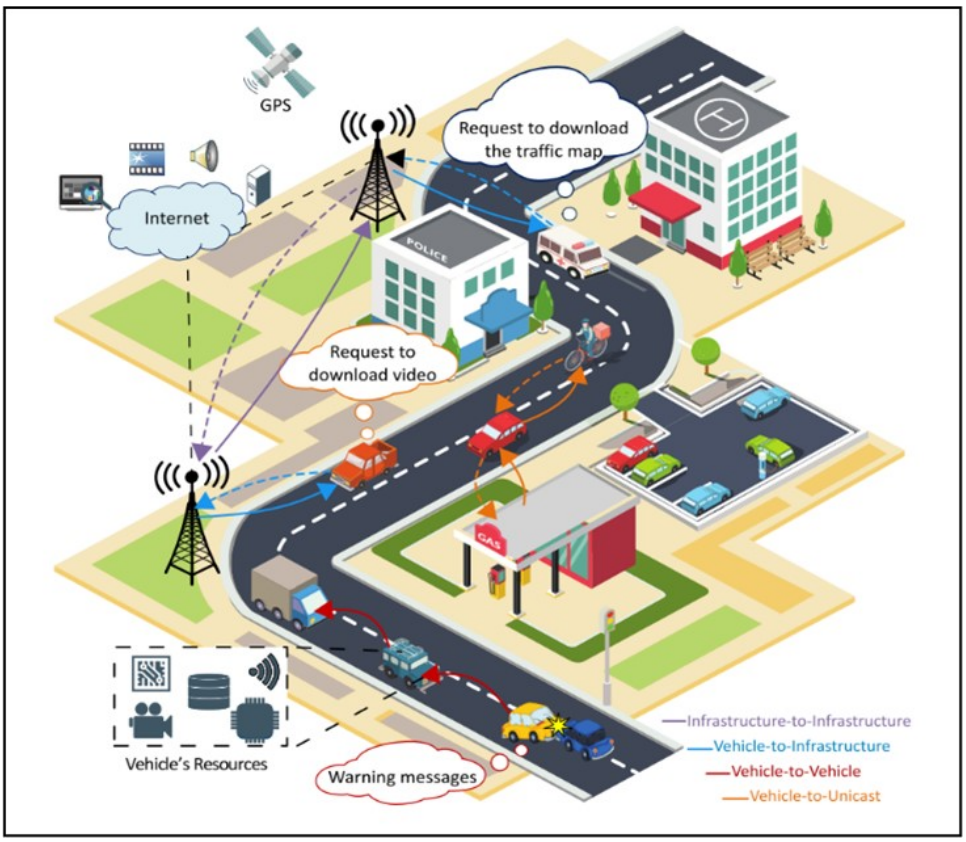


Figure 2.2 A general architecture of the vehicular network (directly reproduced from [2])

to-Pedestrians. As RSUs are also parts of the telecommunication infrastructure, V2R is included in V2I in this review. Vehicle-to-cloud and vehicle-to-pedestrians are out of the scope of this work. Due to the factors such as high-speed mobility, shorter inter-connection time, and fast-changing network topology, the protocols designed for the conventional wireless network do not work well for vehicular networks [37]. Dedicated Short Range Communications (DSRC) [38] and Wireless Access in Vehicular Environments (WAVE) have been proposed to tackle these challenges. Meanwhile, IEEE has improved the 802.11 protocol, referred to as IEEE 802.11p, to be used as WAVE [39].

Characteristics and Applications This subsection briefly discusses a range of distinguishing features of vehicular networks compared with classical wireless mobile networks and their potential applications.

The vehicle network is characterised by four main aspects:

- *Rapidly changing topology*

  A typical feature of moving vehicles is their high speed, especially at the motorway resulting in highly dynamic network topology. Besides, some vehicles' movements are unpredictable, which is also a cause of topology changes [40]. As a consequence, the lifetime of communication among nodes is usually rather short. Apparently, this situation gets even worse when vehicles are moving in opposite directions compared with that where vehicles move in the same direction. In an urban scenario, extra challenges arise because of multiple roads and crossroads. Such intermittent behaviour has a negative impact on routing mechanisms and quality of service (QoS) guarantees.

- *Variable network density*

  The network density of a vehicular network largely depends on the traffic density which usually changes according to the time of the day (urban traffic) or areas (suburban traffic). Low vehicle densities may cause a sparse network scenario so, how to optimally deploy RSUs in such scenarios is a research topic. However, hand-off management mechanisms are important when handling movement from one access point to another.

- *Predictable mobility*

  Different from other MANETs, the mobility patterns of the nodes in vehicular networks are anticipated due to the fact that vehicles are constrained by road topology. However, although there are methods to predict mobility patterns, it is still a rather challenging task and requires prior knowledge of the neighbours, excellent communication connectivity and efficient prediction algorithms [41].

Instead of predicting the exact position of vehicles, this thesis focuses on predicting the next RSU that vehicles are likely to access in the future. This is what is going to be studied in depth in this thesis by sequence prediction algorithm and reinforcement learning techniques.

- *Computational and caching capabilities*

  In contrast to traditional mobile networks, a vehicular network is comprised of vehicles as moving nodes, which means that there is a chance that these nodes are equipped with sufficient resources for communication, computation, and storage to support many vehicular applications. Researchers have tried to leverage these resources to improve network performance. For instance, in [35] the authors studied mobility features of moving vehicles and proposed a mobility-aware edge caching scheme by utilising the caching ability on vehicles. Their simulation results have shown that the proposed scheme achieved the lowest latency since more content is stored in the edge nodes and less content needs to be transmitted from the content provider.

According to the properties and characteristics of vehicular networks, their applications can be generally categorised as infotainment applications, safety applications, and traffic information applications.

- *Infotainment Applications*

  This classification sometimes is also referred to as the comfort/entertainment application. It aims to improve the comfort level of both drivers and passengers in the car. This is becoming increasingly important especially when the era of autonomous driving is ahead. These applications not only provide essential information for the journey such as weather information and locations of restaurants, petrol stations, or hotels but also offer passengers the opportunity to enjoy online gaming and video streaming while on the go.

- *Safety Applications*

As it primarily concerns human lives, this category has rigorous requirements on network delay, QoS, and security. Safety applications using V2V communication or V2I communications or both include but are not limited to intersection collision avoidance, public safety, sign extension (including in-vehicle signage, curve speed warning, etc.), vehicle diagnostics and maintenance, emergency warnings (blind spot warnings or road condition warnings), and so forth [42, 43]. Various access technologies e.g., LTE (Long-term Evolution), WiFi, WiMax, and visible light communication (VLC) are being used to ensure that safety-related messages are transmitted efficiently and successfully [44, 45].

- *Traffic Information Applications*

  These are applications that offer drivers real-time information about the traffic ahead. This can not only improve the driver's view but at the same time help them select a less congested route to save fuel. Communication methods for such applications are normally unicast and unlike safety applications, less stringent requirements are needed in terms of transmission. What is worth mentioning is that there are two types of traffic view, the short-range local traffic view and the long-range extended traffic view respectively, among which the former can be realised via direct communication with neighbouring or close vehicles and the latter via multi-hop communications [46, 47].

As a special type of wireless network, vehicular networks not only possess basic communication features e.g., cellular communication and D2D communication but they also have enabled many applications from safety, transportation efficiency to information and entertainment. Like in other mobile networks, video on demand (VoD) is also a main type of content delivery traffic in vehicular networks. Therefore, it has attracted considerable attention in academia to improve the delivery efficiency of content, especially large-size videos, in high-speed vehicles.

Caching has become a promising technology for higher network efficiency and relieving backhaul traffic. Ding et al. [48] conducted a study of edge caching in vehicular

networks. In this study, RSUs, which have weak backhaul capacity to the core network, not only perform transmitting signals but are also provided with large storage capacity to cache popular content. They aimed to minimise the average time that an OBU (i.e., a vehicle) downloads a file. Three algorithms were proposed to allocate content to RSUs, which are the optimal algorithm, sub-optimal algorithm, and greedy algorithm. Specifically, the optimal one is based on exhaustive search and chooses the best one that has the lowest average delay performance among all file-distribution schemes whereas the sub-optimal method allocates the most popular files in all RSUs and makes substitutions repeatedly. The greedy algorithm gradually fills the RSU storage with the best choice each time. It has been concluded that the average delay for downloading a file can be reduced by 70% compared to no caching at RSUs. In addition, the storage capacity of RSUs, vehicle speed, and the number of RSUs have different impacts on the performance of the three algorithms.

Similarly, RSU caching was also focused on in [49]. The idea that makes this paper different is that the authors involved multiple content providers (CPs) in the caching problem. Their motivation behind this is that because of RSUs' storage limitation and the high cost of deploying caches for mobile network operators (MNOs), CPs should compete for caching resources (via a certain payment) to let its popular content stored at the network edge so that MNOs can benefit from this. An auction-based solution was proposed where the CPs acted as bidders while the RSUs' caching storage owned by MNOs played the role of objects. Then multi-object auctions were set up and solved by Market Matching Algorithm so that eventually, all the storage of RSUs could be effectively allocated with contents. However, a practical implementation is needed to see how effectively this model can work in commercial vehicular networks.

Su et al. [15] analysed the strategy of vehicles of deciding where to get the requested content to reduce transmission delay and proposed a novel caching scheme based on the collaboration among vehicles and RSUs in order to improve hit ratio and reduce caching overhead efficiently. This paper developed a cross-entropy-based dynamic content caching algorithm to cache the content at the edge of VCNs and this algorithm

which considered both the content size and popularity was compared with PWDP, LRU and LFU (detailed explanation of the three schemes can be found in [15]) in terms of relative delay, hit ratio and overhead. Although positive results were obtained, the impact of the number of RSUs was not evaluated (4 RSUs only), as the increase of RSU may affect the total cache size. Liu et al. [50] considered the joint problem of cache resource allocation among the RSUs and content placement in urban vehicular networks, by exploiting the vehicles-RSU "contact pattern" set. A greedy cache allocation algorithm was proposed to solve the formulated problem and compared with the greedy algorithm on equal cache (GAEC) and the popularity-based caching algorithm on the equal cache (PAEC), the proposed algorithm performs 10% better due to the consideration of non-uniformity of cache size, and the joint optimisation of cache allocation and content placement. However, the authors considered the contact pattern as certain, which may not be the case in a real transportation network as most vehicles may change their routes based on external factors. Yao et al. [51] proposed to use the Prediction based on Partial Matching (PPM) method in Vehicular Content-Centric Network (VCCN) to predict vehicles' probability of reaching different hot spot regions and select nodes with longer sojourn time in a hot region as caching node. However, PPM-based approach is only suitable for selecting users belonging to grouped clusters, which does not fit with RSU-OBU content prefetching scenarios.

Researchers have also studied in-network cache strategy in Information-Centric Networking (ICN)-based vehicular networks. For instance, Zhao et al. [52] studied the caching policy in ICN vehicle-to-vehicle scenario. They defined community similarity based on content similarity and moving similarity and proposed a Community Similarity and Population-based Cache Policy. Content naming was also mentioned in this paper as it is an important factor to calculate content similarity so that a vehicle can decide whether to cache the data being forwarded. Furthermore, to decrease cache replacement overhead, they also put forward a popularity prediction-based cooperative cache replacement strategy. By conducting simulations on OMNet++, the average time delay was reduced significantly via the proposed scheme. Similarly, the work in [53]

considered an efficient caching strategy in a V2V scenario based on CCN architecture. Different from previous works, the proposal of their caching policy took into account different requirements of various vehicular applications (emergency applications, intelligent transport, and entertainment) as well as the peculiarities of vehicular networks. As a result, the proposed scheme determined the caching time and chose the valuable content to store according to the type of application.

2.2.3 Proactive Caching

Proactive caching is a new caching paradigm studied in recent literature. It has been recognised that future wireless networks will be disruptive in terms of the way they interact with end users. Unlike the traditional BS-centric architecture, terminals are assumed as "dumb" and the interaction between the network and users is completely reactive. However, such an interaction mode will change in the future mobile networks to context-aware, user-centric, and proactive/anticipatory in essence [54]. In terms of caching, the general idea of proactive caching is to predict the possible files that users will request in the future and prefetch the content at proper local caches, by exploiting the network's and users' context information, anticipating users' demand and behaviour and investigating content related information e.g., popularity. This means when a user actually initiates a request, the content may already be stored in its own spare storage space or local cache in proximity and information can be directly pulled out from the cache instead of accessing the backhaul network. To this end, machine learning techniques should be used to fully utilise the huge amount of user data in the network and find the optimal prediction results.

A number of works have been conducted to investigate the benefit of proactive caching in wireless networks. In [55], Bastug et al. provided a case study of proactive small cell networks (SCNs) and investigated the problem of backhaul offloading in SCN where proactive caching played a vital role. Supervised learning (collaborative filtering) was adopted to exploit user-file correlation to infer the probability that the u-th user

requests the i-th file. By training some existing information regarding users' preferences, every SBS obtained a popularity matrix and made its proactive caching decision to store the most popular files greedily until no storage space remains. However, the mobility issue was not taken into account in this paper. The authors extended their work in [68], where they adopted collaborative filtering to model and predict the spatio-temporal user behaviour for proactive caching decisions. They intended to use a big data platform in the core network of a Turkish telecom operator, to tackle the complex problem of tying content popularity prediction with user behaviour. Nevertheless, this work seems to be just an extension of the case study in [55] with real-world big data. The investigation on user behaviour was not studied deeply.

Doan et al. [56] concentrated on video proactive caching in cellular networks. Different from previous works that focused only on published videos, this paper also tried to anticipate the popularity of unpublished/new videos. More specifically, a complete process for anticipating video popularity was considered, where they dealt with both published and unpublished videos and utilised old videos as the training set (frequently updated by predicting the popularity of old videos) to predict the popularity of new ones. Through the knowledge of every video's popularity, they proposed a proactive caching strategy for minimising the load of video traffic on the backhaul link. Comparably, Somuyiwa et al. [57] considered proactive content caching on mobile users in the framework of an online social network (OSN). By taking the time variations in popularity into consideration, the authors tried to figure out two questions in proactive caching: what content to cache and when to cache. Then, the proactive caching problem was modelled as a Markov decision process to minimise the long-term average energy cost which are related to the number of contents downloaded and the channel conditions. In general, their proposed caching policy is a threshold-based proactive caching policy and reinforcement learning techniques were adopted to optimise the threshold values.

Despite much work done on proactive caching in generic mobile networks, proactive caching in vehicular networks has not witnessed much research yet. The peculiarity

of vehicular networks is their intermittent connection caused by high-speed mobility. Therefore, mobility-aware caching is essential in dense deployment networks. In such a scenario, when a vehicle is requesting the content of a large size, it may pass several BSs or RSUs, which is very likely to cause a degradation of transmission service. Thus, it may be very important to optimally and effectively cache the contents at the edge nodes on the vehicle's path so that they could be obtained by the user as needed. For this purpose, accurate prediction of vehicular routes or flow of traffic may be remarkably beneficial towards improving caching performance [35]. Khelifi et al. [20] put forward a proactive caching scheme based on vehicular mobility prediction on top of a NDN architecture. The authors focused on RSU caching by predicting the next RSU along a vehicle's path with Long Short-Term Memory (LSTM). As a result, the remaining chunks of the content that is being requested in the previous RSU could be pre-located in the caches of the next few RSUs. The proposed scheme outperformed other caching strategies including Leave Copy Everywhere (LCE), Leave Copy Down (LCD), Edge Caching (EC), and Consumer Cache (CC) (detailed description of these strategies can be found in [58]). However, one shortcoming of this work is when they calculated the connection duration between the vehicle and the RSU based on velocity and distance in an intersection scenario, some possible factors in practice were not considered such as the time spent in waiting for traffic lights or congestion.

Another more comprehensive work regarding mobility prediction-based proactive caching in vehicular networks was conducted by Zhao et al. [12]. The authors also intended to predict the future connected RSUs but they collected a huge amount of traces from real-world VANET testbed deployed in the city of Porto, Portugal, and adopted a hybrid Markov chain-based location predictor to estimate the next connected RSUs of a vehicle. Besides, a lightweight OTT content prefetching mechanism was raised such that more popular chunks of videos could be stored in the RSU cache. By combining the accurate forecasts of vehicle mobility and the prefetching scheme, excellent system performance had been achieved such as RSU cache efficiency and bandwidth savings. Likewise, Grewe et al. [59] conducted their proactive caching work

in a NDN-based VANETs. They added additional information in the initial Interest packet of a vehicle including its position, velocity, and Interest request Frequency so that the data store in the network can decide the right placement of data chunks at RSUs. However, this work only considered a simple scenario in a motorway and did not take into account the mobility direction of vehicles.

All in all, proactive caching stands for a promising caching strategy in wireless networks to tackle the pressure brought by the explosion of mobile data. In vehicular networks, due to many unique features, proactive caching also plays a vital role in addressing the content delivery problems in such networks. On the other hand, research on proactive caching in vehicular networks is far from enough and can be further explored in the future such as considering an optimal caching scheme by jointly considering mobility and popularity, prediction-based caching in different scenarios or with various applications, and so forth. Therefore, it is a valuable research direction to investigate optimal proactive caching strategies in vehicular networks by leveraging the huge amount of big data in such networks, to achieve effective and accurate prediction of vehicle mobility.

2.2.4 Prediction Algorithms Overview

The performance of proactive caching relies on prediction, whether it is for mobility prediction or content popularity prediction. This subsection will focus on machine learning (ML) algorithms that are helpful for proactive caching and provides an overview of prediction algorithms that have been used in the literature for the purpose of proactive caching.

Accurate predictions on mobility, the preference of users, as well as content popularity are important to deliver excellent proactive caching performance. The prior knowledge of users' behaviour and the network content helps the network in many aspects such as energy efficiency, resource management, network congestion, and even lower cost of network deployment. Moreover, the potential value behind big data should be

effectively exploited by machine learning techniques so that communication networks could be more intelligent and efficient. Therefore, some prediction algorithms, which have been adopted in the literature, are introduced in this subsection based on two classifications: mobility-based prediction and popularity-based prediction.

From the viewpoint of user mobility, the relationship between social and geographic data can be used to predict user demand. One of the important types of data is GPS data collected from users' embedded GPS modules, which is especially the case for vehicular users. For example, Nguyen et al. [56] presented a potential approach for predicting users' movement from historical location data. They developed an Android application called Movement Predictor, from which they could collect location data from registered users by GPS signals. They also proposed several ways to extract features of the gathered data and compared three supervised learning models: Markov model, Support Vector Machine and decision tree. In the end, they integrated a properly trained model into the App for a better user experience. In [20], the authors adopted a deep learning algorithm, LSTM specifically, to allow the current RSU to predict the next possible RSU a user will connect to. The data they used in this paper was collected from Simulation of Urban Mobility (SUMO) traffic simulator. Similar work to [20] that also used LSTM is studied in [19]. The authors [12] designed a hybrid Markov chain-based location predictor that is able to switch between first-order and the second-order Markov model based on the available data quality. Similarly, Zhao et al. [60] also predicted the motion pattern of vehicles using a second-order Markov model. Their data was based on 38,900 taxis in Shanghai from May 1 to June 10, 2016, including vehicle ID, timestamps, longitude and latitude, speed, angle, and passenger state (0 for no passenger and 1 for passengers inside). By properly processing the collected data, they computed the cumulative distribution function of the entropy of moving vehicles and with this, predicted the motion pattern of the vehicles. Besides, two more factors (holidays and flow of traffic) were introduced to the model for more accurate prediction. Parija et al. [61] proposed a multi-layer neural network model to predict the future location of subscribers based on the past predicted information.

However, the limitation of this work is that only the users with regular movement can be predicted due to the constrained definition of movement patterns.

In addition to the above-mentioned mobility prediction mechanisms, another popular data mining or machine learning algorithm, sequence prediction, would also be an effective method. Sequence prediction algorithms have been used in various real-life applications such as webpage prefetching and product recommendation. The concept of sequence in such algorithms means an ordered list of symbols. If we regard the RSUs that a vehicular user connects to along its journey as individual symbols ordered by time, there is a possibility of using sequence prediction to anticipate the next location/next RSU of the user. There are a number of approaches to this subject, among which PPM (Prediction by Partial Matching) [62], DG (Dependency Graph) [63] and All-K-Order-Markov (AKOM) [64]. However, as demonstrated by Gueniche et al. [81], both of these schemes build lossy models and ignore some relevant information from training sequences when making predictions. Therefore, Gueniche et al. [5] proposed an efficient tree-based data structure named Compact Prediction Tree (CPT) which can losslessly compress all training sequences and evaluated its performance with various real datasets to be more accurate than PPM, DG and AKOM. As an enhancement of CPT, Gueniche et al. [4] extended their previous work and proposed CPT+ to reduce CPT's size and prediction time and increase its accuracy. The work in [8] has adopted CPT+ as the main algorithm for mobility-prediction based proactive edge caching.

Given the dynamic nature of user behaviour, it is essential to integrate popularity prediction into caching to achieve better network resource utilisation and user experience [65]. Proactive ability can be obtained with popularity prediction to help the network store popular content in-network caches. On the other hand, it also poses challenges to the desirable algorithms in terms of execution speed, prediction accuracy and scalability [65]. [66] and [67] have proposed a typical framework for popularity prediction. The system requires various types of data (e.g., content demand, users' interests, locations, etc.) to be processed so that the transmission patterns as well as social similarities

and differences can be determined. ML technique was implemented to predict video popularity, the performance of which can further be improved by social, temporal, and spatial variations. The authors also developed the ML model to estimate where the video content would be more desirable with different settings and they utilised the output of the model for content replacement schemes. In addition, time series data is also used in video popularity prediction tasks since popularity growth graphs reflecting video popularity can be used to predict future popularity [65]. Xiaoqiang et al. [68] adopted a first-order gray model originated from system control to predict future popularity with popularity records in the past. Rather than focusing on predicting a video's popularity as a whole, Zhang et al. [69] put forward a chunk-based cache replacement method in Information-Centric Networking architecture by exploiting the relationships among chunks of the same video. What they found was that request records of previous chunks of a video stream could be used to predict the popularity of future chunks with a linear weighted combination. However, the methods proposed in [68] and [69] were in an offline mode without parameter update. Thus, to cope with the continuously varying distribution of popularity, online prediction is used with the capability of changing models and updating parameters. In [70], the authors evaluated three experts for popularity prediction, i.e., single exponential smoothing (SES), double exponential smoothing (DES) and the basic expert. SES and DES smooth past observations exponentially to compute future prediction depending on a smoothing parameter while the basic expert does not rely on any tuning parameters but adds to the current solicitation the difference between the current and the previous [65]. In addition, the application of popularity prediction in mobile edge networks can be found in [71], where Hoiles et al. used game theory to improve content dissemination. Two intertwined aspects were considered in their framework. First, video popularity was determined based on users' tendencies and the popularity prediction scheme took video metadata, such as publisher, title, description, and so forth, into account to estimate user request probability which indicates the anticipated demand for videos.

Then, they used the values as the input for the proposed game-theory caching algorithm for optimal edge caching.

However, the limitation of most content popularity prediction methods is that they may require RSUs to collect personal data and preferences of users, which often contain sensitive information. Given the ever-increasing restrictions on security and privacy, this will become increasingly difficult for network operators. Moreover, they are not very effective in vehicular networks because vehicles are fast-moving objects and this causes validity issues in the popularity prediction. In contrast, mobility prediction by predicting the next RSU is more applicable for network operators because the next RSU is usually restricted to a finite set of neighbours and no additional user-sensitive data is required to perform the prediction with effective prediction algorithms, as will be shown in later work chapters. Moreover, it is also essential in dynamic vehicular environments which pose challenges such as intermittent connectivity, hence the main research topic of this thesis.

2.3 Conclusion

In conclusion, the chapter has provided a comprehensive review of the literature that is highly relevant to the thesis. This chapter first reviewed content caching techniques in general mobile networks and further extended to a detailed review of mobile edge caching in vehicular networks and proactive caching techniques. Since proactive caching relies significantly on prediction algorithms, the chapter has also covered an extensive review of machine learning algorithms that have been applied to proactive caching problems.

Chapter 3

Underpinning Techniques

3.1 Introduction

The main purpose of this chapter is to comprehensively discuss the underpinning techniques that support the research work. These techniques include vehicular traffic simulation and network simulation method applied throughout the thesis, the underlying sequence prediction algorithm used in Chapter 4, and reinforcement learning techniques, in particular, multi-armed bandit learning, applied in Chapters 5 & 6.

The rest of the chapter is organised as follows. Section 3.2.1 introduces the simulation methods used in this thesis for mobility and network simulation, together with a brief review of other methods that have been used in vehicular networks in the literature. Section 3.3 introduces in detail the theory of the underlying sequence prediction algorithm - CPT+. Section 3.4 reviews reinforcement learning techniques in general and what follows is an elaboration on the theoretical background of the multi-armed bandit problem.

3.2 Simulation Methods

3.2.1 Vehicular Traffic Simulation

Mobility or traffic simulators are used to simulate real-world traffic. The models developed by these traffic simulators are through refining synthetic models and verifying the outcome with real traces and behavioural investigations. More importantly, for the mobility prediction problem in this thesis, *microscopic modelling* is needed because it generates models that capture an individual vehicle's location, trajectory, velocity, acceleration, and architecture parameters such as the number of intersections, the number of lanes, etc. Such a level of modelling can be achieved by applying traffic simulators and this is the advantage of these simulators compared to traditional tools such as MATLAB [72].

For vehicular traffic, one of the popular simulators recently is Simulator of Urban MObility (SUMO) [3, 73, 74]. Started in 2001, SUMO is an open-source, highly portable, continuous traffic simulation package that generates microscopic models and is specifically designed for large networks. Since then, it has been developed to not only provide a traffic simulation but rather a suite of applications [74], and become more and more popular in academia thanks to its applicability and flexibility in many fields including vehicular networks and intelligent transportation system. Khelifi et al. [20] used SUMO to generate mobility traces based on a part of a Paris intersection and Sun highway in France, both from Open Street Map. Likewise, Elsayed et al. [75] adopted SUMO for traffic generation with 48 road segments and 1000 moving vehicles. The work in [8] also used SUMO to simulate the commuting traffic model of 174 vehicles in two real-world cities. Additionally, to verify the feasibility of the proposed vCache concept, real-world traffic traces were used to create mobility models and position traces for each simulated vehicle with SUMO in [76].

SUMO has integrated applications such as network generation, demand generation and simulation. The general process of the generation of vehicular traffic data used in this thesis is as follows. First, a road network on which the simulation is based should be created. In SUMO, this can be achieved by either using an application named `netgenerate` or importing a digital road map with `netconvert`. *OpenStreetMap* [1] is a free editable map of the whole world and is a good source for SUMO network. This thesis will apply the `netconvert` to the extracted areas of real cities for road network generation. An example network is shown in Figure 3.1. The second important element for SUMO simulation is *traffic demand* i.e., vehicles running through the network. SUMO allows users to generate traffic demands *aka trips* between arbitrary *traffic assignment zones* (TAZs), which is the main method used later in the thesis for generating commuting traffic data. Alternatively, one can apply `randomTrips.py`, a Python tool provided in SUMO package, to generate a set of random trips for a given network. With the trips created for the network, detailed *routes* of these trips

[1] http://www.openstreetmap.org/

Figure 3.1 A screenshot of the road network of Las Vegas shown in SUMO GUI

can then be generated with `duarouter` tool which is developed based on Shortest or Optimal Path Routing rules. Finally, `sumo` is called to simulate vehicular traffic in the given road network and routes, which is a pure command line application for efficient batch simulation and it is helpful and efficient when the researcher only needs traffic traces [74]. The format of the output file from `sumo` is *fcd* that contains floating car data including name, position, angle, and type for every vehicle because such level of detail is essential for the network simulation in this thesis. Alternatively, SUMO provides a graphical user interface (GUI) `sumo-gui` for more customised options and it offers all features of the command line version `sumo` supports [74]. Figure 3.2 shows a screenshot of a single intersection simulated in SUMO user interface. Appendix A provides a comprehensive tutorial of the above process, together with examples of the output data for this thesis.

There are some other available mobility simulators in the literature. Fiore et al. [77] introduced VanetMobiSim which is also an open source vehicular mobility generator. It can produce detailed vehicular movement traces employing different mobility models. However, this simulator does not support real-world map import and the website of

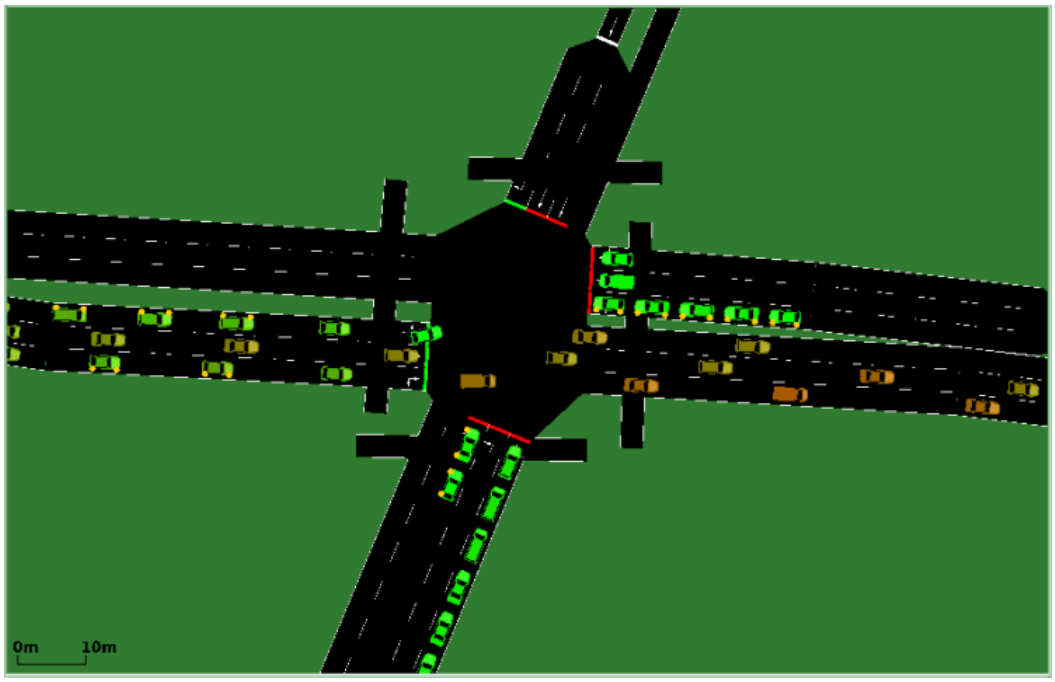


Figure 3.2 A screenshot of a section in SUMO GUI, directly reproduced from [3]

this tool is not particularly friendly to learners due to few tutorial documents being available. The Opportunistic Network Environment simulator (The ONE) [78] was specifically designed for evaluating delay-tolerant networking (DTN) and opportunistic networks. Yao et al. [51] conducted their cooperative caching experiment based on ONE. Traffic Software Integrated System - Corridor Simulation (TSIS-CORSIMTM) [78] is also a microscopic traffic simulation software package but it is designed for signal systems, freeway systems, or combined signal and freeway systems and unfortunately, this is not an open source software.

Despite the above mobility simulators having their own benefits, SUMO seems to be a good choice in terms of its open source, comprehensive tutorials, powerful trace operations, and easy integration with other network simulators. Therefore, it has become the underpinning mobility modelling technique for the research work in this thesis and the technical details will be elaborated on in the next chapter.

3.2.2 Network Simulation

Network simulation for the vehicular network in this thesis is based on Discrete Event-driven Simulation (DES) [79, 80] method which is implemented in MATLAB [81]. Mobile network simulation can be performed through a series of events so this makes network simulation based on DES possible. Generally, a discrete event is a circumstance that causes an instantaneous change in one or more aspects of the system state. As an event occurs, the system proceeds by executing all the changes (associated with each event) to the system in a chronological sequence. The system clock time advances to the time when the next event is due to happen and therefore, such simulation is referred to as event-driven.

In this thesis, events for the vehicular network mainly include "*departure*" (when a vehicle starts its travel and enters the network), "*arrival*" (when a vehicle arrives at its destination and leaves the network), "*content request*" (when a vehicle requests content transmission from currently connected RSU), "*handover*" (when handover happens), "*transmission completed*" (when the requested content has been transmitted to the vehicle completely). These events are known as an *event list*. Figure 3.3 demonstrates a basic procedure of event-driven simulation.

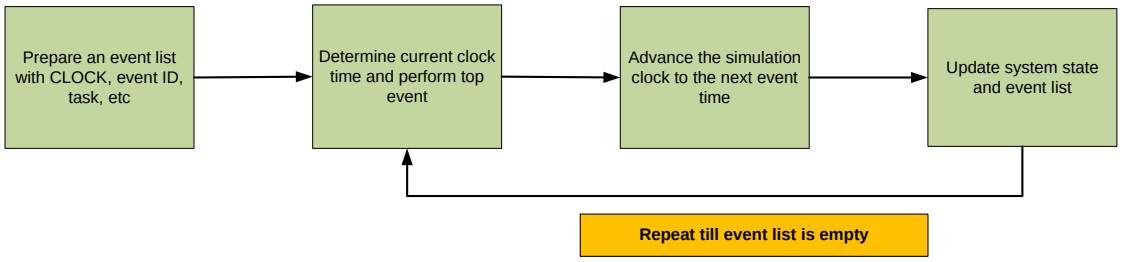


Figure 3.3 Discrete-event driven simulation procedure

Some DES-based network simulators such as Optical Micro–Networks Plus (OMNeT++) [82], Network Simulator Version 2 and Version 3 (NS-2 and NS-3) [83] have been applied in the literature. These network simulators are powerful when underlying networking protocols and communication models are necessary. However, this thesis has chosen to

develop a dedicated DES-based simulation with MATLAB. The reasons for this choice are summarised as follows. First, the primary focus of this thesis is the performance of mobility prediction algorithms regardless of the underlying communication and networking models (as will be clarified in the network model of Chapter 4). Thus, with such simplified requirements, some risks, such as unreliable bugs that may be caused by modules and components in these simulators, can be avoided by developing a dedicated and controllable DES simulator. In addition, significant background work is required to well manage these simulators e.g., OMNeT++, which seems unnecessary given the focus of and assumptions made by this thesis. Third, the *floating car data* from SUMO is the main source of traffic data in this thesis and the dedicated DES simulator in MATLAB is specifically developed to deal with it. Although OMNeT++ and NS-3 may provide interfaces for SUMO, they usually require an external module implemented and this can be another source of unreliable bugs. Therefore, combining all these factors, the specifically developed DES-based simulator in this thesis can provide a straightforward and efficient approach for simulating the vehicular network.

In summary, the DES-based network simulation together with the SUMO simulator in Section 3.2.1 constitutes the simulation module of this thesis. Figure 3.4 shows a structure of the module where the mobility traces generated by SUMO are fed into the DES-based module in MATLAB.

3.3 Sequence Prediction Algorithm - CPT+

Mobility prediction is one of the effective approaches to achieving proactive caching and machine learning is a well-known approach to achieving accurate predictions [84]. An intuitive way to implement mobility prediction in vehicular networks is by predicting the next RSU that a vehicle may visit in the future, which becomes the main focus of this thesis. From a certain point of view, predicting the next RSU is like predicting the next symbol in a sequence, if RSUs of a vehicle's trajectory are modelled as a sequence

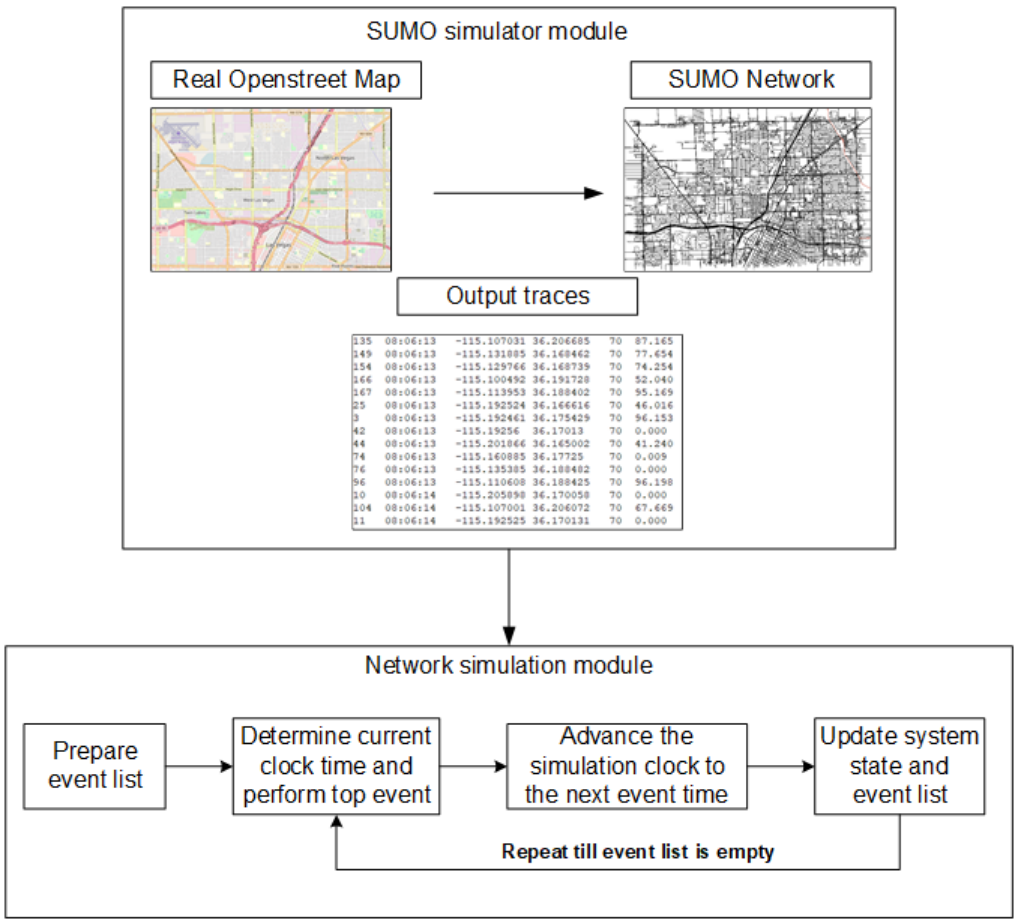


Figure 3.4 Simulation module of the thesis. SUMO generates the vehicular traffic data and the data is then processed in the DES-based network simulation module.

of symbols. Therefore, *sequence prediction* algorithm can be applied to address the problem. It has been one of the most popular machine learning tasks, which consists of predicting the next symbol(s) based on the previously observed sequence of symbols. These symbols could be a number, an alphabet, a word, an event, or an object like a webpage or product. A general understanding of sequence prediction is that it is needed whenever people need to predict what is likely to occur after a previous event. A few applications can be found in various industries. For example, Web Page Prefetching: given a sequence of web pages that a user has visited, the most likely page that a user will visit can be anticipated and pre-loaded, saving time and improving user experience. In the following, the section will introduce an effective algorithm

for sequence prediction, named *Compact Prediction Tree plus* (CPT+), which is the underpinning algorithm for the work studied in Chapter 4.

Proposed by Gueniche et al. [4], CPT+ is essentially an enhanced version of its predecessor, *Compact Prediction Tree* (CPT) developed in [5]. CPT is a sequence prediction model that compresses training sequences without information loss by exploiting similarities between subsequences. It has been proven in [5] to be more accurate than other sequence prediction models e.g., Prediction by Partial Matching (PPM). With the aim of reducing the space and time complexity of CPT, the enhancement that the authors made in CPT+ include three strategies: FSC (Frequent Subsequence Compression), SBC (Simple Branches Compression), and PNR (Prediction with improved Noise Reduction). Such improvement is the main reason why CPT+ has been adopted in SPPC system over CPT. Nevertheless, despite such enhancement, the fundamental prediction mechanism of CPT+ and CPT remains the same and their prediction performance is comparable on many datasets [4] as shown in Figure 3.5. Therefore, the focus in this subsection will be on elaboration on such mechanisms, in general, to help understand what is essential for the SPPC system in terms of modelling, training, and prediction. The technical details of the above enhancement are out of the scope of this chapter and can be found in [4].

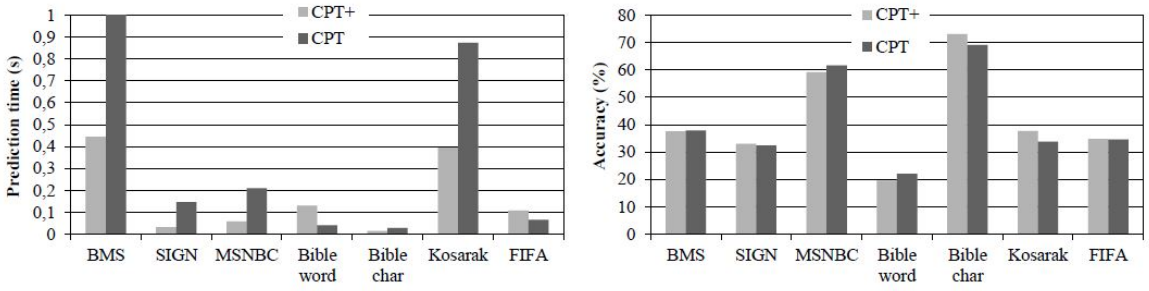


Figure 3.5 Execution time and accuracy comparison of CPT and CPT+ on various datasets (directly reproduced from [4])

3.3.1 Training

The training phase is to build the Compact Prediction Tree and it results in three data structures: i) a Prediction Tree (PT), ii) an Inverted Index (II) and iii) a Lookup Table (LT) [4]. During training, sequences in the training dataset are inserted individually and incrementally to establish the three data structures. The example in Figure 3.6 illustrates the process of creating the necessary structures in CPT+ by continuous insertions of sequences $s_1 = \langle 4, 5, 6 \rangle$, $s_2 = \langle 4, 5 \rangle$, $s_3 = \langle 4, 5, 7, 6 \rangle$, $s_4 = \langle 5, 6 \rangle$ and $s_5 = \langle 8, 4, 5, 4 \rangle$, and this example will be referred to in the following introduction of PT, II, and LT.

- *Prediction Tree* This is a typical *prefix tree* [4] that contains all the training sequences. A node in the tree contains an item, a list of children nodes, and a pointer to its parent node as shown in Figure 3.6. Each full/partial branch is a compact representation of a training sequence, by a path starting from the root to an inner node or a leaf. When a training sequence is inserted, the root is checked first to see if it has a direct child node that matches the first symbol of the sequence. If this is false, create a new child of the root with the first symbol, then the cursor is moved to the newly created child and the remaining symbols are inserted sequentially (e.g., the insertions of sequences s_4 and s_5). Otherwise, the first symbol in the sequence that does not match any symbols in any branch of the tree will be added as a new child to a tree node, and the rest of the sequence is inserted, e.g., the insertion of sequence s_3. According to [5], the time complexity of constructing this tree for N training sequences is $O(N)$, achieved by reading sequences one by one with a single pass of the dataset. If two sequences have first v symbols in common, they share v nodes in the PT (e.g., sequences s_1, s_2, s_3). Therefore, its space complexity is often less than the worst case $O(N \times averageLengthOfSequences)$ [5].

- *Inverted Index* The purpose of the II structure is to provide a means to quickly find which sequence a symbol appears in or find all the sequences that contain

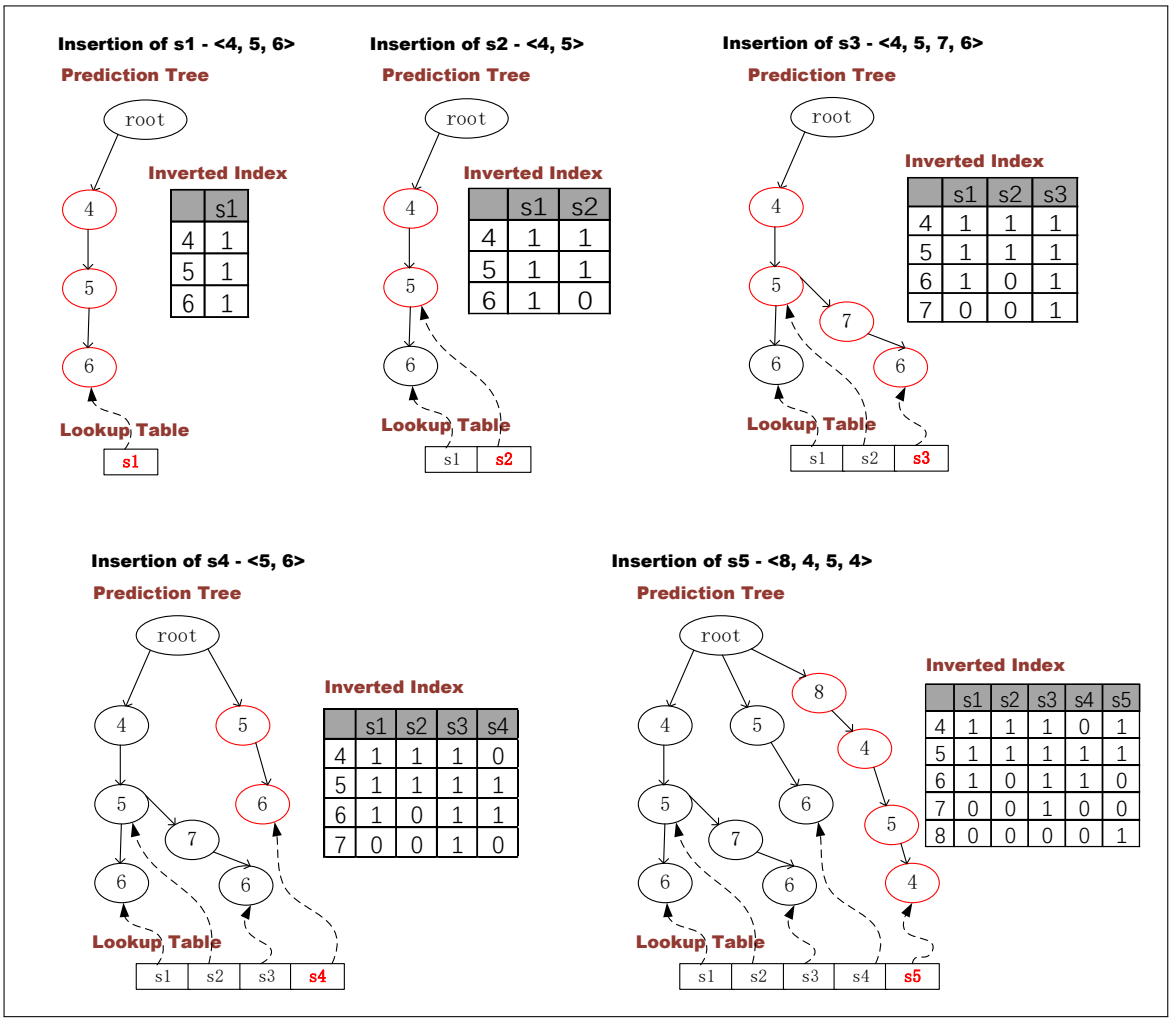


Figure 3.6 An illustration of the construction process of the three data structures in CPT+ (redrawn from [5])

a given symbol or a set of symbols. It is implemented with a *Hash table* where the keys are each unique symbol that appears during the training process and each key maps to a corresponding *bitset* [4]. The bitset of a symbol contains N bits i.e., N training sequences and the s-th bit is set to 1 indicating the presence of the symbol in the s-th sequence; 0 otherwise. The example II in Figure 3.6 clearly illustrates what it looks like after construction. The construction of II requires an average time of $O(N)$ similar to PT and takes $(N + b) \times u$ bytes where u is the number of unique symbols i.e., keys and b is the size of a symbol in bytes [5].

- *Lookup Table* The LT is an associative array that allows to access any training sequence in the PT in constant time [4]. As shown in Figure 3.6, the dashed arrows show that for each sequence ID in the LT, it connects to the last node of the sequence in the PT, hence providing an efficient way to retrieve sequences from the PT using their IDs [5]. Once a sequence is successfully inserted, the LT is updated accordingly. Thus, the PT and LT together represent the training dataset without any loss. Same with the PT, the LT takes $O(N)$ to construct and $(b + p) \times N$ memory space where b is the size of a symbol in bytes and p is the size of a pointer in bytes [5].

3.3.2 Prediction

With a well-trained CPT model, one is able to use the model to perform sequence prediction. This is generally done by the following three steps:

- *Finding similar sequences* Given a sequence S, to predict the next symbol of S, the CPT model needs to find all the sequences that are similar to S. More formally, for a sequence S of n symbols:

$$S = \langle s_1, s_2, ..., s_n \rangle,$$

the suffix of S of size $x \in [1, n]$ is defined as:

$$S^x = \langle s_{n-x+1}, s_{n-x+2}, ..., s_n \rangle.$$

Sequences similar to S are those that contain all the symbols in S^x in any order and in any position, and they are used to predict the next symbol of S [5]. The Inverted Index constructed in the training process is a useful data structure to find similar sequences to S. This can easily be done by performing the intersection of the bitset of each symbol in S^x. For example, if the last $x = 2$ symbols of S are

considered for similarity comparison and symbols $\langle 4, 5\rangle$ are in subsequence S^2, the similar sequences in Figure 3.6 are s_1, s_2, s_3 and s_5 because the intersections of these sequences are all true.

- *Extracting consequent* For each similar sequence of S, the rest of the sequence, excluding the common symbols, is defined as the *consequent* with respect to S. It starts from the symbol after the last one that is identical to S till the end of U. Formally, let $U = \langle u_1, u_2, ..., u_m\rangle$ be a similar sequence to S. The consequent of U is the longest subsequence:

$$U_{consequent} = \langle u_{v+1}, u_{v+2}, ..., u_m\rangle \ such\ that \bigcup_{k=1}^{v}\{u_k\} \subseteq S^x \ and \ 1 \leq v \leq m$$

  Therefore, for the same example above, the consequent of s_1, s_2, s_3 and s_5 are $\langle 6\rangle$, $\langle null\rangle$, $\langle 6, 7\rangle$ and $\langle 8\rangle$, respectively.

- *Constructing Count Table* The unique symbols of all the consequents of the similar sequences to S are then stored in a data structure called *Count Table* (CT), which is defined as a hash table with *symbols* as keys and *scores* as associated values [5]. Thus, for an individual prediction task, there will be a unique CT that stores a list of potential candidate symbols as well as their scores and the symbol with the highest score will be returned as the prediction. The primary measure for scoring these candidate symbols is defined as the *support* that is the number of times a particular symbol appears in similar sequences of S. In cases where two candidate symbols have equal support, their *confidence* is used for prediction, which is defined as the support of a symbol divided by the total number of training sequences that contain this symbol. For the example that has been considered earlier, symbol $\langle 6\rangle$ has the highest support (Figure 3.6), hence the prediction.

3.4 Reinforcement Learning

The vehicular environment is rather dynamic and time-varying. This poses challenges to classic machine learning models such as the neural network model, LSTM, to perform mobility prediction because the requirement for the training phase makes it challenging to adapt to a time-varying environment. Therefore, it is meaningful to seek an online learning method to address the mobility prediction problem in this thesis.

Reinforcement learning (RL) is a machine learning technique that aims to establish solutions to decision-making problems, where a *learning agent* in RL problems learns *policy* for its *actions* under different *states* [85, 86]. The key feature of RL is that it does not require any prior knowledge of the environment, which enables the potential to achieve full self-organisation and high adaptability in cognitive mobile networks [87]. This feature makes RL a suitable approach for online learning. More importantly, this also makes RL a feasible solution to modelling an arbitrary vehicular environment where an accurate analytical model is often unfeasible to build.

The nature of the RL technique also indicates its potential to be used in the problem of mobility prediction i.e., the next RSU prediction in this thesis. An RSU in a vehicular network can act as a learning agent and its neighbouring RSUs are its actions. Thus, predicting the next RSU is essentially a decision-making task for the agent RSU, where it needs to learn the policy for its neighbours such that the best decision about the next RSU is made, equivalent to a correct prediction. As a special instance of RL, multi-armed bandit (MAB) learning rather than the classic RL technique is chosen to address mobility prediction in Chapters 5 & 6 due to its unique features. However, a detailed review of RL techniques is essential before introducing MAB as it helps understand the rationale behind the choice for MAB learning.

3.4.1 An Overview

The goal of the learning agent in any RL problem is to learn a *policy function* which maps perceived *states* of the environment to *actions* which need to be taken under these states. The whole process of building policy function is often known as the *trial-and-error* approach. During the process, the RL agent interacts with the environment without making any assumptions about the environment model e.g., its structure or property, takes an action under a particular state at a time step, and moves to the next state. The outcome of the action taken helps reinforce the current policy.

Depending on the outcome of the taken action, a numerical *reward* associated with the *state-action* pair is generated for the agent by the *reward function*. The reward value received by an agent assists it to assess the desirability of the choice of action i.e., whether the action taken was good or bad. It is also useful for the agent to reconstruct its current policy accordingly so that a potential action that may lead to a higher reward will be taken in future interactions and ultimately maximise the overall reward value.

Another important term and challenge in RL is the *value function*, also known as *value table* or *Q-function / Q-table* in the case of *Q-learning*. It indicates the expected reward of the agent's actions in the long run. Unlike the reward function that informs the quality of an action taken previously at each timestep, the *value* of a state-action pair represents the total discounted sum of rewards expected to be received over the future, starting from that state. The value function is a crucial part of RL because it allows RL agents to take an action with the highest value rather than an action with the instant highest reward.

Furthermore, any RL agent or algorithm will face the challenge of the famous trade-off between *exploration* and *exploitation*. With the estimated value function or value table, in each state a learning agent always faces two options:

- *Exploiting* its current knowledge: select the *greedy* action that guarantees the best reward among all other known actions
- *Exploring* other possibilities: select the non-greedy action i.e., the one that is known to have a lower reward or another unknown action, because it is also possible for them to be better and become the new greedy action.

Approaches to resolving the exploration-exploitation dilemma in RL problems are plenty such as ϵ-greedy [24], upper-confidence bound algorithm [24], Thompson sampling [88], etc. The aim of this thesis is not to find out a sophisticated way to balance exploration and exploitation. Therefore, the most straightforward ϵ-greedy is adopted in later chapters.

Depending on the methods used to solve RL problems, a *model* may or may not be required. The model represents the dynamics of the environment as well as their correlation, including state, action, reward, transition probabilities between state-action pairs, etc. A *model-based* RL requires a well-built mathematical model that can demonstrate a precise and complete relationship between the elements of the environment and allow a learning agent to compute a suitable policy [89, 90]. It involves the estimation of the environment model in the form of a *transition probability matrix* (TPM) and a *transition reward matrix* (TRM), and computes a policy from the estimated TPM and TRM by using dynamic programming to solve the Bellman optimality equation [24]. However, one of the drawbacks of the model-based RL approach is its computational complexity which increases exponentially as the environment gets a huge number of states. *Model-free* RL, on the other hand, does not require a model to construct and fit the TPM and TRM, hence more computationally efficient than the model-based RL. The popular Q-learning and multi-armed bandit problem are representatives of this class.

3.4.2 Model-based Reinforcement Learning

The model-based approach solves the RL problem by computing optimal policy with a transition probability matrix (TPM) and transition reward matrix (TRM), as discussed earlier. Both TPM and TRM are the way to estimate the environment model and they are achieved by observing the actions performed by the learning agent and their outcomes. The TPM contains information about the probability of being in a particular state, performing a certain action, and transitioning from one state to another. The role of TRM, on the other hand, is to store the immediate rewards received after performing an action in a state, i.e., after *state-action-state* transition. A common method to establish TPMs and TRMs is through counting [89], which is also referred to as the maximum likelihood model estimation in [90]. With the developed TPMs and TRMs by the estimation method, an optimal policy is obtained by dynamic programming (DP) by solving the Bellman optimality equation below recursively:

$$Q^*(s,a) = \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right] \tag{3.1}$$

where $Q^*(s,a)$ is the action value representing the long-term cumulative reward by taking action a at state s, $P(s,a,s')$ is the value from TPM representing the probability of transitioning to state s' by performing action a under state s, $R(s,a,s')$ is from TRM representing the expected immediate reward when transitioning to state s' from s by taking action a, $\gamma \in [0,1]$ is the weighting factor related tot the importance of future rewards with respect to the immediate reward. $\max_{a'} Q^*(s',a')$ is the greedy action in state s' that is derived by the *greedy* policy below:

$$\pi(s) = \arg\max_{a} Q^*(s,a) \tag{3.2}$$

Explicitly building up TPMs and TRMs in the model-based RL approach can be computationally inefficient, especially in the context of a dynamically changing environment which is highly relevant to this thesis where the studied environment is a

vehicular network. In addition, for the purpose of online learning and decision-making problem in an arbitrary mobile environment, the model-based approach may not be as flexible as the model-free method that will be discussed shortly.

3.4.3 Model-free Reinforcement Learning

In contrast to model-based RL, model-free RL [86] is relatively more efficient in terms of computational complexity because it omits the intermediate procedure of constructing TPMs and TRMs and $Q^*(s, a)$ is estimated from received rewards directly. Since the environment estimation no longer needs the construction of TPMs and TRMs, the values of the state-action pairs are directly estimated and stored in the value table and a full DP algorithm is also not required for policy derivation. All these features make model-free RL a more popular solution to RL problems.

One of the most popular model-free RL algorithms is Q-learning proposed by Watkins [91]. It has been widely used in mobile networks domain such as the Dynamic Spectrum Access problem [22] and caching problem in vehicular networks [19]. In Q-learning, the learning agent keeps updating its Q-table with the reward received by taking an action with the following formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{3.3}$$

where s' is the next state of s by taking action a, a' is the action that could be taken in state s', $Q(s, a)$ is the Q-value of the current state-action pair, $\max_{a'} Q(s', a')$ is the maximum Q-value in state s', $\alpha, \gamma \in [0, 1]$ is the learning rate and discount factor, respectively, and r is the reward awarded to the agent for action a.

According to Equation (3.3), Q-learning is an *off-policy* approach because the agent assumes that a greedy policy rather than its current policy is followed in the next state. Thus, the Q-values are updated using the greedy action a' of the next state s'. It is not experimentation-sensitive [92]. In contrast, there are *on-policy* learning algorithms such

as *State–Action–Reward–State–Action* (SARSA) [93]. The action-value updating rule of SARSA is shown in Equation (3.4). The difference from an off-policy algorithm is that the agent in on-policy algorithm uses the value of the action a' of the next state s' that is selected using the current policy. In other words, the action a' is what actually happens at the next state s' so on-policy algorithm is experimentation-sensitive.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a)) \tag{3.4}$$

Both approaches have been widely applied in various fields in academia and industry. However, they can be clumsy and inefficient in some RL problems where the environment does not have to be represented by states. The agents then become *stateless* or *single-state* (these two terms will be used interchangeably in the thesis), which means that by taking an action, no state transition happens for the agents. The objective of the learning algorithm then becomes to estimate the expectation of a single reward for each available action to the learning agent:

$$Q(a) = E[r_t] \tag{3.5}$$

where $Q(a)$ is the Q-value of action a and $E[r_t]$ is the expected immediate reward that the learning agent would receive after taking action a at timestamp t. Proposed by Claus and Boutilier [94], the Q-value update equation for *stateless* Q-learning is simplified as:

$$Q(a) \leftarrow (1-\alpha)Q(a) + \alpha r \tag{3.6}$$

One advantage of modelling the RL environment as stateless and applying algorithms such as stateless Q-learning instead of the classical counterparts is that it significantly reduces the number of Q-values required to be estimated by the agent. This feature would increase the adaptability of cognitive wireless networks, in particular vehicular networks which are highly dynamic in topology and mobility. Another advantage is the potential reduction in computational and time complexity which could be brought

by the simplicity of the value function in Equation (3.6) as opposed to Equation (3.3) and (3.4) in Q-learning and SARSA. Again, this would benefit cognitive devices such as RSUs because it takes them less time to learn appropriate proactive caching policies in a new or dynamically changing vehicular environment. Therefore, the proactive caching problem in Chapters 5 & 6 is formulated as a stateless RL problem and is solved by the classical *multi-armed bandit* learning, which is a single-state learning problem [24] and will be discussed in detail in the next section.

3.5 Multi-armed Bandit Problem

The multi-armed bandit (MAB) problem, also known as k-armed bandit problem, is a special instance of reinforcement learning. It is also in essence a decision-making problem and can therefore be applied to the mobility prediction for the next RSU in this thesis, as described at the beginning of Section 3.4. However, different from a traditional or a full RL problem where a learning agent may have multiple states associated with the environment (e.g., positions in a game), it only has a single state in MAB problem [24] (i.e., no state transition). From this perspective, MAB is essentially identical to *stateless Q-Learning* [94] and can also be treated as a model-free reinforcement learning technique.

A well-known scenario of the bandit problem (shown in Figure 3.7) is where a gambler in a casino sits in front of a slot machine with one or multiple arms (referred to as a *one*-arm bandit and k-armed bandit respectively) and tries to get payoffs by pulling the arm(s). The ultimate goal of the gambler is to achieve the highest cumulative rewards through learning the inherent reward pattern of each arm by pulling the arms and gradually concentrating on the best arm. The inherent reward pattern of each arm can be seen as a stochastic process where its reward is generated with certain probabilities. During the learning process, the gambler will face the *exploration-exploitation* dilemma [88]: where the gambler tries out the potential arms that may

return high payoffs (exploration) or pulls the arm that has yielded the highest reward from the past experiments (exploitation). This is a non-trivial process and carefully balancing exploration and exploitation is crucial in MAB problems.

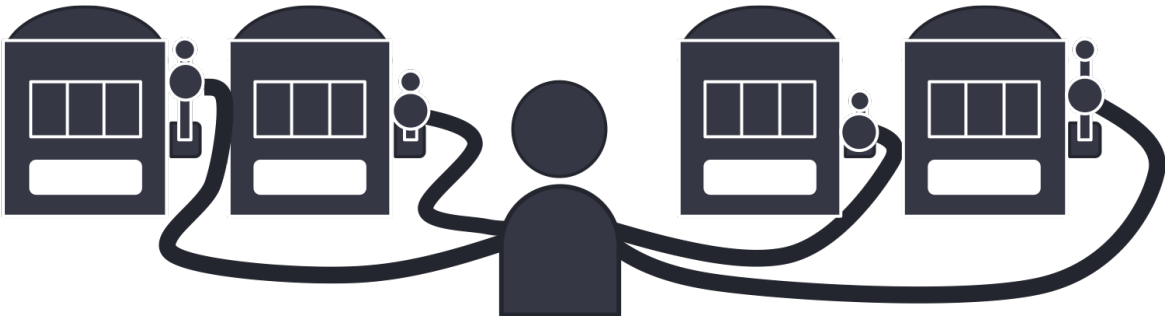


Figure 3.7 Multi-armed bandit problem[2]

While MAB has been widely used and proven to be effective in areas such as ad placement, computer game-playing, etc., its application in vehicular networks seems to be limited. Dai *et al.* [95] proposed a multi-armed bandit learning algorithm called Utility-table based Learning to solve the distributed task assignment problem in a MEC-empowered vehicular network. The work in [96] focused on task caching problems in the edge cloud. The authors proposed an intelligent task caching algorithm based on a multi-armed bandit algorithm and evaluated its benefits in task latency performance. Authors of [97] discussed the potential of using a MAB problem in future 5G small-cell networks as well as its applications and future research directions. A detailed example of using a MAB model for energy-efficient small cell activation in 5G networks has been provided in [97]. Xu *et al.* [98] investigated collaborative caching problems in small-cell networks by learning the cache strategies directly at small base stations online by utilising multi-agent MAB.

In the following a canonical example of MAB - the Bernoulli bandit problem and the contextual multi-armed bandit problem will be discussed as they are closely related to the proposed learning algorithms for proactive caching in Chapters 5 & 6.

[2]https://www.kaggle.com/code/marlesson/what-is-multi-armed-bandits/notebook

3.5.1 Bernoulli Multi-armed Bandit

In general, a MAB problem can be formally given as a tuple [99]: $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A} = \{a_1, a_2, ..., a_k\}$ is a set of k actions (i.e., arms) and $\mathcal{R} = \{\theta_1, \theta_2, ..., \theta_k\}$ associates action a_i with its reward probability distribution defined by θ_i. Consider a k-armed bandit problem $\langle \mathcal{A}, \mathcal{R} \rangle$. The agent takes actions from action set $\mathcal{A}$ and any action played will generate a success (reward 1) or failure (reward 0). Action $a \in \mathcal{A}$ produces a success with probability $\theta \in \mathcal{R}$. In other words, for an action a, a reward $r = 1$ is produced with probability θ and $r = 0$ with probability $1 - \theta$. In this case, θ can be viewed as the expected reward of taking action a, is unknown to the agent, and is invariant in a *stationary* MAB problem. One natural way to estimate such θ is to use *sample-average* method [24] by averaging the rewards actually received. The estimated value of θ at time step t can be denoted as:

$$\begin{aligned} Q_t(a) &= \frac{\text{sum of rewards when } a \text{ taken prior to t}}{\text{number of times } a \text{ taken prior to t}} \\ &= \frac{\sum_{i=1}^{t-1} r_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \end{aligned} \tag{3.7}$$

where A_i is the action taken at time i, $\mathbb{1}_{condition}$ is 1 if *condition* is true and 0 if not, and $r_i = \{1, 0\}$ is the reward of *i-th* selection of action a. According to the law of large numbers, Equation (5.1) converges to θ as the denominator tends to infinity. A more intuitive way to illustrate this is through the probability density function of $Beta(\alpha = successes, \beta = failures)$ distribution as shown in Figure 3.8. Consider a 95% confidence interval, in the late stage of the sample-average process after 1000 trials with 500 successes and 500 failures, the range that captures the true probability θ is [0.469, 0.531], i.e., $P(0.469 < \theta < 0.531) = 0.95$. However, the intermediate stage with 100 trials (50 successes and 50 failures) returns a much wider range of [0.403, 0.597] for the same 95% confidence interval and the initial stage with only 10 trials gives an even wider range of [0.212, 0.788]. Thus, the more trials, the more certain one can be about the approximation to the true probability θ. By taking the proper action

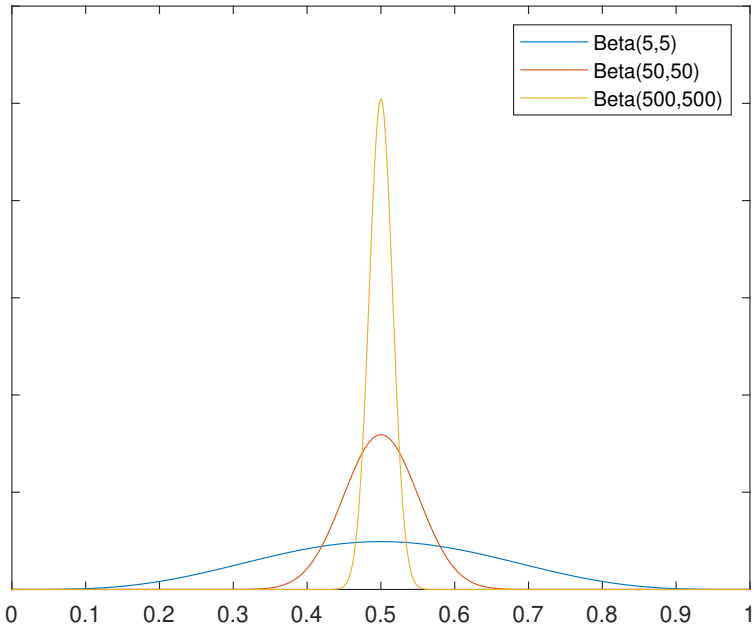


Figure 3.8 An example of the sample-average process shown with Beta distribution

with the associated action-selection strategy (e.g., ϵ-greedy), it is also to maximise the cumulative rewards $\sum_{t=0}^{T} r^t$ where T is the given time horizon.

3.5.2 Contextual Multi-armed Bandit

As an extension of the general MAB problem, the contextual multi-armed bandit (cMAB) problem associates actions with side information or *context* [100]. To some extent, context is similar to *state* as introduced in traditional reinforcement learning. On the one hand, context is also information perceived from the environment and if the action set is associated with the context, the learning agent is learning a policy that allows it to take proper action under different contexts. On the other hand, the main effect of context in cMAB is to help the agent to distinguish one bandit problem from another [24] so that better performance can be achieved. The action taken at a particular context s only affects the immediate reward and makes no difference to other context s' as well as their rewards. Because of this, the learning agent in cMAB

is still stateless. This is very different from a classic RL. Therefore, it can be seen as an intermediate between the MAB problem and the full RL problem.

A cMAB problem can be formally given as a tuple: $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$, where $\mathcal{A} = \{a_1, a_2, ..., a_k\}$ is a set of k actions (i.e., arms), $\mathcal{S} = \{s_1, s_2, ..., s_j\}$ is a set of j contexts, and $\mathcal{R} = \{\theta_{1-1}, \theta_{2-1}, ..., \theta_{j-k}\}$ associates action a_k and context s_j with its reward probability distribution defined by θ_{j-k}. This is formally formulated as follows:

- Consider a cMAB problem $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$. The aim of any agent in the cMAB problem is to learn a policy that maps contexts to actions, that is, $\pi(a \in \mathcal{A} \mid s \in \mathcal{S})$. Another viewpoint is that they now become multiple independent MAB tasks associated with contexts, and the agent aims to learn the best policy under these various contexts. Every time an agent is assigned a MAB task (possibly with a certain probability), it will observe context, take the action by looking at the current context, and eventually learn the best action. The agent takes an action a_k from its action set $\mathcal{A}$ under context $s_j \in \mathcal{S}$ and this will generate a success (reward 1) or failure (reward 0). The action $a_k \in \mathcal{A}$ produces a success with probability $\theta_{j-k} \in \mathcal{R}$. In other words, for an action a_k reward $r = 1$ is produced with probability θ_{j-k} and $r = 0$ with probability $1 - \theta_{j-k}$. In this case, θ_{j-k} can be seen as the expected reward of taking action a_k at situation s_j and is unknown to the agent. The estimated value of θ_{j-k} at time step t is denoted as

$$\begin{aligned} Q_t(a_k \mid s_j) &= \frac{\text{sum of rewards when } a_k \text{ is taken under } s_j \text{ prior to t}}{\text{total number of times } a_i \text{ is taken under } s_j \text{ prior to t}} \\ &= \frac{\sum_{i=1}^{t-1} r_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \end{aligned} \tag{3.8}$$

  where A_i is the action taken at time i, $\mathbb{1}_{condition}$ is 1 if *condition* is true and 0 if not, and $r_i = \{1, 0\}$ is the reward of i-th selection of action a. The cumulative rewards are now to be maximised across $\mathcal{S}$ over a certain amount of time T.

General formula of action value-updating and selection

The sample-average approximation method for action-value estimation shown in Equations (3.7) and (3.8) can have a more compact representation with incremental implementation [24]. For simplicity, the term $Q(a)$ is used to denote the Q-value of action a regardless of context s. For action a which has been selected for n times, the estimated value is:

$$\begin{aligned} Q_{n+1} &= \frac{1}{n} \cdot \sum_{i=1}^{n} r_i \\ &= \frac{1}{n} \left(r_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} r_i \right) \\ &= \frac{1}{n} \left(r_n + (n-1) Q_n \right) \\ &= Q_n + \frac{1}{n} \left(r_n - Q_n \right) \end{aligned} \tag{3.9}$$

An important parameter in the incremental value updating rule of Equation (3.9) is $\frac{1}{n}$, the *step-size*. As can be noted from the Equation (3.9), this step-size declines as n grows. In fact, this is fairly effective in a ***stationary*** bandit problem where the reward probabilities (i.e., θ) remain unchanged over time. Vehicular networks, however, are dynamic environments with varying traffic densities and may result in a ***non-stationary*** bandit problem. Therefore, recent rewards should be given more weight when updating action values. This is often achieved using a constant step-size denoted with $\alpha \in [0, 1]$ and Equation (3.9) therefore becomes:

$$Q_{n+1} = Q_n + \alpha (r_n - Q_n) \tag{3.10}$$

A general recursive form of Equation (3.10) is represented as:

$$Q(a) \leftarrow (1 - \alpha) Q(a) + \alpha r \tag{3.11}$$

where $Q(a)$ is the quality value of action a, named **Q**-value as in **Q**-learning, r is the reward associated with the most recent trial and is determined by a reward function, and $\alpha \in [0, 1]$ is the step-size or *learning rate* . Notably, Equation (3.11) is identical to the value function in Equation (3.6) of stateless Q-learning, one reason of which is because of their stateless nature.

In addition, the approaches to resolve the exploration-exploitation dilemma in MAB problems are plenty such as ϵ-greedy, upper-confidence bound algorithm, Thompson sampling [88], etc. These approaches are often referred to as *action-selection* strategies that are used to balance exploration and exploitation. ϵ-greedy method is a straightforward and widely used action-selection method and has been applied to the problem in later chapters. Given the estimated action values $Q(a)$ of actions in $\mathcal{A}$, the ϵ-greedy method is used to make a selection: the best action is selected with a probability of $1 - \epsilon$; otherwise, actions will be selected randomly with a small probability ϵ regardless of their action values.

$$A_t = \begin{cases} \arg\max_a Q(a), & 1 - \epsilon \\ random, & \epsilon \end{cases} \tag{3.12}$$

3.6 Conclusion

In conclusion, this chapter has discussed the underpinning techniques of the research work in this thesis. First, the SUMO traffic simulator is introduced with the aim of presenting the general process of generating traffic data for this work. Second, the theory of the sequence prediction algorithm - CPT+, which is the underlying algorithm of the work in Chapter 4, is presented. Finally, before introducing the technical details of the multi-armed bandit (MAB) problem, the principle learning technique for the work in Chapters 5 & 6, the chapter has also presented a comprehensive overview of reinforcement learning (RL) in general with the aim of highlighting the relationship and difference between MAB and RL.

Chapter 4

Sequence-Prediction based Proactive Caching in Vehicular Networks

4.1 Introduction

As described in the previous chapters, predicting the next RSU can be an effective approach for mobility prediction. The motivation of the work in this chapter is to act as a pioneer to explore the effectiveness and feasibility of the approach. Specifically, the work models RSUs in a vehicle's trajectory as a sequence of symbols, and predicting the next RSU is like predicting the next symbol of the sequence. To this end, this work decides to apply the sequence prediction algorithm - Compact Prediction Tree plus (CPT+) [4], which has been introduced in depth in Section 3.3. Inspired by the features and advantages of sequence prediction, this chapter aims to investigate mobility-aware proactive caching by proposing a Sequence-Prediction based Proactive Caching system, dubbed SPPC. SPPC models RSUs that vehicles have connected to as sequences of symbols and uses CPT+ as the main sequence prediction algorithm to predict the next potential RSU, through sufficient training with historical data. The variant of the SPPC system proposed in this chapter will also serve as an important baseline system in the following chapters.

The remaining chapter is as follows. In Section 4.2, the fundamental vehicular network architecture for this thesis will be introduced. In addition to the architecture, Section 4.2 also introduces caching mode as well as the proactive caching strategy which will be used throughout the thesis. Section 4.3 introduces the design of the SPPC system. Section 4.4 introduces the simulation as well as the relevant performance results, and Section 4.5 concludes the chapter.

4.2 MEC-Enabled Vehicular Network Architecture

The vehicular network architecture discussed in this section is the fundamental architecture for Chapters 4, 5 and 6, although it may differ in some details due to the differences in the applied proactive caching algorithm, which will be discussed in detail where it is used. These differences will be clearly specified in relevant chapters. The vehicular

network considered in this thesis is deployed with RSUs that are MEC-enabled, as depicted in Fig. 4.1. The RSUs are capable of edge computing and caching with MEC servers. Along with the computing units, they are intelligent to learn and predict the next possible RSU a vehicular user may connect to and the caching units enables them to pre-caching content when a pre-caching request is received from other RSUs. Vehicular users frequently request content from RSUs after they enter the network. Despite the equipped MEC servers, computing resource consumption and content replacement techniques are out of the scope of this thesis.

Consider a vehicular network $\mathcal{G}$ in an urban area with M RSUs in a set $\mathcal{M} = \{m_1, m_2, ..., m_M\}$. There are residential areas and workplace areas in $\mathcal{G}$ where L vehicles in the set $\mathcal{V} = \{v_1, v_2, ..., v_L\}$ depart and arrive on a daily basis. An RSU $m_i \in \mathcal{M}$ has neighbouring RSUs and it predicts the next RSU which is normally one of its neighbours. In addition, a multi-functional central node is available to serve and coordinate RSUs in various ways. For the sequence prediction-based system in this chapter, its main function is to provide RSUs with collected training data. For the work in Chapters 5 and 6, this node is important because its main responsibility is to observe the result of a previous prediction and feedback a reward to a prior RSU so that the RSU can refine its learning policies. Furthermore, a content database $\mathcal{C} = \{c_1, c_2, ..., c_K\}$ exists in the Content Provider that stores K types of content with various sizes. Each type of content is represented by $\{f_1, f_2, ..., f_{c_K}\}$ fragments each of which is of a constant size F_c. The content is requested by the vehicles in a uniform way. In other words, there is no preference for content that a vehicle decides to request. The content considered throughout the thesis is abstracted type and not restricted to a particular kind. The key property of the content here is fragment-based and in this way, the proactive caching strategy introduced later can cache the required fragments. Such abstraction is practical in reality such as video streaming with Dynamic Adaptive Streaming over HTTP [101], downloading large high-resolution regional maps, or simply transmitting a large dataset. This will be covered in greater detail in Section 4.4.4.

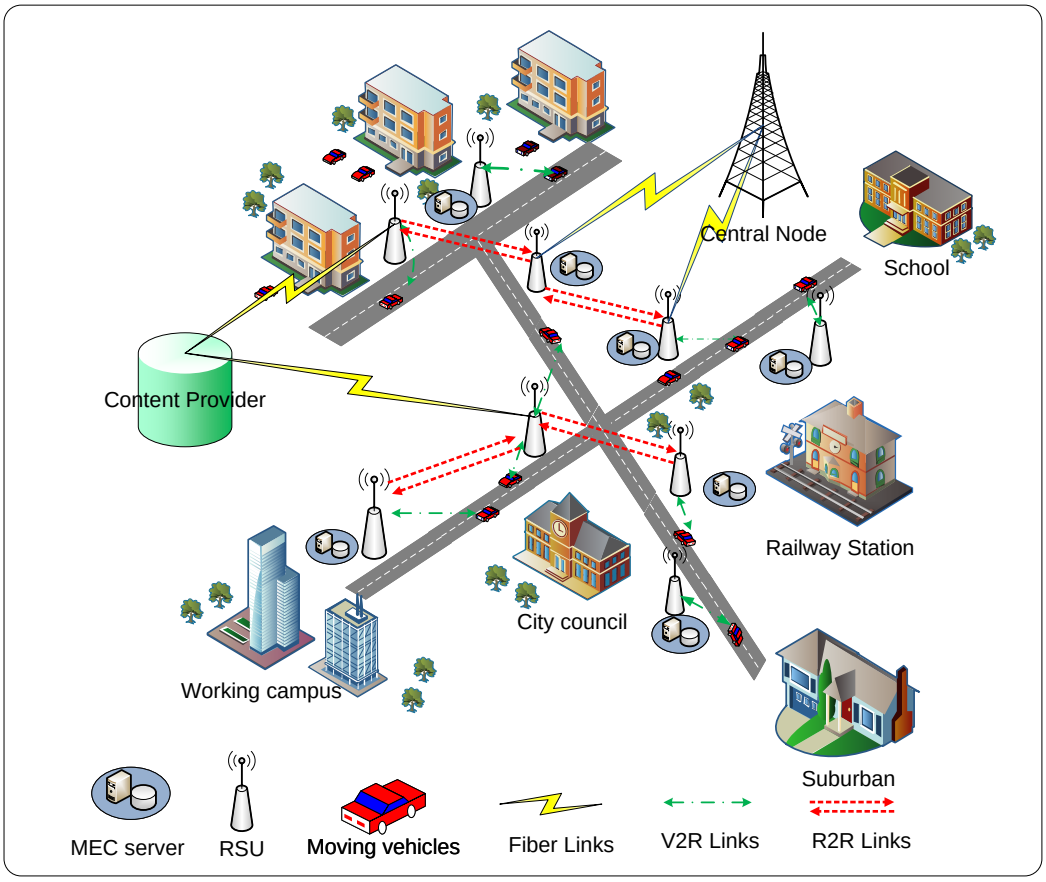


Figure 4.1 Architecture of MEC-enabled vehicular network

The communication model implemented in this thesis only characterises some basic features of transmission because the goal of the work is to anticipate where to cache precisely. Therefore, the following assumptions are made:

- A vehicle connects to the geographically closest RSU
- The underlying physical and MAC layers' problems e.g., packet loss, interference, and re-transmissions are not considered in vehicular communications and thus the transmission rate e is a constant
- The dwell time of the vehicles in the coverage area of an RSU is extracted from the test trace being simulated and is known so that the number of content fragments can be derived

- A vehicle will not request new content until it finishes consuming the current one; when handover occurs, the vehicle continues its unfinished transmission

4.2.1 Caching Mode

The caching nodes i.e., RSUs in vehicular networks may have two caching modes, *reactive caching mode* and *proactive caching mode*:

- ***Reactive caching:*** RSUs store the content they have transmitted to the vehicles in a reactive way, i.e. after the content has been first requested and used it will be stored in the cache.
- ***Proactive caching:*** RSUs store the content in a proactive way by prediction algorithms, in anticipation that it might be used at some point in the future.

For different purposes, a vehicular system may implement a single caching mode or a mixture of both. For example, the SPPC system investigated later in this chapter implements a hybrid mode where RSU nodes not only store the content reactively but also proactively as requested. However, proactive caching systems in Chapter 5 & 6 are pure proactive systems. Whether a system applies a hybrid caching mode or a single mode will be explicitly pointed out throughout the thesis.

4.2.2 Proactive Caching Strategy

There is a general proactive content caching strategy in all the proactive caching systems considered in this thesis. A representative proactive caching procedure can be described as follows. After a vehicle $v_i \in \mathcal{V}$ connects to an RSU $m_i \in \mathcal{M}$, m_i uses the prediction algorithm to predict the next RSU that the vehicle is likely to access next, e.g., $m_j \in \mathcal{M}$. Now assume v_i requests a new transmission for content $c_K \in \mathcal{C}$ from m_i. If m_i calculates that v_i cannot complete this transmission within the dwelling time, then m_i sends the *proactive caching request* message to m_j to ask it to

perform proactive caching on c_K from fragment No. e.g., f_r. The number of fragments, F_R, needed to be cached depends on T_R the remaining dwelling time of vehicle v_i in m_i, i.e., the interval between content request is sent and handover happens, and the backhaul link rate ω. Thus, $F_R = \min(\left\lfloor \frac{T_R \times \omega}{F_c} \right\rfloor, f_{c_K} - f_r)$. Next, v_i hands over to a new RSU. If this new RSU happens to be m_j, then this is a correct prediction and the pre-cached content is hit. In this case, m_j satisfies the remainder of v_i's previous transmission by its cache instead of having to request that from the content provider, hence realising seamless transmission and reducing network delay. Otherwise, the new RSU has to finish the remaining transmission through the content provider via the backhaul network. A transmission delay μ is thereby introduced via $\frac{F_R \times F_c}{\omega}$.

Notably, in a system with a mixed caching mode such as SPPC in the next section, the RSUs may have already stored the content that the vehicle is requesting because of previous transmissions. Additionally, the RSUs in such a system will also store the content they have been asked to proactively cache (because they are predicted to be the next RSU for a vehicle), even though they are the "wrong" prediction i.e., cache miss. Therefore, the above F_R could be smaller than $T_R \times \omega$, depending on the status of local caches. On the other hand, in pure proactive caching systems in Chapter 5 & 6, if cache miss happens, the pre-cached content for a particular vehicle is abandoned by the RSUs, which complies with the feature of pure proactive caching. Besides, there may be further prediction feedback to m_i with the assistance of the Central Node after a handover happens depending on the underlying prediction algorithms, which will be clarified in the appropriate places throughout the thesis.

4.3 Sequence-Prediction based Proactive Caching System

In the proposed SPPC system, the fundamental sequence prediction algorithm is CPT+ introduced in Section 3.3. The SPPC system relies on CPT+ to train an offline tree

model using sequences of RSUs and make predictions of the next RSU with the properly trained model. Therefore, the SPPC system is an offline learning system. The topic in this section is to introduce how the SPPC system is designed and solves the proactive caching problem.

SPPC system design

The inspiration for modelling the proactive caching problem as a sequence prediction problem comes from the fact that the RSUs in the vehicular network that a vehicle passes through during its journey can be seen as a sequence where the RSU ids are the symbols. Therefore, predicting the next RSU that a vehicle is likely to connect to for a proactive caching task is exactly like predicting the next symbol of a sequence. Thus, CPT+ is a good choice for this purpose.

The SPPC system integrates CPT+ prediction module, which is available in the SPMF library[102], into the discrete-event driven network simulation module (will be shown in the next section). Since the CPT+ module is available in JAVA language [103] and the network simulation module is implemented in MATLAB [72], the plug-in for CPT+ is done by implementing a JAVA interface in MATLAB, which can be referred to the MATLAB documentation regarding external interfaces[1]. The prediction functionality of the SPPC system works in the same way as CPT+ algorithm and can be described as follows.

- *Training data* The vehicles' training data is the historical sequences of RSUs that they have connected to and is stored in individual files named by vehicle IDs. For instance, the historical sequences of RSUs associated with vehicle 140 are stored in a file named "*140.txt*". Such "*id.txt*" file is later used as training data to train a specific CPT+ model to perform the prediction task for this particular vehicle. Figure 4.2 is an example of a training dataset collected for vehicle 140

[1]https://uk.mathworks.com/help/matlab/matlab_external/accessing-your-own-java-classes.html

and the format of these sequences should comply with the input requirements of the SPMF library. In the text file, each line represents a sequence and each symbol from a sequence is separated by a single space and a "-1". The value "-2" at the end of each line indicates the end of a sequence.

```
 1  11 -1 13 -1 14 -1 15 -1 -2
 2  18 -1 17 -1 16 -1 15 -1 24 -1 -2
 3  18 -1 17 -1 16 -1 15 -1 -2
 4  18 -1 17 -1 16 -1 24 -1 -2
 5  10 -1 11 -1 13 -1 14 -1 15 -1 24 -1 -2
 6  10 -1 11 -1 13 -1 14 -1 15 -1 -2
 7  10 -1 11 -1 13 -1 14 -1 15 -1 -2
 8  18 -1 17 -1 16 -1 15 -1 -2
 9  18 -1 10 -1 18 -1 11 -1 17 -1 16 -1 24 -1 -2
10  10 -1 11 -1 13 -1 14 -1 15 -1 -2
11  11 -1 13 -1 14 -1 15 -1 -2
12  10 -1 11 -1 13 -1 14 -1 15 -1 -2
13  11 -1 18 -1 17 -1 16 -1 15 -1 -2
14  10 -1 11 -1 13 -1 14 -1 15 -1 -2
15  11 -1 13 -1 14 -1 15 -1 -2
16  11 -1 10 -1 11 -1 13 -1 14 -1 15 -1 -2
17  10 -1 11 -1 13 -1 14 -1 15 -1 24 -1 -2
18  11 -1 13 -1 14 -1 15 -1 24 -1 -2
19  11 -1 10 -1 11 -1 13 -1 14 -1 15 -1 -2
20  17 -1 18 -1 17 -1 16 -1 15 -1 -2
21  10 -1 11 -1 13 -1 14 -1 15 -1 -2
22  18 -1 17 -1 16 -1 15 -1 -2
```

Figure 4.2 A screenshot of the training dataset of vehicle 140. This is the dataset format required by CPT+ model. In this example, there are 22 rows and each row represents a complete sequence of RSU IDs of a trip that vehicle 140 has made. "-1" is a separator between two RSU IDs and the end of the sequence is indicated by "-2".

- *Prediction* The prediction process in the actual simulation is straightforward. When an RSU needs to predict the next RSU for a vehicle, it will first extract the training dataset of this vehicle and train a CPT+ model. In the SPPC system, the RSUs that a vehicle has connected to since it enters the network form a sequence. The predicting RSU, after establishing a prediction model with the vehicle's training data, considers all the past RSUs (including itself) of the vehicle as a sequence, denotes here as S_{RSU}. As discussed in 3.3, the similar sequences

to a particular sequence S is based on the suffix of S. The RSUs in the SPPC system use the S_{RSU} as a whole to find those similar RSU sequences and then construct the Count Table. The returned, i.e., the predicted, RSU ID is the one that has the highest score in the Count Table, which is likely to be the one the vehicle connects to next.

In terms of the caching policy, the SPPC system implements both *Reactive caching* and *proactive caching*, as described in Section 4.2.1 and Section 4.2.2. To make this more explicit, the caching strategy of the SPPC system is summarised as follows. In the SPPC system, RSUs are able to cache content and satisfy users directly with caches. They not only are able to cache content reactively but also able to cache proactively and store it regardless of a cache hit or miss. Notably, the size of the local cache in SPPC system is unlimited and this also applies to the systems in Chapter 5 & 6, because the thesis concentrates on proactive caching performance only instead of content replacement strategies that are needed when the cache size is limited. When the current RSU has predicted the next RSU (e.g., RSU_n) for a vehicle, it will transmit a message to RSU_n which will pre-cache the relevant content if the vehicle cannot finish consuming it in the current connection. This is done by calling a function in the actual simulation. The "request" message to RSU_n from the current RSU contains content ID of $c_i \in \mathcal{C}$ and fragment index number f_r which is the starting fragment from which RSU_n should cache c_i. As described in Section 4.2.2, the remaining fragments of content c_i that RSU_n needs to proactively cache depends on $F_R = \min(T_R \times \omega, f_{c_K} - f_r)$ and its local cache status. Assuming that the content fragments for pre-caching are in set F_{pre} and the fragments that are already stored locally are in set C_{cached}, the actual fragments remaining to be cached can be denoted by the following:

$$\{f \mid f \in F_{pre} \text{ and } x \notin C_{cached}\}$$

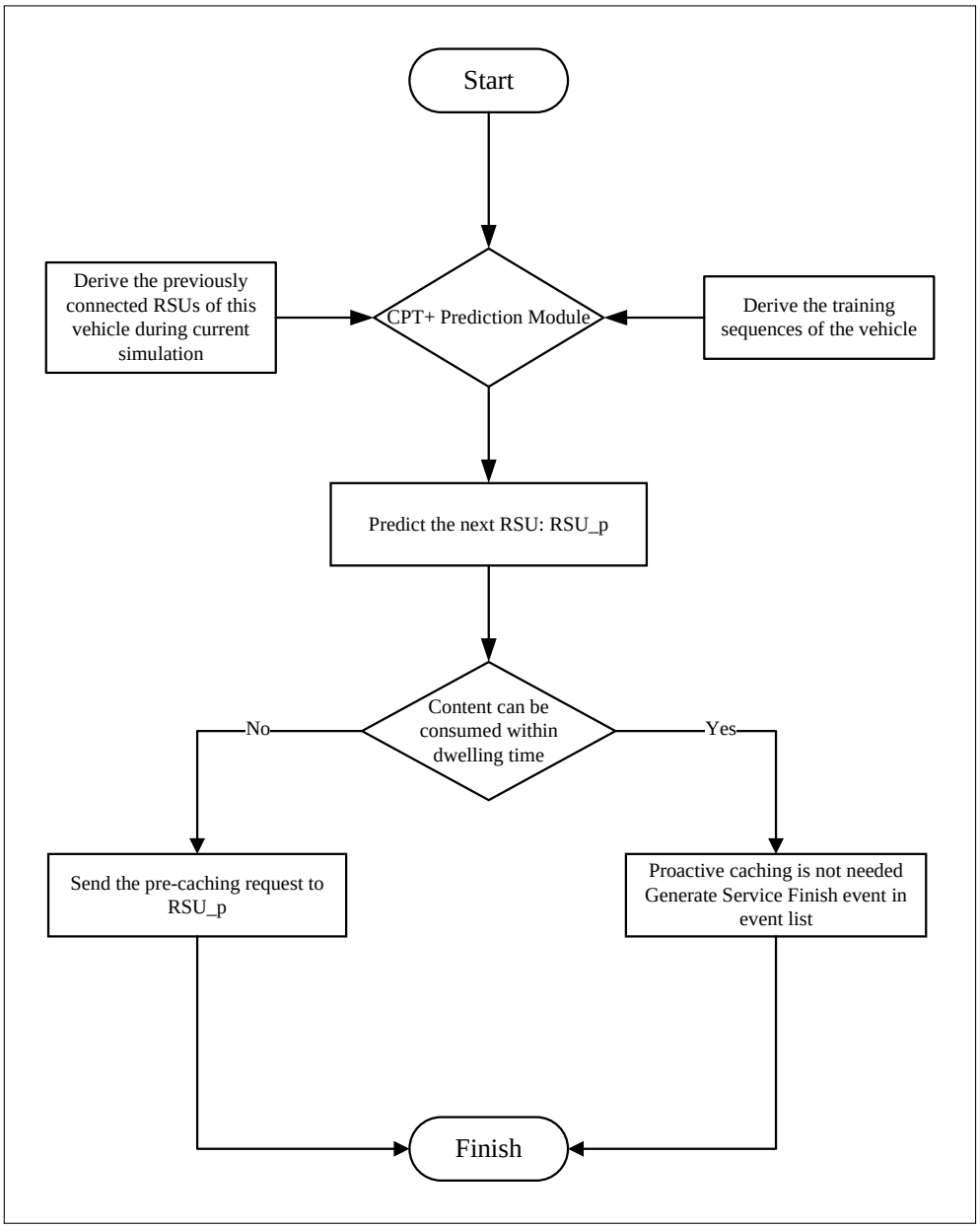


Figure 4.3 Flowchart of the SPPC proactive caching system

This is a general cycle of an RSU in the SPPC system. When a vehicle connects to a new RSU and sends a content request or when a vehicle switches to a new RSU and has unfinished content transmission from an earlier RSU, the RSU needs to build a CPT+ model with the training data of the vehicle and predict the next RSU for this vehicle. *Start* means content transmission request received from a connecting vehicle, to *Finish* when proactive caching is performed or content transmission is finished.

4.4 Simulation and Performance Evaluation

The simulation in this work includes two parts: traffic simulation and network simulation. Vehicle traffic traces are generated by Simulation of Urban MObility (SUMO) [73] and they are processed with a discrete event-driven network simulation program introduced in Section 3.2.2.

4.4.1 Traffic Simulation and Scenario

SUMO is used to simulate a real transportation network discussed in Section 3.2.1. The scenario considered here is the daily commuting routine of people living in a particular urban area. An area in Las Vegas, USA is the primary area, and Manchester, UK acts as a secondary city to generalise the application of the SPPC system to two cities with different road layouts. For both areas shown in Figure 4.4, five traffic assignment zones (TAZs) are defined in SUMO and in total 174 vehicles travel from and to these zones as their origins and destinations. These TAZs are designed to simulate realistic residential and office areas. It is assumed that a TAZ contains both residential and office areas. In order to simulate vehicles with the same daily routine, each vehicle has its own fixed departure and arrival zones. The vehicles emerge at a particular street in the departure TAZ when their trips start and disappear at a particular street in the arrival TAZ when their trips end. Thus, this means that during the simulation, especially in the late stage, there will be vehicles leaving the network. However, each vehicle may have different departure times and lanes (which may result in daily route differences) from test trace to test trace. Again, this is to imitate that people in reality may set off for work at different times, park at various places in a local area, and take slightly different commuting routes, despite having the same workplace (TAZ). The vehicles' routes between two TAZs are defined by the tool *duarouter* and follow the Shortest or Optimal Path Routing rule. They depart at the *maxSpeed* and follow the default Car Following Model to keep the maximum speed which is safe in the sense of

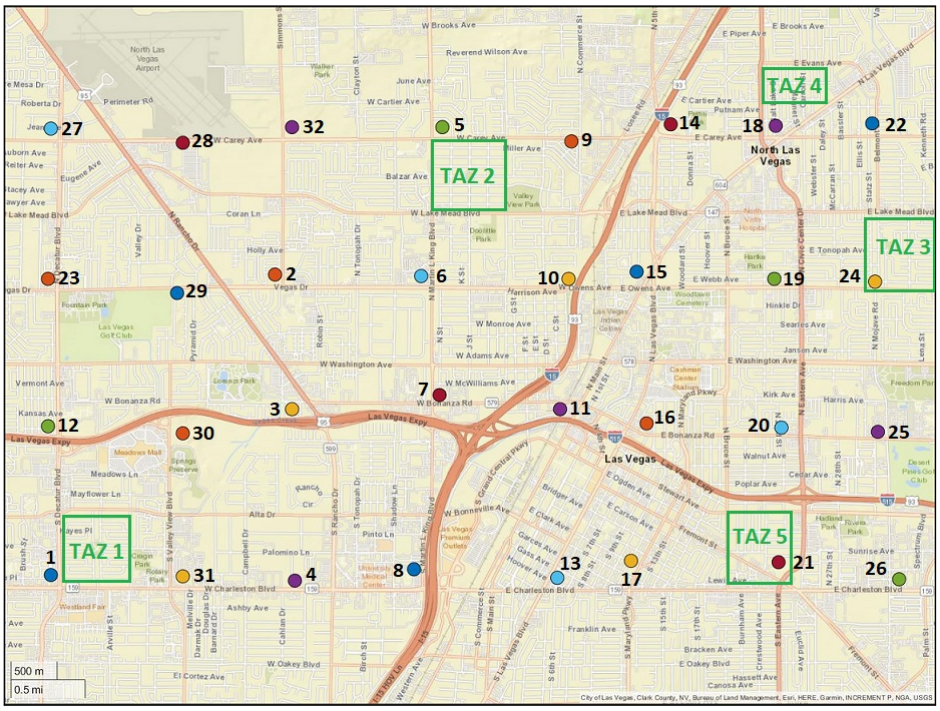
being able to stop in time to avoid a collision. Other road behaviours apply as well such as lane changing, accelerating/decelerating, intersections, etc. Technical details about these settings can be found in SUMO documentation[2].

Two types of simulation traces have been used: training traces and test traces, as required by the underlying CPT+ algorithm. The training traces are used to train the CPT+ model and the test traces are for the actual simulation. 22 training traces are generated in SUMO with the same origin-destination configuration for the 174 vehicles, meaning that these vehicles follow the same routine 22 days of a month. These traces were then processed to extract RSU sequences of the vehicles and these sequences are stored in each vehicle-specific file, *"id.txt"*, which has 22 RSU sequences as training data as the example shown in Figure 4.2. In addition to training data, 30 test traces of these 174 vehicles are then generated to have sufficient tests.

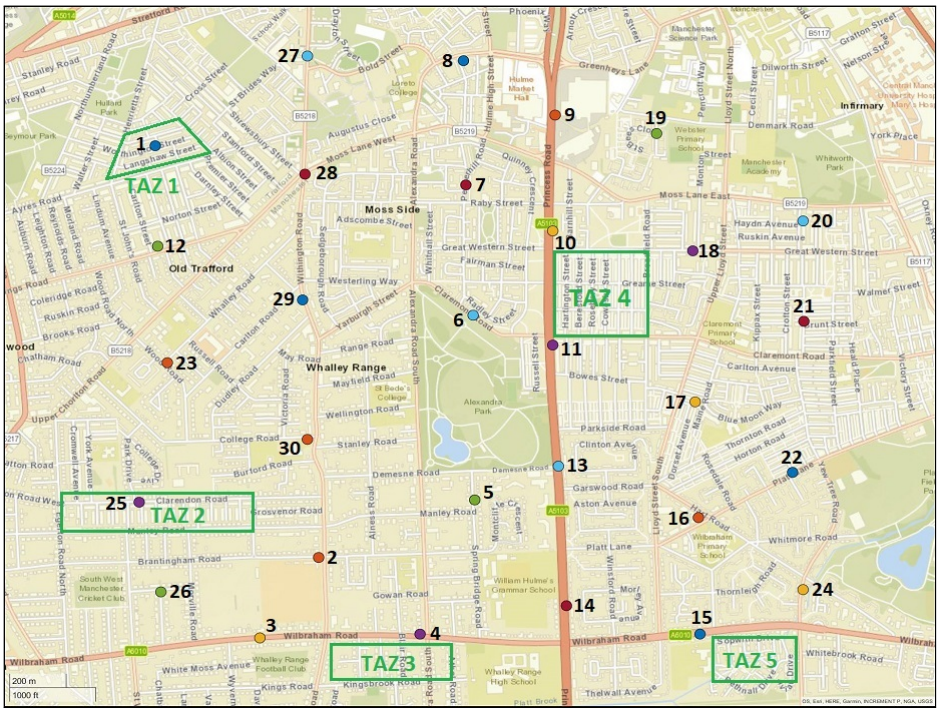
4.4.2 Network Simulation

Network simulation in this chapter follows the discrete event driven simulation (DES) method introduced in Section 3.2.2. In addition to DES-based simulation, *Monte-Carlo simulation* is adopted to evaluate system performance in this chapter. Thus, a complete cycle of the simulation is as follows. Concretely, 7 different content pool sizes, [2 5 10 15 20 25 30], are configured. The simulation for a given proactive caching system e.g., SPPC, and a given content pool of size e.g, 15, 30 test traces are simulated one by one and the content type in each pool is randomly selected from a larger content database for every test. Table 4.1 summarises the relevant parameters. The network parameters such as transmission rate and backhaul link rate, are empirical values and they have no impact on the performance of proactive caching.

[2]https://sumo.dlr.de/docs/



(a) RSU and TAZ distribution in Las Vegas



(b) RSU and TAZ distribution in Manchester

Figure 4.4 RSU and Traffic Assignment Zone (TAZ) distribution in the two urban areas for traffic and network simulation.

Table 4.1 Simulation Parameters

Parameter	Value
No. of Vehicles	174
No. of test traces	30
No. of training dataset	174
No. of training sequences	22 per training dataset
SUMO Simulation Time	1 hour (8:00 - 9:00)
No. of RSUs	32 (Las Vegas) / 30 (Manchester)
Backhaul Link Rate	$\omega = 5Gbps$
Transmission rate	$e = 50Mbps$
Fragment size	$F_c = 100MB$

4.4.3 Performance Evaluation

This section evaluates the performance of the following four systems:

- *SPPC system*: the proactive caching system using the sequence prediction algorithm i.e., CPT+
- *Baseline Proactive Caching System*: the system using the basic proactive caching scheme where the next RSU to pre-cache is selected randomly from the neighbouring RSUs of the current connecting RSU. No vehicular big data is used as training data in this system.
- *non-proactive caching system*: the system that only implements reactive caching.
- *no caching system*: the system that is not equipped with caches.

The figure of merit considered for performance evaluation includes:

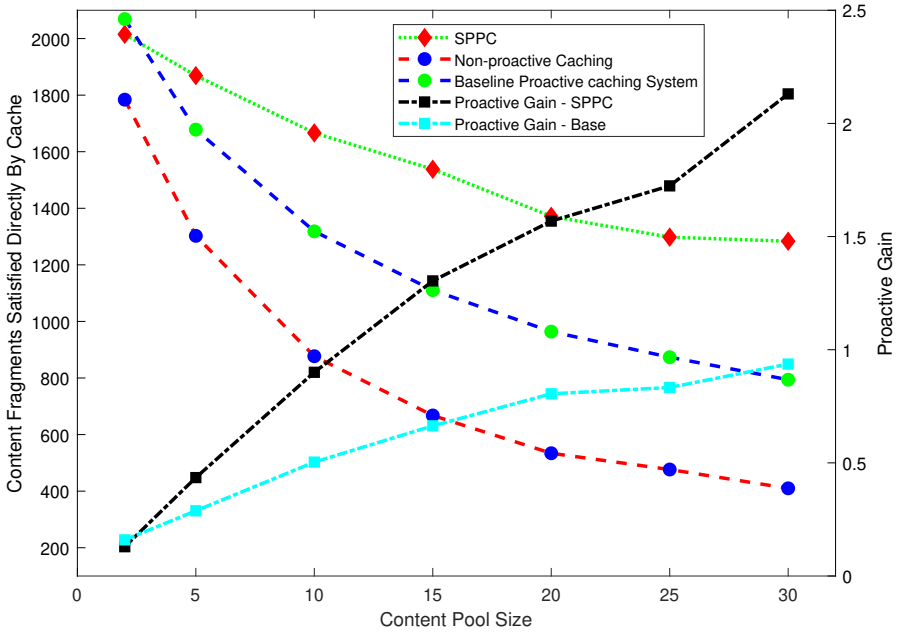
- *Content Fragments Satisfied via Backhaul Network (CFvBN):* the number of content fragments that are not transmitted from the caches of RSUs to vehicles
- *Content Fragments Satisfied via Cache (CFvC):* the number of content fragments that are transmitted from the caches of RSUs to vehicles

- *Cache Utilisation:* utilisation of caches in the system i.e., the ratio of content fragments transmitted directly by caches
- *Proactive Gain:* the gain achieved by proactive caching over the reactive caching system
- *Network Delay:* the amount of time of delay introduced because of RSUs requesting content from a content provider via backhaul network:

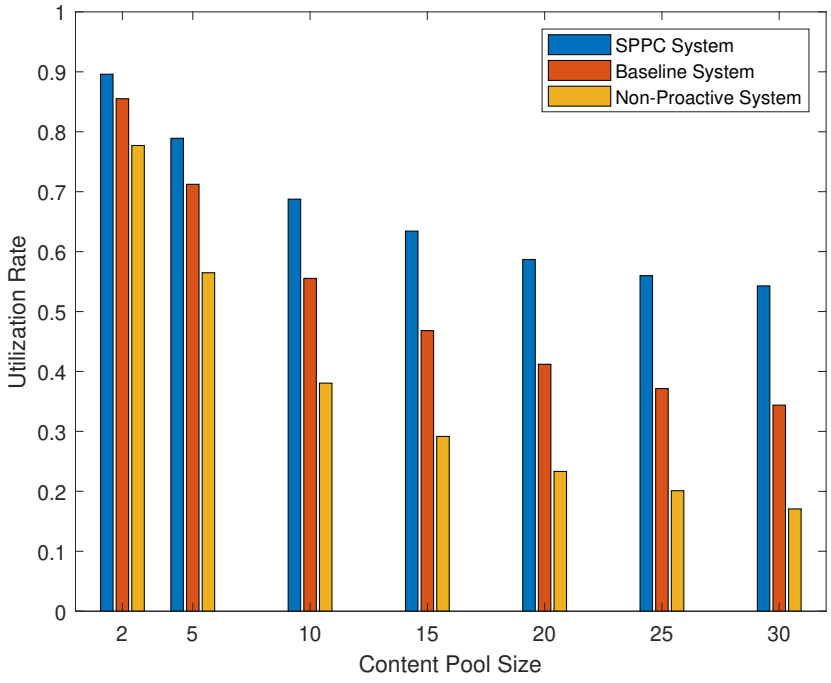
$$\begin{aligned} \tau &= \frac{NumberOfFragmentsByBackhaul \times SizeofFragment}{BackhaulTransmissionRate} \\ &= \frac{F_R \times F_c}{\omega} \end{aligned} \tag{4.1}$$

Figure 4.5 and Figure 4.6 demonstrate the performance of various systems in Las Vegas city. It has shown that the *SPPC system* outperforms its counterpart proactive system - *Baseline Proactive Caching system* (referred to as baseline system in the following) and the other two general systems. To illustrate, Figure 4.5 shows the caching performance of the three systems with cache enabled. Specifically, Figure 4.5a illustrates that with various content pool sizes, the SPPC system serves the most portions of content from system caches, which can dramatically improve vehicular users' service experience in a high-speed vehicular network. When the content pool size is large (e.g., 30) its performance can triple that of the reactive caching system by achieving a 200% gain compared to a 100% gain of the baseline system. Despite the remarkable gain, a decline in the CFvC can also be seen in the SPPC system due to the possible inaccuracy of sequence prediction for the next RSU. Figure 4.5b asserts the previous results. By introducing sequence prediction, cache utilisation remains at a superior level in the SPPC system even with a large content pool size.

Figure 4.6 shows the overall performance of the four systems with 7 content pool sizes. As shown in Figure 4.6a, each point on the line is the median value of 30 results (30 test traces) and as the pool size increases the number remains at a certain level for the no-caching system, whereas it rises in SPPC system and the other two. This is because

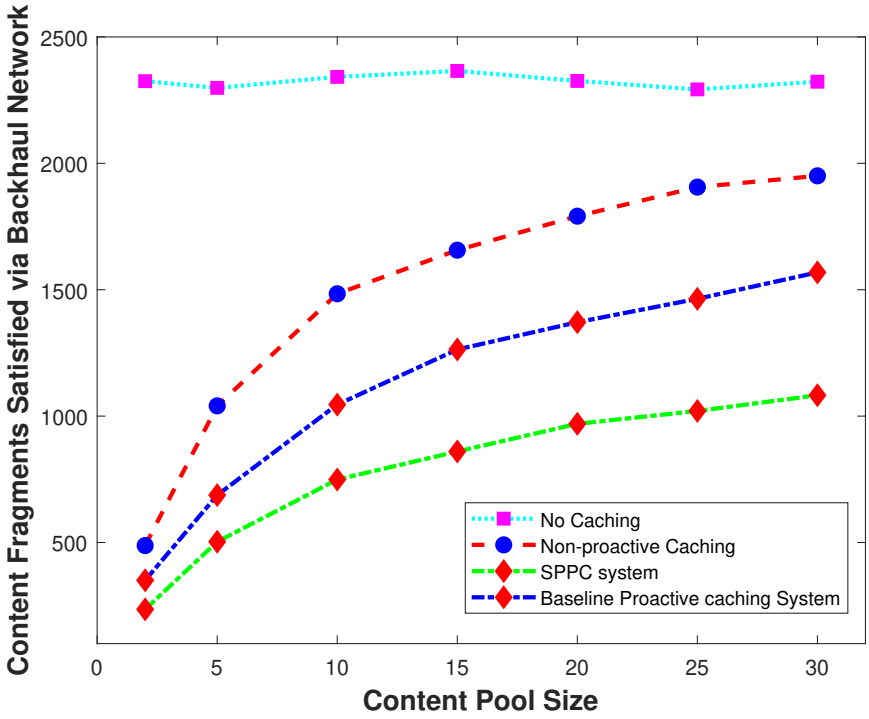


(a) Trend in the number of fragments served directly by the caches of the three systems (the median of 30 tests) at different content pool sizes and the gain of two proactive caching systems over the non-proactive caching system

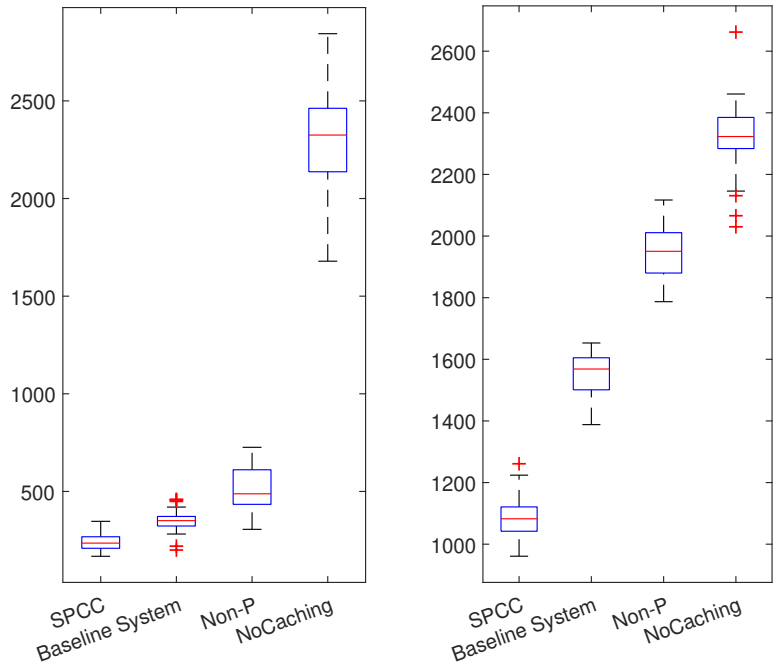


(b) Cache utilisation rate of the three caching systems at different content pool sizes

Figure 4.5 Performance of the SPPC proactive caching system compared to the baseline proactive caching system and the non-proactive caching system.



(a) Trend in the number of fragments served by the backhaul network of the four vehicular systems (the median of 30 tests) at different content pool sizes



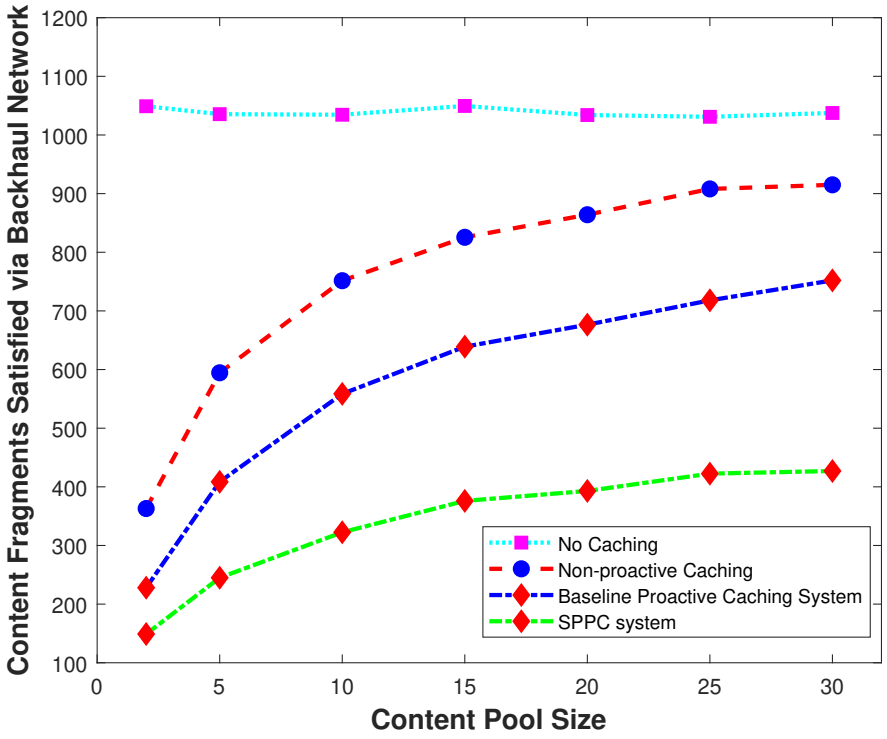
(b) Value distribution of (a) over 30 tests. The left boxplot is at content pool size of 2 and the right is of size 30

Figure 4.6 System performance benchmark of the four vehicular systems

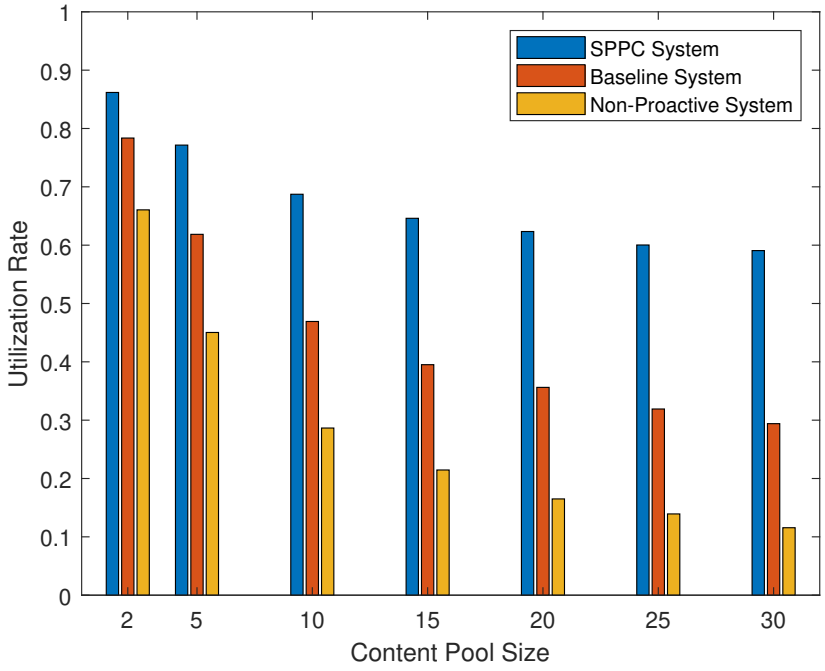
larger content size pool results in a higher chance that a vehicle may request content that is not cached. Therefore, the performance of reactive caching is approaching the no-caching system as the content pool size continues to increase. However, the SPPC system offsets more significantly the negative effects introduced by a large content pool than the baseline system, and the gap between the SPPC system and the other two systems is enlarged, which is believed to benefit from the gain of correct proactive caching. Figure 4.6b shows the box plots of CFvBN value distribution with content pool size 2 (left) and 30 (right). It can be seen that the SPPC system has the best performance with the smallest variation among all the tests. Results of other pool sizes fall in between.

As can be seen in Figure 4.7, some of the key system performance aspects of the Manchester area demonstrate a similar trend as in Las Vegas where the SPPC system still outperforms other systems considerably. The system performance comparison in a second city verifies the generality of the proposed scheme even in a location with more complex road planning. It is worth mentioning that as Manchester area is relatively smaller than the Las Vegas area as a whole, meaning that vehicle travelling time is shorter. This results in different absolute performance. However, as the relative size of the downtown of the two areas is similar, it has no impact on the conclusion that SPPC is an effective proactive caching system.

The Network delay comparison of the four systems in two cities is shown in Figure 4.8. In this figure, the result of the content pool size of 30 is analysed because the previous results show that the SPPC system has the most significant advantages in this case. A period of 1000s to 4500s after simulation begins is chosen to model a relatively steady state of vehicular traffic density. Every single point in this figure is calculated in the following way. Take the example of the SPPC system in Las Vegas. During each test run, whenever an RSU needs to request the desirable content from the remote server, the delay computed by Equation 4.1 will be recorded, as well as the timestamp, in a table datastructure. Therefore, after 30 tests, there will be in total 30 tables storing the statistics of network delay, where each table has two columns: the

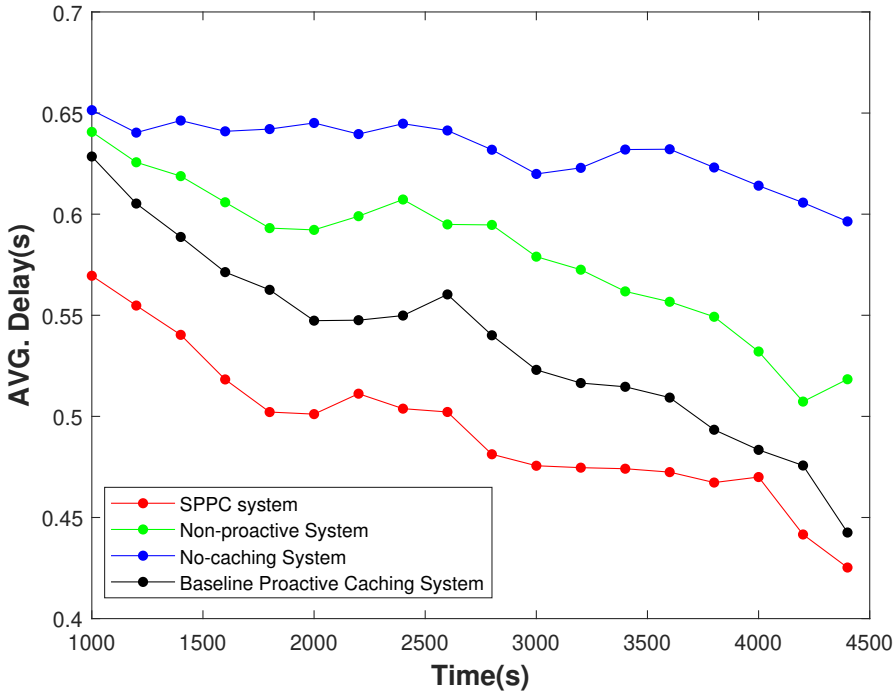


(a) Trend in the number of fragments served by the backhaul network of the four vehicular systems (the median of 30 tests) at different content pool sizes

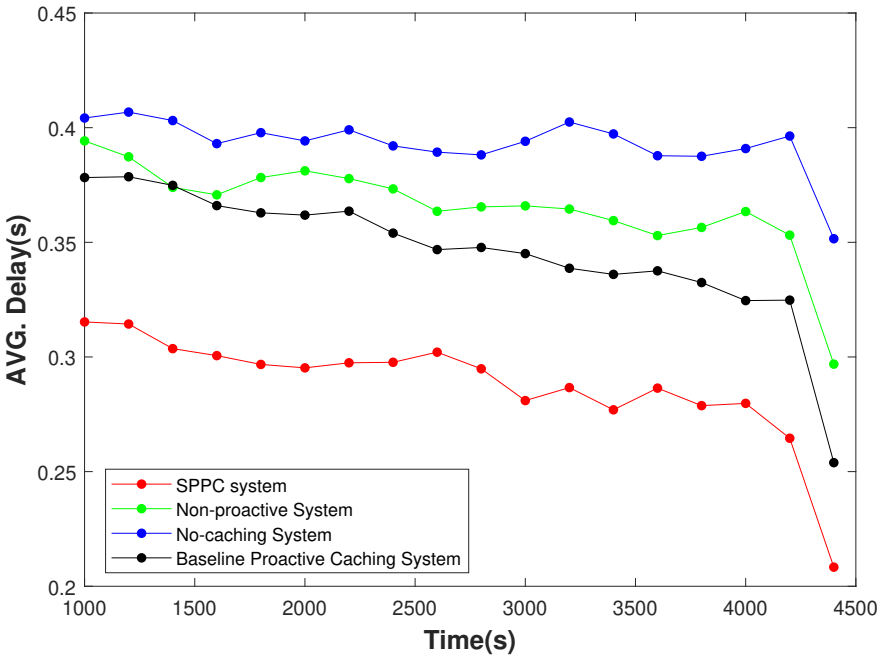


(b) Cache utilisation rate of the three caching systems at different content pool sizes

Figure 4.7 System performance of the four vehicular systems in Manchester



(a) Network Delay in Las Vegas - WS = 500s



(b) Network Delay in Manchester - WS = 500s

Figure 4.8 Network delay of the benchmarking systems in two cities. The sample frequency between points is 200s and a time window size (WS) of 500s is applied to average delay computation. Eventually, each point is the average of 30 tests.

left is the timestamp when the delay was recorded and the right column is the value of the delay. Then, a timestamp sample frequency of 200s and a window size (WS) of 500s are chosen. This means that the separation among the points on the horizontal axis of Figure 4.8 is 200s and the value of each point is computed by selecting a timestamp in a table every 200s, averaging the previous delays over the past 500 seconds, resulting in 30 values for a particular timestamp and eventually averaging the total 30 values. It validates the superiority of the proposed system from a different view where the SPPC system provides considerable delay reduction.

Benefiting from its excellent proactive caching capability, the network delay of the SPPC system is at the lowest level in the two cities among all the comparing systems. It can be observed that there is a declining trend in all four systems. One reason for this is that during each simulation test, more content is cached in local caches reactively due to earlier requests (except for the no-caching system) so the delay in the 3 caching systems has a similar decline rate but the two proactive caching systems have a relatively lower level of delay thanks to proactive caching feature. Another contributor to this, as demonstrated at the beginning of Section 4.4.1, is that vehicles keep leaving the network during the simulation, although a relatively steady state has been chosen. This explains why the no-caching system also shows this declining trend but at a slower rate than the other three systems with caching. On average, the SPPC system in Las Vegas (Figure 4.8a) can reduce the network delay by 24% and 18% in contrast to the no-caching system and non-proactive caching system, respectively. It is also approximately 9% better than the baseline system.

4.4.4 Discussion

Simplification of the communication model

It is worth noting that the communication systems considered in this thesis only characterise the basic features of the underlying communication model, as described in

Section 4.2. For example, vehicles will access the closest RSU and switch to the closest RSU. While this may not be the case in a general mobile network depending on the radio access technology (RAT) e.g., 4G or 5G, the RAT does not actually influence the mechanism and prediction performance of the proposed algorithms including the SPPC algorithm and the multi-armed bandit based algorithms in the following chapters. In addition, such simplification also includes the modelling of the network delay introduced by the backhaul network so the evaluation of network delay may not be perfect. However, this is sufficient to observe how a good proactive caching algorithm can benefit the wireless network. In other words, the more accurately the next RSU is predicted, the more effective a proactive caching algorithm is. This motivates the research work to keep exploring approaches to improve prediction accuracy. Therefore, this will be the primary focus of the remaining thesis.

Legitimacy of fragment-based content delivery

In the proposed MEC-Enabled vehicular network, the content transmission from RSUs to vehicles is abstracted as fragment-based. For some types of content requests such as web page requests, this may not be the case as the data volume involved is small. However, such abstraction is completely legitimate when delivering large data files or streaming content (e.g., videos) over many sources such as HTTP-based live streaming. Such services often involve continuous transmissions of large amounts of data, where intermittent connections between vehicles and RSUs have posed challenges. A representative technique is adaptive segmented HTTP-based content delivery such as Dynamic Adaptive Streaming over HTTP (DASH) [101], which breaks the content into a sequence of small segments (*aka* chunks), which are then served over HTTP. The work in reference [12] was conducted under this context and similarly, the work in [8, 20] also considered chunk-based video delivery. In the considered vehicular network in this thesis, when asking for the same content once a vehicle is connected to the new RSU, there is no need to request the same content from the origin server but it

will be served by the local cache of the RSU from the specific fragment or chunk (if the fragment is pre-cached). Therefore, it is a legitimate approach to base this study on fragment-based content delivery network, given the increasing popularity of live streaming multi-media content.

4.5 Conclusion

This chapter focuses on proactive caching at the next RSU to address the challenges brought about by high-speed mobility in vehicular networks, with the aim of improving vehicular users' network experience. To this end, the sequence prediction algorithm, CPT+, has been adopted and the proactive caching system, SPPC, with CPT+ as the underlying prediction algorithm is proposed to predict the next possible RSU of a vehicle's path. Vehicles' historical mobility data is used to train the CPT+ model. The simulation results have shown the superiority of the SPPC system in Las Vegas over the Baseline Proactive Caching system, the non-proactive caching system, and the no-caching system, with the proactive gain increased by a maximum 200% and network delay reduced by up to 24%. The performance of the SPPC in Manchester remains at a superior level. The better performance over the baseline system also highlights the important role of vehicular big data in accurate prediction. The chapter provides a use case and illustrates the feasibility of exploiting sequence prediction to solve proactive caching problems in vehicular networks. The variant of the SPPC system also serves as an important baseline proactive system in the following chapters.

Chapter 5

Proactive Edge Caching in Vehicular Networks: An Online Bandit Learning Approach

5.1 Introduction

The benefits of mobility prediction-based proactive caching have been well demonstrated in Chapter 4. By accurately predicting the next RSU and performing proactive caching on it, the network can achieve seamless content transmission to vehicles. One key constraint on the adaptability and applicability of the CPT+ based SPPC system is its requirement for offline training and its increasingly complex tree model. Some other studies in the literature have similar limitations. The Long Short Time Memory (LSTM) model used in [19] and [20] for the next RSU prediction requires a vast amount of offline training with labeled data. Similarly, the work in [51] requires numerous data to train the Prediction-based on Partial Matching (PPM) model and the vehicles need to send their trajectories to every RSU they visit, which inevitably raised concerns about transmission overhead and privacy issues. Therefore, these limitations have motivated the work in Chapter 5. The purpose of this chapter is to address the problem of proactive caching at the next RSU in vehicular networks through model-free reinforcement learning (RL) in an online learning mode, which is the first fundamental difference from [19, 20, 51]. Besides, the prediction model in [19, 20] is considered in a centralised way, i.e., prediction is made by a central node for a vehicle after the offline training stage. In contrast, the work in this chapter considers a distributed system where RSUs are learning and predicting independently, which is the second substantial difference.

Specifically, the RL technique adopted in this chapter is the *multi-armed bandit* (MAB) and its extension *contextual multi-armed bandit*. As discussed in Section 3.5, the reason for using MAB instead of the traditional RL methods is because the MAB model is a single-state model with no state transitions (i.e., stateless) and it is applicable and practical to consider the environment as stateless, given a highly dynamic vehicular network. The proactive caching problem is treated as a decision-making problem and the feasibility and prediction performance of bandit learning are investigated. To this end, two proactive caching algorithms based on MAB are proposed, *non-contextual*

MAB and *contextual MAB* (cMAB), by modelling the problem as a MAB and cMAB problem. The motivation for exploring these two MAB models is to further study the benefit on prediction performance by introducing *context* in contextual MAB.

Furthermore, this chapter also aims to investigate the uncertainty behind the proposed proactive systems with *Subjective Logic* framework, because it is very helpful to verify and support the superiority of the proposed systems from the theoretical perspective. Uncertainty is inextricably linked to learning algorithms and their models and is an important concept in machine learning methodology [104]. Assessing and quantifying uncertainty helps us to understand more precisely the benefits that these models can bring. The subjective logic framework, first proposed in [105], has emerged as an effective method for uncertainty evaluation. This formalism makes it possible to express specific forms of probability distributions by generating a *multinomial opinion* over a discrete set of elements. It provides a concise formalism to represent Dirichlet-multinomial and Dirichlet-categorical models [99] and therefore, the opinion induces a categorical distribution over the element set that allows evaluation of the overall uncertainty as the entropy of the distribution. The work in [99] used a subjective logic framework to solve bandit problems, where the action selection is based on sampling the multinomial opinion over the action set. They quantified the overall uncertainty of the proposed system with the entropy of the categorical distribution. The authors in [106] argued that the Beta distribution and subjective logic are isomorphic in terms of fusion while finding the equivalence between uncertainty and entropy of Beta models. It has also been used for assessing uncertainty in deep networks as studied in [107] and [108].

The rest of the chapter is structured as follows. Section 5.2 mainly introduces the technical details of the proposed MAB-based algorithms. Section 5.3 introduces the Subjective Logic uncertainty analysis framework and elaborates on how it is applied to the analysis of proactive caching systems. Section 5.4 shows the simulation results. Section 5.5 discusses the theoretical analysis, time complexity and convergence of

the proposed algorithms. Section 5.6 discusses an extended work on two-dimensional cMAB and Section 5.7 concludes the chapter.

5.2 Design of MAB-based Proactive Caching Algorithms

The primary focus of the proactive edge caching problem in this thesis is where to pre-store relevant content in the immediate future. Therefore, it is vital for an RSU to predict, as accurately as possible, the next potential RSU a vehicle is about to hand over to. Intuitively, this may not seem to be closely related to a MAB problem. Based on the MAB principles introduced in Section 3.5, the main topic of the following is to demonstrate how to associate them together and how to solve the proactive caching problem with the proposed algorithms.

5.2.1 System Model

The underlying vehicular system architecture considered in this chapter is identical to that discussed in Section 4.2, shown in Figure 4.1. However, unlike the earlier SPPC system, the MAB-based system in this chapter is a pure proactive caching system, which means that content will not be cached reactively and vehicles are served either by content cached proactively when a cache hit or by the content server in the backhaul network in case of a cache miss. In addition, the central node/server now plays a role in providing prediction feedback to RSUs so that they can update their decision policies. As depicted in Figure 5.1, the functionality of the central server that connects multiple RSUs is to transmit *prediction/cache hit feedback* message which acts as *rewards* in the MAB-based learning algorithms, when a vehicle connects to a new RSU. This helps RSUs focus on independent prediction, sending and receiving proactive caching

requests and performing caching tasks. Despite the assistance of the central node, the RSUs still make predictions in a distributed way.

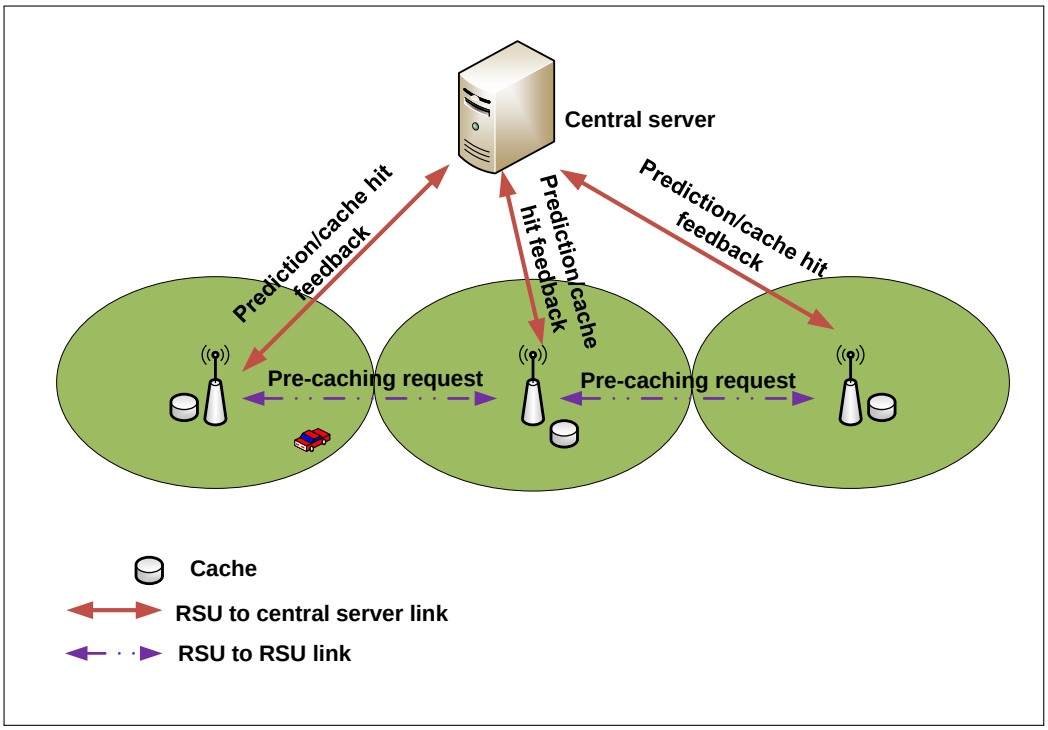


Figure 5.1 An illustration of the distributed structure of and signalling in proactive caching systems in this chapter

5.2.2 Mapping Proactive Caching to MAB Problems

As discussed earlier in Section 3.5, a MAB problem is composed of an action set, rewards and agent as well as context in a contextual bandit (cMAB) problem. Here, an agent aims to maximise its cumulative rewards by taking appropriate actions from the action set in a given period of time. Regarding the next-RSU proactive caching scenario in vehicular networks, an RSU assists a vehicle to successfully hit the content that was previously being transmitted. They resemble each other in terms of node selection and success or failure (reward). Therefore, the proactive caching problem can be seen as a MAB problem using the following mappings:

- **RSU as bandit learning agent**: Any RSU in the vehicular network acts as a learning agent, and its neighbouring RSUs are equivalent to its actions. Predicting the next RSU as a proactive caching node is actually making a decision on one of the agent RSU's neighbours.

- **Stateless RSU**: In general, the state of a reinforcement learning agent is associated with the environment. Since the interaction of an RSU with the vehicular environment can be extremely dynamic and complicated to represent, the single-state feature of MAB resolves this problem. In other words, an agent RSU is single-state or stateless which means that it does not transfer to a new state by taking an action.

- **Action selection as next RSU prediction**: The agent RSU will either exploit its current knowledge to select the *greedy* action/neighbour or explore other non-greedy actions that may return a higher reward depending on the exploration-exploitation scheme adopted.

- **Reward generation**: When handover happens, the system will return a reward to the previous agent RSU. This is achieved by determining whether there is pre-cached content in the RSU after the handover, or alternatively whether the RSU is the previously predicted one. The reward in return helps an agent RSU compute the estimated values of its actions.

- **Previous RSU as context**: The agent RSU may also make use of contexts for its action selection as in a *contextual bandit problem*. By identifying the previous RSUs that the visiting vehicles coming from as contexts, it can map such contexts into various bandit tasks and perform more effective learning. Technical details about the contextual information will be covered shortly.

In a vehicular system with multiple RSUs, the problem becomes a *multi-player, multi-armed bandit* problem where each individual RSU is an independent player and learns its own best action or best policy. On this basis, two algorithms are designed to address

proactive caching: *non-contextual (Bernoulli) MAB* and *contextual MAB*, and the detailed design will be elaborated in the following subsection.

5.2.3 Addressing Proactive Caching with bandit learning

The two bandit learning algorithms that address proactive caching are explained from three aspects: *action selection and value estimation*, *reward function*, and *context information*.

a) **Action selection and value estimation** Two critical elements in MAB problems are action selection and action value update. Since the purpose of the thesis is not to find a sophisticated action-selection strategy, the popular ϵ-greedy method introduced in earlier Equation (3.12) is used to make a selection: the best action is selected with a probability of $1-\epsilon$; otherwise, actions will be selected randomly with a small probability ϵ regardless of their action values.

$$A_t = \begin{cases} \arg\max_a Q(a), & 1-\epsilon \\ random, & \epsilon \end{cases} \tag{5.1}$$

Another important method is the action value estimation, also known as *action-value method* in the literature. In Section 3.5, a recursive form of value estimation function has been developed with Equations (3.9) and (3.10) and it has been adopted in the proposed algorithms:

$$Q(a) \leftarrow (1-\alpha)Q(a) + \alpha r \tag{5.2}$$

where $Q(a)$ is the quality value of action a, named **Q**-value as in **Q**-learning [22][94], r is the reward associated with the most recent trial and is determined by a reward function, and $\alpha \in [0,1]$ is the step-size or*learning rate*.

b) **Reward function** The reward function R is used to generate a reward associated with the action taken previously when an outcome is observed. Given an action a taken at timestamp t and the observed outcome as b (may occur immediately), its reward can be computed with $r_t = R(b)$. In a Bernoulli MAB problem discussed in Section 3.5, the reward function R is actually the outcome itself (1 or 0), meaning that $r_t = R(b) = b$. Here, the reward function considers punishment for a caching miss or a wrong next RSU prediction. Such a reward function has been successfully applied to Dynamic Spectrum Access problems in [22, 109].

$$R(b) = \begin{cases} 1, & b = \text{True} \\ -1, & b = \text{False} \end{cases} \tag{5.3}$$

The outcome b is determined by observing whether a vehicle switches to the predicted RSU, equivalent to a cache hit or miss if pre-caching request was sent to the RSU. The relevant reward will then be generated with Equation (5.3) and fed back to the earlier decision-making RSU. With Equations (5.2) and (5.3), the learning agent aims to update its estimate of each action $Q(a) = E[r_t]$, make an action selection and maximise its cumulative rewards $max \sum r_t$.

It is worth noting that, due to the constant α adopted in Equation (5.2) and the negative reward introduced in Equation (5.3), $Q(a)$ is no longer the success probability θ as in the Bernoulli multi-armed bandit (Equation (3.7)) but directly represents the expected reward of the action a.

c) **Contextual information**

The above methods for updating actions' Q-values, selection, and reward function can be applied to both non-contextual and contextual bandit problems. However, the agent in the general non-contextual MAB learning could face the dilemma where two or more of its actions may converge to very close estimated Q-values, which creates great uncertainty when predicting an accurate next RSU node. Therefore, the motivation for proposing the contextual MAB algorithm is to

resolve this as best as possible. The agent in a contextual MAB problem maps *contexts* to its action set and associates a specific Q-table with each individual context and aims to learn a policy under different them. In the vehicular network, vehicles may come from various directions which can be useful contextual information. If the agent RSU can make use of it and split it to separate bandit tasks, it is likely to improve the overall accumulated rewards.

Specifically, the context introduced on top of a non-contextual MAB-based algorithm is the previous RSU that a vehicle connected to before the current agent RSU. The rationale behind this is that the previous RSU is a very easily accessible context and this does not require additional effort on signalling extra information, compared to other types of context e.g., road information, vehicle angel, etc. Once a vehicle connects to an RSU and starts to request content from it, the agent RSU needs to predict the next RSU (action selection) and inform it to pre-cache the needed content if necessary. In contextual MAB, the agent RSU now needs to first identify the previous RSU as context and learn the action values associated with it so that decisions are properly made under a particular context. The equivalent equations to Equations (5.1) and (5.2) for action selection and Q-value updating in contextual MAB become:

$$A_t = \begin{cases} \arg\max_a Q_t(a \mid s_j), & 1-\epsilon \\ random, & \epsilon \end{cases} \tag{5.4}$$

$$Q(a \mid s_j) \leftarrow (1-\alpha)Q(a \mid s_j) + \alpha r \tag{5.5}$$

where s_j is the detected context at t.

As mentioned earlier, being able to distinguish the incoming directions could help resolve the dilemma in a non-contextual MAB problem. Prior RSUs can be straightforwardly accessed and used as a reference to such directions compared to other sorts of information (e.g., road information, vehicle angel, etc.), and

this enables the agent RSU to solve separate bandit tasks associated with them, thereby guaranteeing a more effective policy learned than in a non-contextual MAB problem.

Algorithm 1: Non-contextual multi-armed bandit

Initialisation (if not done): For RSU $m \in \mathcal{M}$ with the number of actions (RSU neighbours) $\mathcal{A}_m$, their Q-values are initialised to $Q(a) = 0$ for $a \in \mathcal{A}_m$;
while *not the end of the test* **do**
  if *Content transmission is happening whilst in RSU* m **then**
    Predict the next RSU by:
    $a^* \leftarrow$ selection decision based on Eq. (5.1);
    Precaching content at a^* if needed;
  end
  if *Handover happens* **then**
    $r^* \leftarrow$ observe the reward according to Eq. (5.3);
    Update $Q(a^*)$ with Eq. (5.2);
  end
end

Algorithm 2: Contextual multi-armed bandit

Initialisation (if not done): For RSU $m \in \mathcal{M}$ with the number of actions (RSU neighbours) $\mathcal{A}_m$, their Q-values are initialised to $Q(a) = 0$ for $a \in \mathcal{A}_m$;
while *not the end of the test* **do**
  if *Content transmission is happening whilst in RSU* m **then**
    $s \leftarrow$ detect the previous RSU s before m;
    if *s is a* ***new*** *detection* **then**
      Create an **entry** of s to its action values;
      Initialise $Q(a \mid s) = 0$ for $a \in \mathcal{A}_m$;
    end
    Predict the next RSU by:
    $(a^* \mid s) \leftarrow$ selection decision based on Eq. (5.4);
    Precaching content at a^* if needed;
  end
  if *Handover happens* **then**
    $r^* \leftarrow$ observe the reward according to Eq. (5.3);
    Update $Q(a^* \mid s)$ with Eq. (5.5);
  end
end

The above method for solving proactive caching with MAB learning is summarised in **Algorithm 1** and **Algorithm 2** for non-contextual and contextual bandit learning respectively, which have been applied to the proactive caching problem. Additionally, a general flowchart of MAB-based proactive caching is integrated and shown in Figure 5.2, though contextual MAB may also involve identifying the context and updating its action values correspondingly.

5.3 Uncertainty Analysis Model

In decision-making problems, reducing uncertainty is deemed to be vital as less uncertainty means that an agent is likely to make more accurate decisions. Thus, it is meaningful to assess and quantify the uncertainty in a learning problem. This work uses *Subjective Logic* framework [110] and particularly adjusts it to investigate uncertainty in bandit learning based proactive caching systems. The motivation behind this is to provide a more insightful analysis model for the performance of proactive caching systems and how uncertainty evolves during the learning process. Another aim is to give a greater insight as to how MAB-based systems outperform the others and how the context introduced by the contextual MAB algorithm could benefit the whole system. This section will introduce some background and discuss how this is achieved.

5.3.1 Uncertainty

In the field of machine learning and statistics, a reliable estimation of uncertainty plays an important role in order to create reliable statistical models [99]. In [104], uncertainty in statistical models is classified as *aleatoric* and *epistemic*. Given a set of observed data samples $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ that are generated by an unknown stochastic process P, if the task is to fit a model $p(\mathcal{D} \mid \Theta)$ that describes the observation $\mathcal{D}$, the set of parameters Θ needs to be learned from the collected observations. Apparently, the uncertainty that affects the accuracy of model $p(\mathcal{D} \mid \Theta)$ comes from both the internal

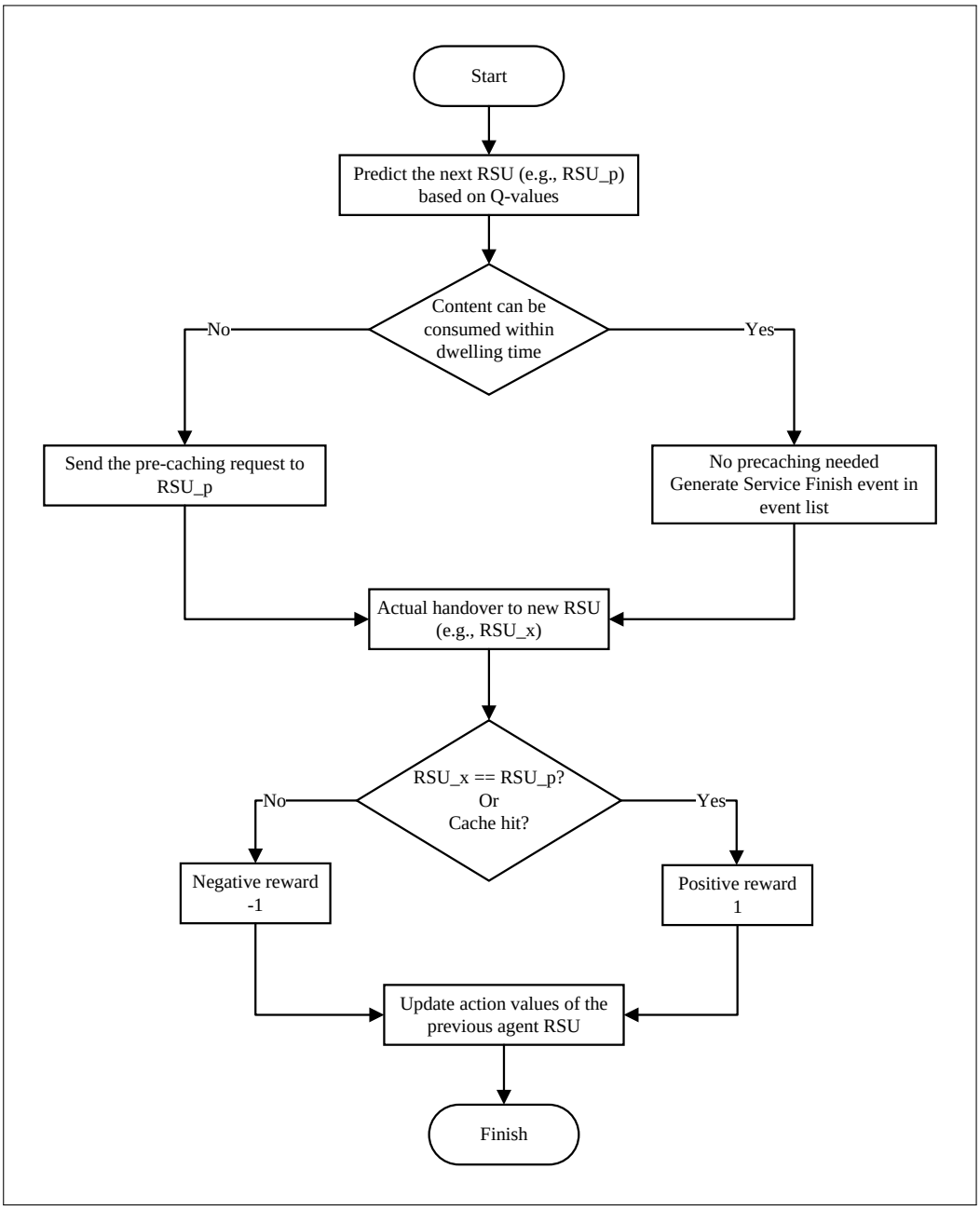


Figure 5.2 Flowchart of MAB-based proactive caching algorithm

This is a general cycle of an agent RSU serving a connecting vehicle, from *Start* when it receives a content request from a connecting vehicle, to *Finish* when its action-value table is successfully updated with corresponding rewards.

randomness of process P and the limitation of the number of observations used to estimate the model. Therefore, these two types of uncertainty can be described as:

- *Aleatoric uncertainty* is inherent randomness in the data generation process P which can be reflected by the variability in the outcome of a trial. A typical example is coin flipping. For this type of uncertainty, however much data is provided, the uncertainty of the final fitted model $p(\mathcal{D} \mid \Theta)$ is unlikely to be less than the underlying model P [99].

- *Epistemic uncertainty* on the other hand, is due to the lack of knowledge about the best model such as finite sample size. Different from aleatoric uncertainty, epistemic uncertainty can be improved by having more samples or trials.

The present study concentrates on the overall uncertainty of bandit learning algorithms, accounting for both aleatoric and epistemic uncertainties, which can be computed as the entropy of the relevant distribution under the subjective logic framework.

5.3.2 Subjective Logic

Subjective logic [110] has been a promising approach to evaluating uncertainties in a statistical model. It is a compact formalism to represent specific forms of probability distributions (Dirichlet-multinomial and Dirichlet-categorical models) [99]. Specifically, given a discrete domain $\mathcal{X} = \{x_1, x_2, ..., x_k\}$ with k elements, there exists an *multinomial opinion* for the domain:

$$o = (\boldsymbol{b}, u, \boldsymbol{c}), \text{subject to } u + \sum_{i=1}^{k} b = 1$$

- $\boldsymbol{b} \in \mathbb{R}^k_{\geq 0}$: *belief vector* that represents the degree of certainty over the k elements

- $u \in \mathbb{R}_{\geq 0}$: *uncertainty scalar* that shows the degree of certainty on belief vector

- $\boldsymbol{c} \in \mathbb{R}^k_{\geq 0}$: *base rate vector* which often expresses the prior probability distribution of the k elements

According to [99], the belief vector acquires the *first-order* uncertainty of the distribution of beliefs over the domain mapping to the aleatoric uncertainty whereas u maps to epistemic uncertainty capturing the *second-order* uncertainty about the belief model. In such a model, the probability of an element x_i in the domain $\mathcal{X}$ with opinion o can be computed with:

$$p\left(x_i \mid o\right) = b_i + uc_i \tag{5.6}$$

An existing mapping between an opinion $o = (\boldsymbol{b}, u, \boldsymbol{c})$ and an evidential Dirichlet pdf $s = Dir_e(e)$ [99] [110]:

$$\begin{cases} e_i = \frac{Wb_i}{u} & \text{if } u \neq 0 \\ e_i = \infty & \text{otherwise} \end{cases} \tag{5.7}$$

whose reverse is:

$$\begin{cases} b_i = \frac{e_i}{W + \sum_{i=1}^{k} e_i} \\ u = \frac{W}{W + \sum_{i=1}^{k} e_i} \end{cases} \tag{5.8}$$

where W is a non-informative prior weight normally specified equal to 2 for consistency.

Equations (5.7) and (5.8) form a theoretical foundation for the uncertainty analysis in the present work. Most importantly, Equation (5.8) allows building the multinomial opinion o over actions of RSUs with experiment observations (i.e., evidence). Thus, the probabilities of actions and overall uncertainty in the form of entropy can be obtained accordingly. How the evidence and the overall uncertainty calculation are defined will be discussed in the following.

5.3.3 Uncertainty evaluation of proactive caching systems

Similar to the uncertainty in decision-making theory, two sources of uncertainty exist in proactive caching systems, corresponding to aleatoric and epistemic uncertainties. On the one hand, for an RSU, the right decision depends on the proactive caching scheme as well as the randomness in the system. These are all inherent aleatoric uncertainty.

On the other hand, epistemic uncertainty in such systems comes from the lack of visits of the RSU or the lack of chances for it to make decisions, which should be reduced as more observations are collected.

To form an opinion over an action set, the evidence of the set needs to be collected, with which the corresponding belief vector and uncertainty scalar of the opinion tuple can be obtained through Equation (5.8). The probability of individual action can be achieved accordingly via Equation (5.6). For an arbitrary RSU with m actions, the subjective opinion $o^t = \left(\boldsymbol{b}^t, u^t, \boldsymbol{c}\right)$ at an arbitrary timestep t conveys:

- the belief of an agent on action a_i being the best action with b_i^t
- the global uncertainty over the beliefs with u
- the prior belief c which is constant

Therefore, at timestep 0 or in the beginning of the learning process, the initial values of the three elements are:

$$\begin{cases} b_i^0 = 0 & \forall i \in [1, m] \\ u^0 = 1 & \\ c_i = \frac{1}{m} & \forall i \in [1, m] \end{cases}$$

which means that the agent has no knowledge about which of its actions is likely to be the best and they have equal probabilities. The uncertainty scalar at this point is the maximum, 1. According to the definition of multinomial opinion in the beginning of Section 5.3.2, $u + \sum_{i=1}^{k} b = 1$, u^0 is equal to 1 if $\sum_{i=1}^{k} b_i^0 = 0$.

The rule used for collecting evidence that supports the belief that action a^t could be the best is straightforward:

$$e_i^{t+1} = e_i^t + \mathbb{1}\left[a^t = a_i\right]$$

where the evidence is updated by adding one piece if $[a^t = a_i]$ is true. Thus, the evidence at any timestep t forms the opinion o^t and with Equation (5.6) a categorical

distribution of the action set can be induced: $p(\mathbf{a} \mid o^t) = \mathbf{Cat}(\mathbf{b}^t + u^t\mathbf{c})$. From this distribution, the overall uncertainty can be calculated as the entropy of the distribution:

$$H = -\sum_{i=1}^{m} p(a_i \mid o^t) \log_2 p(a_i \mid o^t) \tag{5.9}$$

For the non-contextual MAB algorithm, Equation (5.9) can be applied directly because of its single-state feature. In contrast, for contextual MAB, the entropy computation needs to consider the number of contextual situations.

Given an agent that has n contextual situations denoted by $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ with m actions, each of these situations is an independent bandit task as mentioned earlier. As a result, their entropy can be computed, called *context entropy* as:

$$H(s_j) = -\sum_{i=1}^{m} p(a_i \mid o^t, s_j) \log_2 p(a_i \mid o^t, s_j) \tag{5.10}$$

For the agent, the global uncertainty in terms of entropy then becomes:

$$H = \frac{\sum_{j=1}^{n} H(s_j)}{n \log_2 m} \tag{5.11}$$

This draws on the *Exploration Entropy* in a full reinforcement learning problem [111] where multiple *states* are associated with an agent.

In the proactive caching system, the actions of an agent RSU have their own success probability, which is a source of the aleatoric uncertainty. As mentioned earlier, even the optimal model cannot have less uncertainty than the true process. The MAB-based algorithms cannot remove such intrinsic uncertainty but aim to form a belief vector $\boldsymbol{b}$ over the actions that best describes it. For non-contextual MAB, sufficient learning (trials) allows the agent RSU to have the best model for the aleatoric uncertainty, compared to other non-contextual baseline systems (which shall be seen in the results section). In other words, enough evidence results in a small epistemic uncertainty u, and a smaller overall uncertainty means a better-fitted model. Contextual MAB (cMAB), on the other hand, introduces a context (i.e., previous RSU) to further disaggregate

the problem into context-related ones. The aleatoric uncertainty under each context s may be substantially reduced in contrast to the non-contextual case. Therefore, after sufficient learning, the agent RSU will have the best model for the aleatoric uncertainty associated with each context s, thereby less overall uncertainty.

To sum up, Equation (5.9) will be applied to evaluate the overall uncertainty in the non-contextual MAB-based proactive caching algorithm, and Equation (5.10) and (5.11) will assess the contextual MAB-based algorithm.

5.4 Simulation and Performance evaluation

5.4.1 Simulation

Similar to Chapter 4, the simulation for evaluating MAB-based proactive caching system also consists of two parts: traffic simulation and network simulation. Therefore, vehicle traffic traces are generated by SUMO [73] and they are again processed with event-driven network simulation module implemented in MATLAB [81].

Traffic Simulation and Scenario

SUMO is used to simulate a real transportation network discussed in Section 4.2. The scenario considered here is the daily commuting routine of people living in a particular urban area. Same with the work in Chapter 4, two areas in Las Vegas and Manchester are investigated to generalise the application of MAB-based schemes to two cities with different road layouts. The configuration of traffic simulation in SUMO e.g., the number of vehicles, traffic assignment zone (TAZs), time, etc, are also identical to Section 4.4.1. However, differently, since the focus of this chapter is on online learning, training data is no longer needed. Instead, there are 200 test trace files for each city to simulate 200 workdays.

Network Simulation

The network simulation used in this chapter is also DES as in Section 3.2.2, which allows the vehicular network simulation to be performed through a series of events and the composition of the event list can be found in Section 3.2.2. Test traces generated by SUMO are passed to the simulation system sequentially. As the present work concentrates on online learning, a complete cycle of the simulation is testing 200 trace files and the learners (i.e., RSUs) make predictions as they learn throughout the simulation cycle and become increasingly knowledgeable as the simulation runs. The structure of the simulation modules has already been shown in Figure 3.4.

Parameters that are specific for this work are summarised in Table 5.1 and the other relevant ones can be referred to Table 4.1. The closely related parameters to the MAB-based systems in Table 5.1 are learning rate α referenced in [112] and ϵ selected empirically. Unlike Chapter 4 that studied 7 content pool sizes, this chapter uses a fixed content pool size of $K = 30$ to fully concentrate on prediction performance.

Table 5.1 Simulation Parameters

Parameter	Value
α for bandit learning	0.5
ϵ for bandit learning	0.05
No. of test traces	200
Size of content database	$K = 30$

5.4.2 Performance Evaluation

The performance of non-contextual and contextual MAB-based proactive caching systems is compared with three other proactive caching systems:

- *Equal Probability-based Proactive Caching System*: RSUs select a pre-caching node with equal weight from their neighbours. Such a scheme implies that the RSUs are not intelligent and it randomly selects a neighbour as a proactive

caching node. However, this scheme can act as a baseline system with possibly the lowest performance bound, similar to the baseline system used in Chapter 4.

- *Probability-based Proactive Caching System*: In this system, RSUs are intelligent so that they can predict the next RSU based on history. In other words, the RSUs will refer to transition recordings to their neighbours and establish a transition probability vector based on which the prediction is made. This is an intuitive scheme where an RSU believes the neighbour with more frequent handovers deserves a higher weight to become the caching node.

- *CPT+ based Proactive Caching System*: This system is an online variant of the SPPC system proposed in Chapter 4. The SPPC system was proved to be effective but it was designed to be an offline system and its prediction accuracy was not evaluated in Chapter 4. Therefore, to compare its performance to MAB learning based ones, it is redesigned to be capable of online learning using CPT+ model. In this system, there is a central dataset which will store the sequence of RSUs every time a vehicle leaves the network. This process continues from the beginning until the end of the simulation. Therefore, unlike the offline SPPC system in Chapter 4 where there was an individual training dataset for each vehicle, here when an RSU needs to predict the next RSU for a vehicle, it will need to extract the central dataset and train the CPT+ model. Then the prediction is made in the same way as the SPPC system.

Remark: the five systems are referred and denoted in the following as *cMAB* and *MAB* for contextual and non-contextual bandit learning systems, respectively; *EQ*, *PB*, and *CPT+* represent for equal probability-based, probability-based, and CPT+ based systems, respectively.

Evaluation metrics

The evaluation of the proactive caching performance of the systems is the primary focus in this part. An action selection is considered correct when the selected pre-caching neighbouring RSU is the actual RSU to which the vehicle has switched. In the systems considered here, this is identical to a cache hit. Additionally, the extended subjective logic framework discussed in Section 5.4 is applied to the systems to provide an analysis of uncertainty except for CPT+. This is because CPT+ is a fundamentally different algorithm compared to the other four, in terms of its model and algorithm design. The variability of its action set and the difficulty of accessibility to "contexts" have made the extended uncertainty model inapplicable. The entropy calculation for EQ and PB systems is also based on Equation (5.9) as the non-contextual MAB. Furthermore, how proactive caching systems benefit the network is also considered.

The following aspects will be evaluated in the results:

- *Cumulative prediction accuracy*: Denoting the total number of predictions as $Q_{prediction}$ and correct ones as $Q_{correct}$ of test trace n, the cumulative prediction accuracy PA up till trace n is defined as:

$$PA = \frac{\sum_{i=1}^{n} Q_{correct}}{\sum_{i=1}^{n} Q_{prediction}}$$

- *Cumulative distribution function (CDF) of uncertainty*: Aims to show uncertainty at the system level as well as some particular RSUs.

- *Proportion of Proactive Caching Content Fragments*: the proportion of the number of content fragments that are proactively cached and transmitted to vehicular users. This reflects the effectiveness of a proactive caching system.

- *Average network delay:* this is the average network delay caused by cache misses due to wrong predictions in a proactive caching system during a test trace. When a cache miss occurs during a particular test, it is recorded as an entry associated

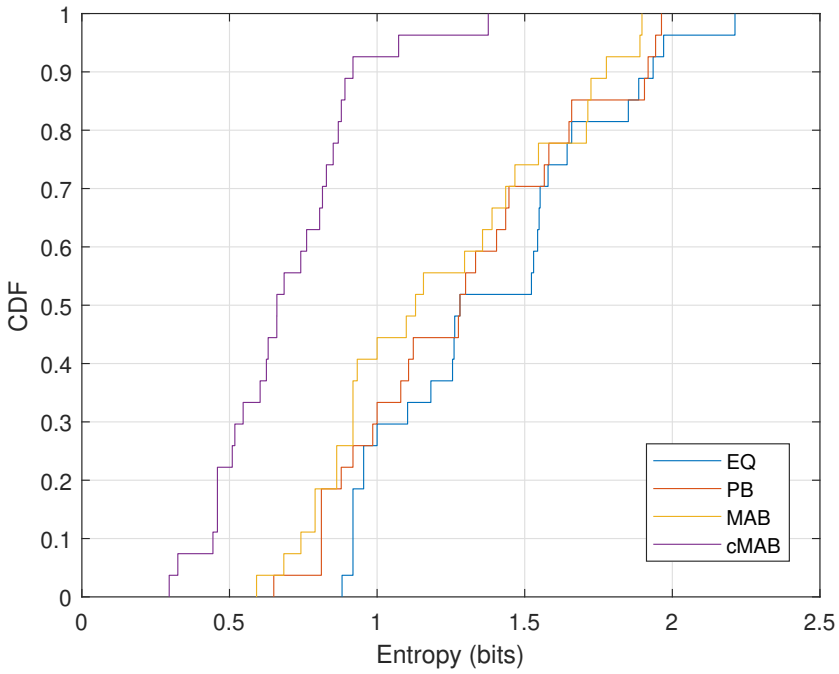
with the number of fragments that have been transmitted through the backhaul network. The average number of such content fragments during the test trace is the summation of them divided by the total number of total entries. The average delay can then be computed by the following equation similar to Equation (4.1):

$$\begin{aligned}\tau &= \frac{\frac{\sum NumberOfFragmentsByBackhaul}{TotalNumberOfDelayEntries} \times SizeofFragment}{BackhaulTransmissionRate} \\ &= \frac{\frac{\sum F_{Backhaul}}{D} \times F_c}{\omega}\end{aligned} \tag{5.12}$$

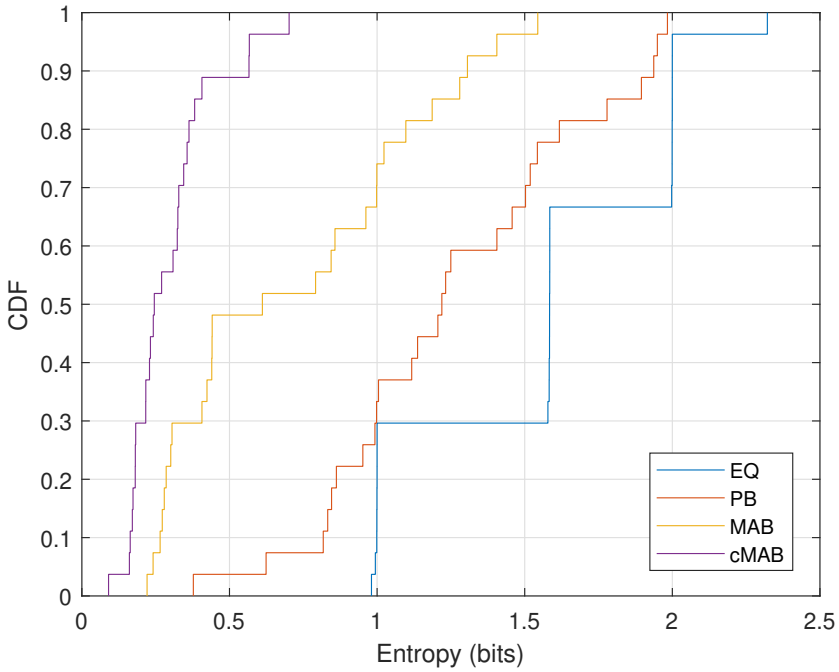
Experimental results

The results of Las Vegas will be elaborated in detail as it is the primary city, while the results of the secondary city Manchester will be demonstrated in a more general way.

Figure 5.3 shows the uncertainty analysis of four proactive caching systems in Las Vegas at a system level. It is the cumulative distribution of the uncertainty (entropy) of 32 RSUs at the end of test trace 1 and 200, respectively. These results illustrate performance before and after learning. The two bandit learning schemes, non-contextual MAB-based (MAB) and contextual MAB-based (cMAB), outperform the other two baseline schemes in terms of the reduced amount of uncertainty in decision-making. Both MAB and cMAB have dramatically reduced the uncertainty level through sufficient learning after 200 traces. The proportion of RSUs with entropy less than 0.5 bits has increased from 0% to 49% and 20% to 90%, respectively. The superiority of the cMAB-based system over its counterpart benefits from introducing the context information. Uncertainty distributions of bandit learning schemes were close to PB and EQ systems in the initial stage of simulation, but this gap has been enlarged in the end. The percentages of RSUs with less than 1-bit entropy are 100%, 80%, 40%, and 0% for cMAB, MAB, PB, and EQ respectively. The PB scheme has not experienced a significant change from this perspective because of its nature. Since the test traces simulate vehicles following their own daily commuting routines, the



(a) CDF of uncertainty at test trace 1



(b) CDF of uncertainty at test trace 200

Figure 5.3 Cumulative distribution function (CDF) of the overall uncertainty in Las Vegas. The figure demonstrates the reduction in uncertainty of the four proactive caching systems in the form of CDF of entropy.

transition probability matrix or the weights used by PB scheme for decisions does not vary too much at the end of traces 1 and 200. By contrast, despite the fluctuations in the initial stage of simulation due to the lack of samples, the EQ scheme is constantly the one with the highest overall uncertainty and converges to a stable state finally. This makes sense from the viewpoint of information theory [113] as the entropy of an RSU with m neighbours is maximised to $\log_2 m$ with equal probability $\frac{1}{m}$ among the neighbours.

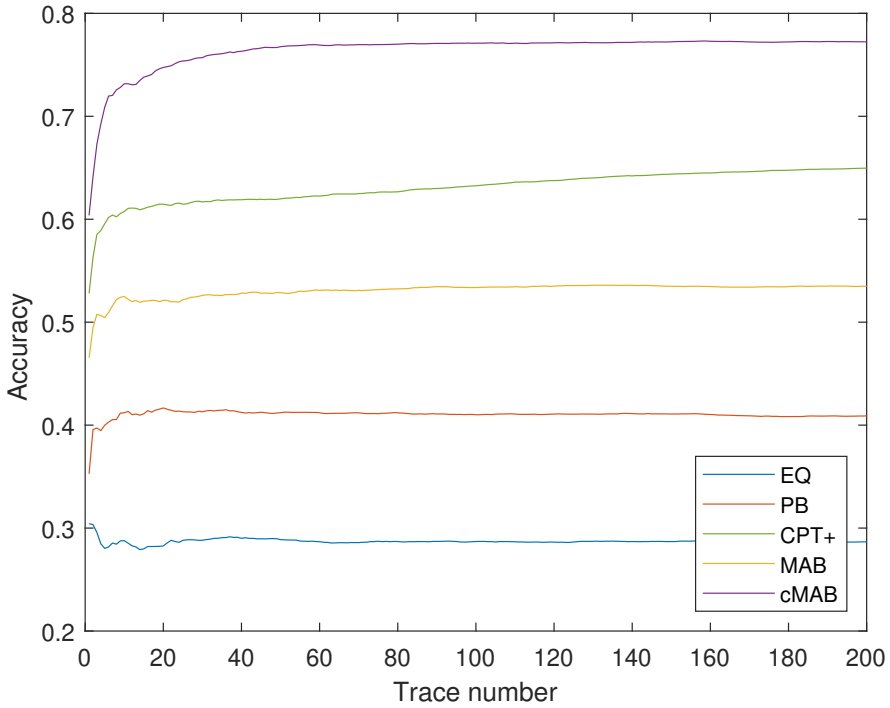


Figure 5.4 Cumulative prediction accuracy of the proactive caching systems in Las Vegas

Figure 5.4 shows the prediction accuracy (or hit ratio) in a cumulative way over the test traces. The accuracy superiority of bandit learning schemes over the PB and EQ is linked to the uncertainty CDF shown in Figure 5.3. For instance, in Figure 5.3b, 90% of RSUs in the cMAB system have less than 0.5 bits of prediction entropy and therefore the highest prediction accuracy, compared to the PB system with an accuracy of 0.4 where only 5% of RSUs have less than 0.5 bits of entropy. Another

practical point to explain this is that in bandit learning based schemes, RSUs make their decision on Q-values and the goal is to maximise the rewards. Therefore, fewer attempts are wasted on those actions that are less likely to be successful, whereas PB and EQ schemes, especially the latter one, attempt "bad" decisions more frequently. Individual examples shall be seen later. In addition, CPT+ is also shown in the figure, whose prediction performance is in between cMAB and MAB. In contrast to MAB, this makes sense since CPT+ relies greatly on a vehicle's past RSUs as a kind of context and this reduces the prediction uncertainty.

However, it is outperformed by the cMAB as a model-free scheme with only one type of context (i.e., previous RSU) required. The MAB scheme reaches its limitation of 53% at a much earlier stage compared to cMAB with an upper bound of nearly 80%. CPT+ seems to have an increasing trend after test trace 200 and it can be inferred that it would reach the performance of cMAB perhaps at test trace 500 because the performance of CPT+ depends on its model: the more data, the better model. However, this is also its limitation in terms of adaptability and flexibility. It is also observed that the introduction of contextual information helps RSUs make more accurate decisions throughout the simulation cycle and meanwhile, it takes relatively longer to fully train the model and converge due to this fact.

Although Figure 5.3 and Figure 5.4 have demonstrated the potential interaction between prediction accuracy and uncertainty reduction, different RSUs may show very different variations on these two metrics. In Figure 5.5, 4 types of RSUs are selected according to the size of their action set. From the top to the bottom row, these are 1 RSU with 5 actions, 8 RSUs with 4 actions, 10 RSUs with 3 actions and 8 RSUs with 2 actions, respectively. The left column is the uncertainty CDF of the corresponding group of RSUs in an aggregated way at the end of the simulation and the right column is the cumulative prediction accuracy of these RSUs. For example, there are 8 RSUs with 4 actions i.e., neighbours in the network of Las Vegas. To achieve the plot on the left-hand side, their uncertainty is computed and collected at the end of each test trace, resulting in 200 by 8 samples for the CDF plot. Similarly, the right-hand

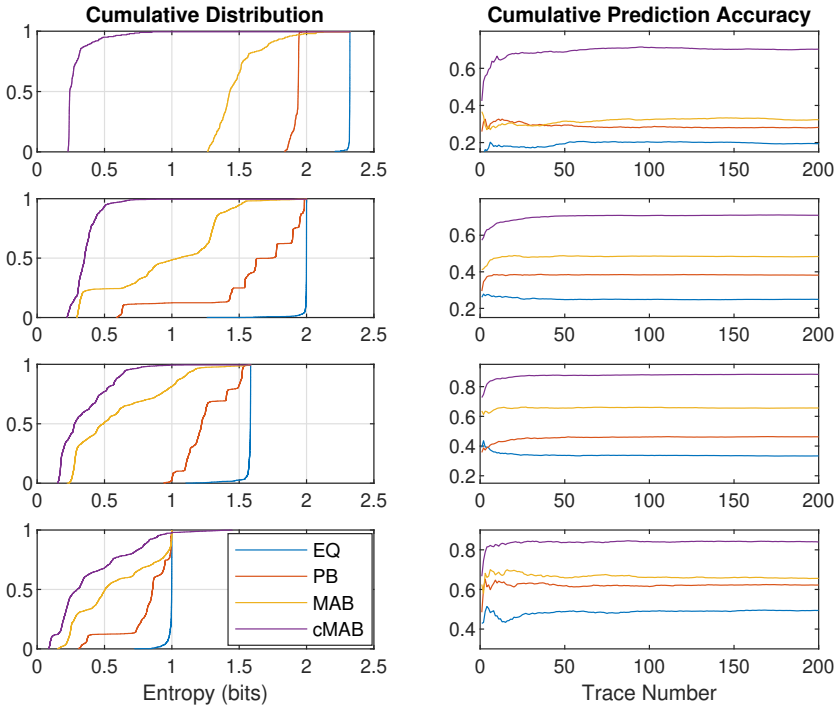


Figure 5.5 Performance of RSUs with different numbers of actions at the end of the simulation in Las Vegas. From the top to bottom, they are RSUs with 5, 4, 3 and 2 actions, respectively. The left column is the CDF of uncertainty (entropy) of these RSUs and the right column is the cumulative accuracy. The same legend is shared by the two columns. The significance of the figure is that it demonstrates that smaller uncertainty results in higher accuracy (horizontally).

column shows the cumulative prediction accuracy of the corresponding RSUs also in an aggregated way. The prediction accuracy of test trace 10 of these 4-action RSUs is $\frac{\sum_1^8 \sum_1^{10} Q_{correct}}{\sum_1^8 \sum_1^{10} Q_{prediction}}$. Both columns share the same legend shown in the bottom left corner. The observation observed from Figure 5.3 and 5.4 can still be seen in Figure 5.5 by looking at it horizontally, where for RSUs with the same number of actions, the superiority in entropy distribution on the left reflects a better performance in the cumulative prediction accuracy. Although it may be difficult to quantify the benefits of the reduction in uncertainty to prediction accuracy at this point, it helps visualise such benefits.

		EQ				PB			
		Action 1		Action 2		Action 1		Action 2	
		successes	failures	successes	failures	successes	failures	successes	failures
RSU 2	Count	215	52	45	201	333	71	32	104
	Success Rate per Action	81%		18%		82%		24%	
	Overall success rate	51%				68%			
RSU 22	Count	219	251	228	222	323	357	134	114
	Success Rate per Action	47%		51%		48%		54%	
	Overall success rate	49%				49%			
		MAB				cMAB			
		Action 1		Action 2		Action 1		Action 2	
		successes	failures	successes	failures	successes	failures	successes	failures
RSU 2	Count	419	100	8	18	404	1	90	4
	Success Rate per Action	81%		31%		100%		96%	
	Overall success rate	78%				99%			
RSU 22	Count	223	253	226	215	265	18	467	143
	Success Rate per Action	47%		51%		94%		77%	
	Overall success rate	49%				82%			

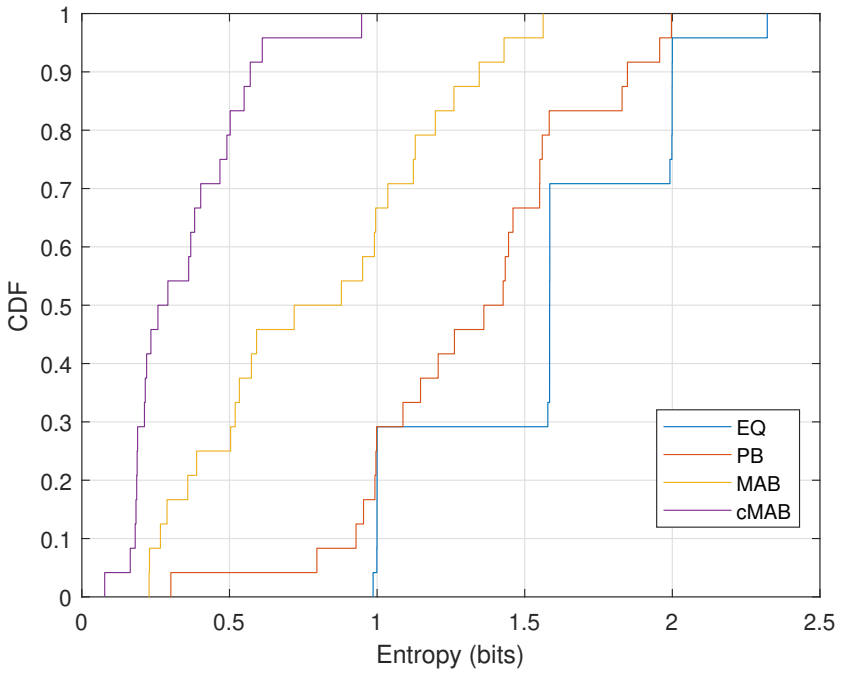
Figure 5.6 Statistics of two RSUs in Las Vegas. The table in this figure shows the accuracy of two RSUs with two actions as well as the improvement in prediction accuracy among the four proactive caching systems.

Even with the same number of neighbours, RSUs may show completely different performance in terms of uncertainty and prediction accuracy, possibly depending on their geographical location, traffic patterns, connectivity patterns, etc. In Figure 5.6, RSU 2 and RSU 22 are selected from the map in Figure 4.4a, both of which have two neighbouring RSUs (actions to be more precise) with unbalanced traffic. Over the 200 test traces, there are 73% and 27% of the 1116 handovers from RSU 2 to its two neighbours respectively, and RSU 22 also has the same proportion based on 2872 handovers. Despite this, proactive caching schemes have shown significantly different performance on these two RSUs and some statistical data in the end of the simulation

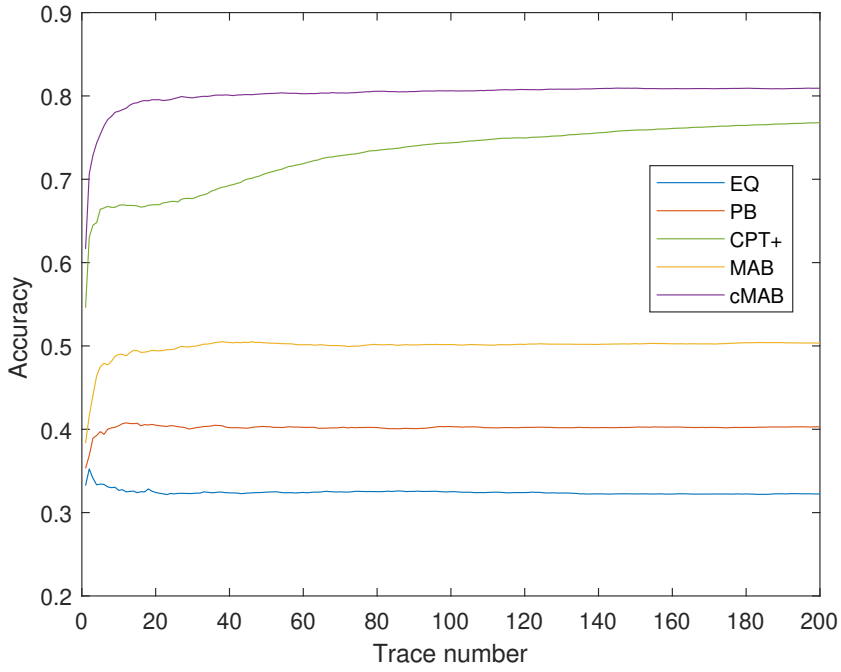
is summarised in the table of Figure 5.6. Without additional context introduced, there is an unknown inherent success rate of each action for non-contextual schemes (EQ, PB, and MAB), denoted as θ^*. For the action 1 of RSU 2, θ^* can be approximately 80% according to the table as the success rates of all the three schemes tend to converge to 80%. For action 2, however, there does not seem to have a clear converging success rate, but it could be 18% as in EQ scheme. The reason that PB and MAB have higher performance for action 2 is because they have fewer selections on action 2 than EQ, referred to the "Count" row. Precisely because of this, during the learning process, MAB leans towards action 1 as it tends to have a better Q-value than action 2 and hence much fewer wrong decisions are made, resulting a 78% overall accuracy. On the other hand, θ^* for action 1 and action 2 of RSU 22 is tending to converge to somewhere around 50%. Consequently, the MAB scheme is unable to tell which action would be a better one as they both have similar Q-values and it shows basically the same prediction performance as EQ and PB.

It is obvious that the introduction of additional contextual information in cMAB has dramatically increased not only the success rate of each action of RSU 2 and RSU 22 but also their overall prediction accuracy to 99% and 82%, respectively. In particular, compared to its counterpart MAB, it has resolved the dilemma with RSU 22 where both actions have similar inherent θ^*. Instead of "hesitating" between the two actions, RSU 22 learns policy under different contexts in cMAB and becomes more certain about which action is likely to be correct. This is even more convincing for the case of RSU 2, where both actions have over 96% accuracy because the actual traffic going through RSU 2 has two directions: RSU 29 $\rightarrow$ RSU 2 and RSU 6 $\rightarrow$ RSU 2 (referred to Figure 4.4a) and cMAB successfully detected RSU 29 and RSU 6 as contexts, optimising its policy.

The system performance of Manchester is shown in Figure 5.7, as a secondary city for generalising the application. Similarly, the distribution of uncertainty among RSUs of four systems at the end of test trace 200 is shown in Figure 5.7a and the cumulative prediction accuracy of all five systems in Figure 5.7b. Bandit learning-based



(a) CDF of Uncertainty at trace 200



(b) Cumulative Prediction Accuracy

Figure 5.7 Cumulative distribution function (CDF) of the overall uncertainty of four proactive caching systems at the end of simulation and prediction accuracy of all the systems in Manchester.

systems still show comparable benefits to that in Las Vegas, especially cMAB whose prediction accuracy has reached 80%. The performance has successfully demonstrated the adaptability of the proposed bandit learning schemes in a relatively more complex transportation network. One of the reasons for this is that the proposed algorithms only rely on information from the vehicular network itself for proactive caching decisions instead of taking additional information from the road network. Despite the advantages over the other two non-contextual systems (EQ and PB) as before in Manchester, the performance limitation of non-contextual MAB can be noticed in contrast to its counterpart MAB scheme. CPT+ still shows similar relative performance to cMAB and MAB but has a faster growth rate compared to Las Vegas. This might be because of the relative area size and traffic pattern difference between the two cities (which will be explained in detail shortly).

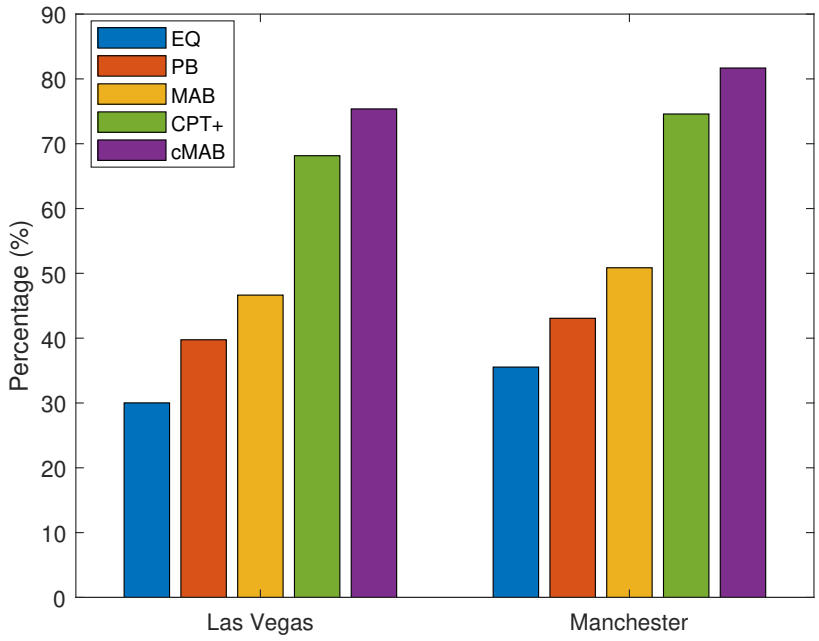
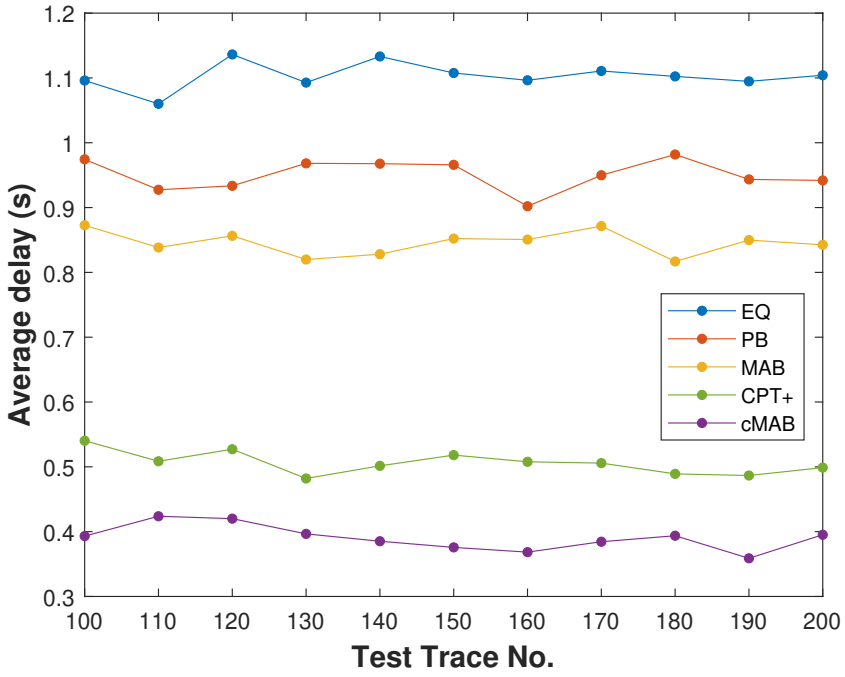


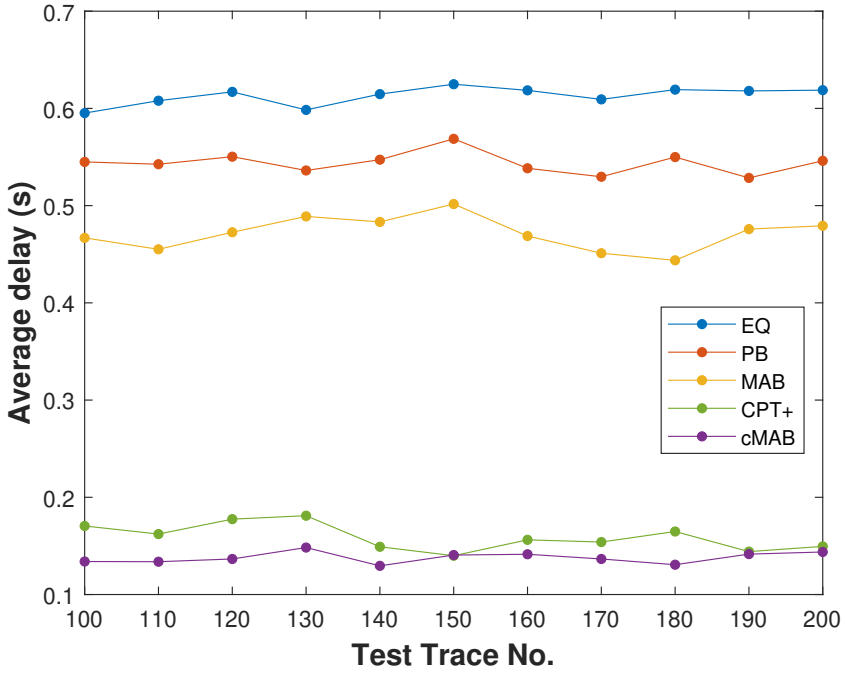
Figure 5.8 Average percentage of the number of fragments served by proactive caching in the last 100 test traces of the simulation. The figure illustrates that higher cumulative prediction accuracy results in better proactive caching performance reflected by higher proportions of content fragments through caches.

One of the major objectives of proactive edge caching in vehicular networks is to provide vehicular users with seamless content delivery by bringing the content close to them accurately. The amount of fragments transmitted directly from RSU caches to vehicular users is measured and is plotted in a bar chart, the *proportion of the average fragments served by proactive caching*, for each of the proactive caching schemes of two cities in Figure 5.8. Overall, the proportions of both cities are consistent with the cumulative prediction accuracy, and the cMAB scheme demonstrates remarkable superiority over the other four. On average, it has achieved 75% in Las Vegas and 81% in Manchester, nearly double that of EQ and PB systems. It can also be concluded that the proposed proactive caching algorithms perform similarly irrespective of the road topology. Notably, the proportions in the two cities are based on different absolute total numbers of fragments transmitted to vehicles (around 1300 in Las Vegas and 750 in Manchester, varying trace by trace). This is because a) the Manchester area is relatively smaller than the Las Vegas area as a whole, b) the connectivity patterns of the two cities are distinct, and c) vehicles' content request pattern and frequency are different from test trace to trace of two cities. However, as the relative size of the centre of the two areas has been kept at a similar level, this is still an effective contrast.

For consistency with Figure 5.8, Figure 5.9 shows the average delay of each set of 10 test traces over the last 100 test traces in two cities. In other words, the average delay of the system for a particular test trace in Figure 5.9 is the mean of the previous 10 test traces. As shown in Equation (5.12), it is computed by adding up all the fragments served by the backhaul network in the past 10 test traces, dividing the outcome by the total number of times when the delay was introduced due to cache miss, multiplying it by the constant size of one fragment (100 MB), and dividing the result by the backhaul transmission rate ω. It can be seen that the performance of the five proactive caching systems in the two cities in Figure 5.9 is consistent with Figure 5.8 as well as the cumulative prediction accuracy in Figure 5.4 and Figure 5.3b. The cMAB system still outperforms the other systems under comparison in network delay due to its excellent accuracy in prediction. The absolute levels of the delay in the two cities are different



(a) Network Delay in Las Vegas



(b) Network Delay in Manchester

Figure 5.9 Average delay on test trace basis. This figure shows the average delay in the system introduced by all the vehicles due to cache misses in every 10 test traces from test traces 100 to 200. Thus, each point of the line is computed by averaging the average delay of the previous 10 traces.

because vehicles travel a shorter distance in Manchester compared to Las Vegas and thereby less overall delay, as explained in the performance evaluation of Chapter 4.

5.5 Discussion

5.5.1 Theoretical Analysis

Theoretically, the two proposed MAB-based algorithms have advantages in terms of computational complexity. For the three non-contextual algorithms i.e., MAB, Equal-probability and Probability-based, the Probability-based one has the highest computational complexity. This is because RSUs in this algorithm require some extra computational resources to store historical traffic information in order to establish a probability distribution over their actions. However, the MAB algorithm is an in-place algorithm where the RSUs' Q-tables get in-place updates, and individual RSUs have their own fixed probability distribution for prediction in the Equal-probability algorithm. Although cMAB algorithm is also an in-place algorithm as MAB, it does require RSUs to build context-related Q-tables and therefore, needs slightly more space than MAB. Nevertheless, this is worthwhile given the significantly reduced uncertainty and improved prediction accuracy by cMAB. CPT+ based algorithm, however, consumes the most resources because it requires building a large prediction tree model to achieve a certain prediction accuracy, which is still outperformed by cMAB.

In addition, the theoretical accuracy of these algorithms can be discussed. Assume a vehicle v connecting to a RSU m with N neighbours (actions). There exists an unknown probability distribution of v actually going to the N neighbours after m, denoted as $\mathcal{A} = [a_1, a_2, \ldots, a_n], n \in N$ and $\sum_{n \in N} \ a_n = 1$. If the RSU m makes a prediction with $\mathcal{B} = [b_1, b_2, \ldots, b_n], n \in N$, then the chance that this is a correct prediction can be simply computed by $\mathcal{P} = \mathcal{A} \ \cdot \mathcal{B} = \sum_{n \in \mathcal{N}} a_n \times b_n$. Depending on which algorithm, $\mathcal{B}$ is different. In the most under-performed one i.e., Equal-

probability algorithm, $\mathcal{B}$ is uniform distribution i.e., $b_1 = b_2 = \ldots = b_n = \frac{1}{N}$ and thus $\mathcal{P} = \sum_{n \in \mathcal{N}} a_n \times b_n = \frac{1}{N} \times \sum_{n \in \mathcal{N}} \ a_n = \frac{1}{N}$. In the Probability-based algorithm, $\mathcal{B}$ is the transition probabilities derived from previous traces, where $b_1 \neq b_2 \neq \ldots \neq b_n$, and therefore $\mathcal{P}$ remains to be $\mathcal{P} = \sum_{n \in \mathcal{N}} a_n \times b_n$. If the traffic pattern through RSU m does not change significantly over time, $a_n \approx b_n$, so $\mathcal{P} = \sum_{n \in \mathcal{N}} \ b_n^2$. In non-contextual MAB, $\mathcal{B}$ depends on Q-values and action selection algorithm (i.e., ϵ-greedy). Therefore, the probability b_n of its neighbour $n \in \ N$ to be predicted as the next RSU is: $b_n = \begin{cases} 1 - \epsilon, & \text{if } n \text{ has the highest } Q\text{-value} \\ \epsilon \cdot \frac{1}{N}, & \text{Otherwise} \end{cases}$. Take an example of an RSU with two action choices (neighbours) with uneven traffic patterns (e.g., 80% vs 20%). Its theoretical accuracy with Equal-probability algorithm is 50% since it has 2 neighbours. Because it has an uneven traffic pattern where one of its neighbours has approximately 80% traffic, the theoretical accuracy with Probability-based algorithm can be calculated as $\mathcal{P} = 80\% \times 80\% \ + \ 20\% \times 20\% = 68\%$. It is because of this traffic pattern that MAB has a dominant action and therefore, the overall theoretical accuracy is $\mathcal{P} = 80\% \times (1 - \epsilon) + 20\% \times \frac{\epsilon}{2} = 77\%$, where $\epsilon = 0.05$. The cMAB algorithm further expands the advantage of MAB and reduces uncertainty by breaking it down into context levels, hence resulting in an even higher optimal boundary. The simulated result of RSU 2 in Figure 9 is consistent with the theoretical values and this can be extended to other RSUs with different numbers of choices.

Furthermore, another notable theoretical advantage of the proposed cMAB and MAB algorithms is their natural capabilities of coping with sudden major changes in the topology or vehicular environment by rapidly adjusting Q-tables and policies, whereas Probability-based and CPT+ based algorithms, become very clumsy in this regard due to high reliance on past data to establish their models.

5.5.2 Time Complexity

The three main functionalities in the proposed MAB and cMAB algorithms are: A - **Next RSU selection** (including ϵ -greedy), B - **Pre-caching content** and C - **Q-table updating with rewards**. From the perspective of actual code implementation, for the MAB algorithm, an agent RSU with k actions requires $O(k)$, $O(1)$, and $O(k)$ time complexity for function A, B and C respectively. This is because functions A and C require action set traversal whereas B only needs insertion manipulation with a vector. In addition, as functions A, B and C are executed sequentially, they account for a $O(k)$ complexity. The system may have multiple RSUs but due to the nature of event-driven simulation, only one of them is "working" at a time. Therefore, assuming the largest action set of these RSUs is K, then the overall performance of N-length test can be represented by $O(NK)$. The major difference between the two lies in the additional context s. Specifically, functions A and C are executed based on s once it is detected. But this works in the same way as in a non-contextual MAB and therefore, their complexity is identical to that in MAB for an arbitrary RSU. Function B remains the same as well. Apart from this, the cMAB algorithm also involves context detection and creation and these additional manipulations account for $O(1)$ complexity. Thus, cMAB has the same overall complexity, that is $O(NK)$ as above.

5.5.3 Convergence

The cumulative prediction accuracy in Figure 5.4 and Figure 5.7b demonstrates the convergence of the proposed MAB-based proactive algorithms. Although a cumulative way to show this may not be perfect, it is still sufficient to illustrate the performance boundary in the commuting traffic scenario that is considered. From the system level, theoretically, the cMAB algorithm should converge slower than the non-contextual MAB because given a statistically fixed number of Q-table updates (identical test traces) for an RSU fewer updates are allocated to each individual context in cMAB in contrast to MAB where all the updates are used for only one Q-table. Such a difference

in convergence rate can be noticed in Figure 5.4 and Figure 5.7b, though the difference is not significant.

During the learning process, Q-values or Q-tables of individual RSUs may converge to rather different values depending on the traffic pattern through it. For example, in the non-contextual MAB algorithm, it has been noticed that a high-accuracy RSU (over 90%) with 4 actions has a converged Q-table with values: $\langle -0.9375, -1, -0.9961, 1\rangle$ at an early stage of the learning process. This demonstrates a convergence to the last action and that there may exist a very determining route of all the vehicles through this RSU. On the other hand, it has also been found that an average-accuracy RSU (approximately 50%) with the same number of choices has a Q-table with values: $\langle -1, -1, -0.5643, -0.4379\rangle$. Throughout the learning process, the RSU tried to converge to the best action by trial and error but failed to do so because the last two actions are almost evenly good. This implies the dilemma in non-contextual MAB and should be resolved by contextual MAB with additional contexts.

5.6 Extending the cMAB System to Two Dimensions

The earlier sections have shown the outstanding prediction performance by cMAB based on previous RSU. The introduction of the previous RSU that a vehicle visited has dramatically reduced the uncertainty during simulation but the prediction performance limit still exists. Thus, the motivation of this section is to seek further improvement in the prediction accuracy of the earlier cMAB system. An intuitive improvement could be introducing additional context to the earlier cMAB system that only uses the previous first RSU as context. A straightforward choice of the additional context is the previous second RSU which is similarly as accessible as the previous first RSU, hence the previous two RSUs as context. The combined context can be thought of as a *two-dimensional* context and the previous one RSU based context is a *one-dimensional*

context. The purpose of this section is to show the results of the two-dimensional cMAB system in Las Vegas and compare the two cMAB systems. To distinguish, in the following, the two-dimensional cMAB system will be referred to as *Prev2RSU* cMAB and the one-dimensional cMAB system proposed in the prior sections will be named *Prev1RSU* cMAB.

The Prev2RSU cMAB system has little difference in nature from the Prev1RSU cMAB system. In other words, the way of designing a cMAB system discussed in Section 5.2.2 & 5.2.3 still applies. The additional effort that an agent RSU needs is to detect the second-to-last RSU and combine it with the last RSU and form a two-dimensional context. Therefore, the detection of the context s in Algorithm 2 now becomes the detection of two previous RSUs.

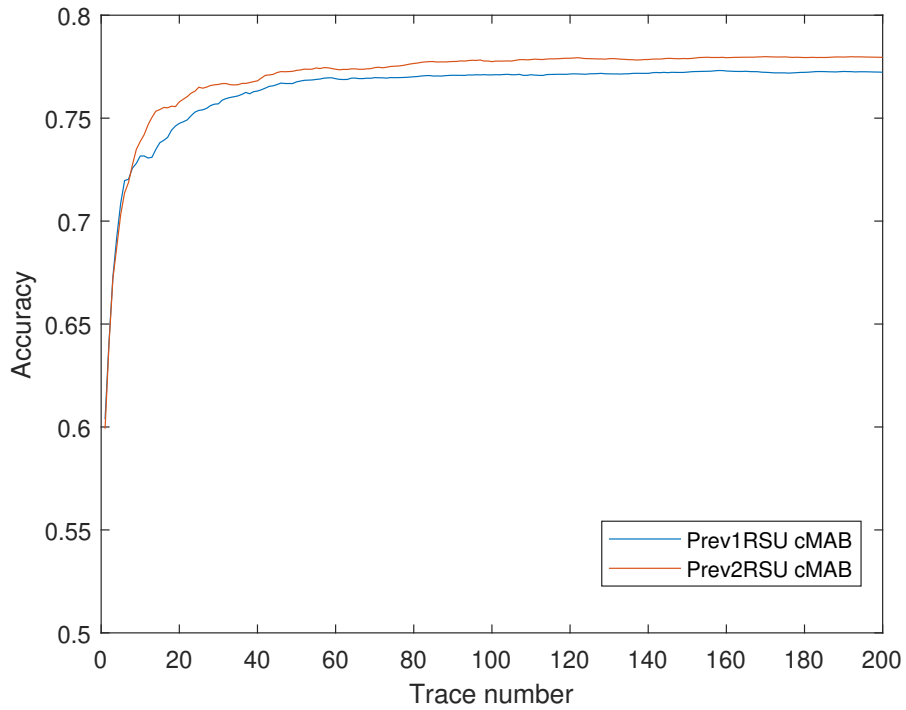


Figure 5.10 Prediction performance comparison of the two cMAB systems with Las Vegas data. Prev1RSU cMAB is the cMAB system that utilises only the previous RSU and Prev2RSU cMAB uses the previous 2 RSUs

Figure 5.10 shows the cumulative performance of the two types of cMAB proactive caching systems in the Las Vegas area. Clearly, there is some improvement (nearly 1%) in the overall prediction accuracy but it is not as significant as that brought about by the Prev1RSU cMAB on top of the non-contextual MAB system as shown in Figure 5.4. This is mainly because the situations where the Prev2RSU context could make a difference are limited, which we will now examine in more detail.

Figure 5.11 shows an example of such situations. The figure illustrates a partial handover

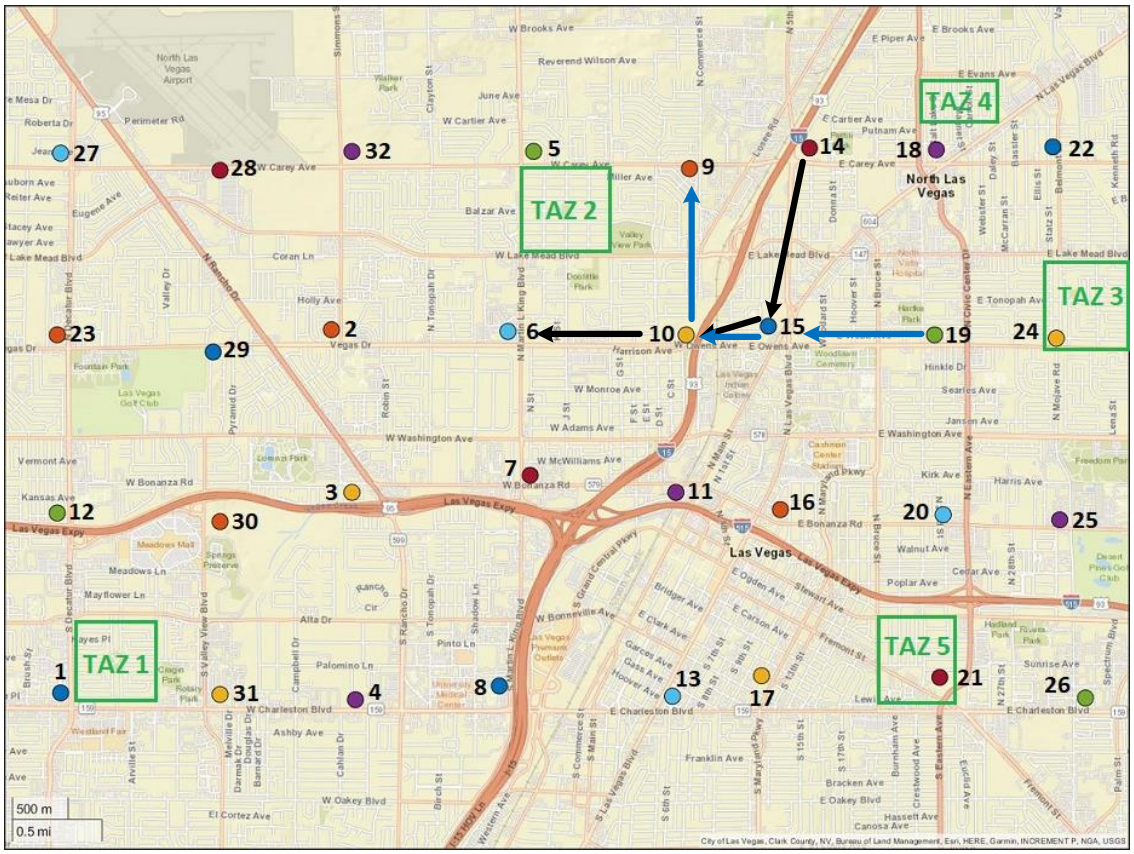


Figure 5.11 An typical scenario where the system can benefit greatly from the Prev2RSU cMAB scheme

order before and after vehicles connect to RSU 10 in the late stage of the test. The blue and black arrows illustrate an order of $(19 \rightarrow 15 \rightarrow 10 \rightarrow 9)$ and $(14 \rightarrow 15 \rightarrow 10 \rightarrow 6)$, respectively. Therefore, for the agent RSU 10 in the Prev2RSU cMAB system, it will have two contexts: $\langle 19, 15 \rangle$ and $\langle 14, 15 \rangle$. The Q-values at the end of the learning

process of the two contexts, $[6 : -0.8802, 9 : 0.9325, 11 : -1.0000, 15 : -1.0000]$ and $[6 : 0.3411, 9 : -0.9675, 11 : -1.0000, 15 : -0.9999]$ reflects the distinct route shown with the arrows. Distinguishing contexts with a positive action value, especially close to 1, is important for accurate predictions. In contrast, RSU 10 under such situation in Prev1RSU cMAB system only has one context $\langle 15 \rangle$ with converged Q-values of $[6 : -0.8063, 9 : -0.9922, 11 : -1.0000, 15 : -1.0000]$, which implies inaccurate predictions. However, such a situation is not common and is related to the geographical locations of the involved RSUs as shown in Figure 5.11, where they are all close to TAZs. When vehicles are reaching a TAZ, they may arrive at different ending points inside it, creating diverse routes. On the other hand, in many other situations, Prev2RSU context does not help the agent RSU more than Prev1RSU context to further reduce its uncertainty about its actions. For example, RSU 2 is found to have a context $\langle 6 \rangle$ with Q-values of $[6 : -0.5000, 29 : 1]$ in Prev1RSU cMAB, but the two contexts $\langle 5, 6 \rangle$ and $\langle 10, 6 \rangle$ it establishes in Prev2RSU cMAB also have converged Q-value $[6 : -0.5000, 29 : 1]$ and $[6 : 0, 29 : 0.9922]$, respectively. This means no gain is achieved by implementing Prev2RSU cMAB.

5.7 Conclusion

Proactive edge caching is deemed to be an important functionality to improve user experience in 5G and beyond wireless networks. Vehicular networks can particularly benefit from this due to their rapidly changing topology. This chapter studied how to cache the content at the next RSU in a proactive way. As a way of addressing this, the chapter has proposed two bandit learning-based proactive caching algorithms: *non-contextual* MAB and *contextual* MAB and compared their performance with three other baseline schemes: *Equal Probability-based*, *Probability-based*, and *Compact Prediction Tree+ based* proactive caching strategy. In addition to this, the *subjective logic* framework has been extended to study the uncertainty associated with different proactive caching systems. With this framework, the overall entropy distribution of

the systems as well as the distribution of representative RSUs have been analysed in detail. Furthermore, two urban areas of Las Vegas and Manchester with different road layouts have been tested to demonstrate the adaptability of the proposed schemes to a diverse set of road layouts.

Numerical results have shown the advantages of the proposed proactive caching algorithms over their counterparts. Contextual MAB-based scheme yields the highest benefit to the system thanks to the introduction of contextual information for uncertainty reduction. In both cities, the contextual MAB-based proactive caching scheme reached a prediction accuracy of approximately 80% compared to roughly 50% of the non-contextual MAB-based scheme. As a result of this, the network performance was dramatically improved with contextual MAB in terms of the number of fragments directly transmitted by caches. The performance of bandit learning-based systems was similar in both cities regardless of road topology. Particularly, 75% and 81% content fragments were proactively served with contextual MAB algorithm and over 53% and 50% with non-contextual MAB algorithm in Las Vegas and Manchester, respectively.

Last but not least, the end of the chapter has extended the cMAB system to a two-dimensional cMAB system and compares the prediction performance of two-dimensional cMAB that implements the previous 2 RSUs as context, to the proposed cMAB that implements the previous RSU as context. Through multiple examples, the section demonstrates the gain limitation of Prev2RSU cMAB and the reason for it in the traffic scenario considered in this chapter. On the other hand, this extended work has also motivated the work in the next chapter to further explore two-dimensional cMAB system that implements additional context from a different domain.

Chapter 6

A Hybrid Proactive Caching System in Vehicular Networks based on Contextual Multi-armed Bandit Learning

6.1 Introduction

Chapter 5 has elaborated on the advantage of using contextual multi-armed bandit (cMAB) learning to help address the proactive caching problem. In the contextual MAB-based algorithm, the use of the prior RSU which a vehicle comes from helps the decision-making RSUs build distinguishing Q-tables and solve independent bandit tasks. This resulted in a reduction in uncertainty. However, despite its advantages, the cMAB algorithm based only on previous RSUs has its limitations. This is because there are still some situations where agent RSUs may face two actions with close Q-values, even with the assistance of the previous RSU as a potential source of the incoming direction. For example, a converged Q-table $\langle -1, -1, -0.5643, -0.4379 \rangle$ in a non-contextual MAB shown in Section 5.5.3 may also happen to a particular context in cMAB, although the likelihood of this has been greatly reduced. In fact, this is the limitation of the *one-dimensional* context or *single-context*. This has motivated the work in this chapter to explore *two-dimensional* context or *dual-context* so as to further resolve the above dilemma, and inspired by the extended work in Section 5.6, the dual-context should come from different domains to potentially maximise the performance. Nevertheless, the novelty of this chapter is not only the proposal of using two-dimensional context in cMAB but more importantly, is the design of a hybrid system with a switching mechanism to take full advantage of cMAB with contexts in different dimensions so that more accurate predictions are achieved.

Concretely, the objective of this chapter is to address the proactive caching problem in vehicular networks using the cMAB learning. To achieve higher accuracy of the next RSU prediction, a *Hybrid cMAB Proactive Caching System* which enables adaptive switching between its two underlying prediction algorithms, *Dual-context cMAB* and *Single-context cMAB*, has been developed. The motivation here is to design a hybrid system that can fully exploit the potential of both dual-context and single-context cMAB in order to seek better proactive caching performance in a variety of scenarios. The traffic scenarios considered in this chapter are also more realistic and comprehensive

and aim to provide an all-round perspective of the performance and application of the proposed hybrid cMAB systems.

The rest of the chapter is structured as follows. Section 6.2 covers the elaboration of the proposed hybrid cMAB system as well as the two parallel cMAB-based prediction algorithms. Section 6.3 introduces the modified simulation environment and traffic scenarios considered in this work. Section 6.4 demonstrates and analyses the simulation results. Section 6.5 investigates and discusses further the limitations of cMAB algorithm. Section 6.6 concludes this chapter.

6.2 Design of the Hybrid Proactive Caching System

The first focus of this section is to introduce the designed hybrid cMAB proactive caching system. Then the underlying dual-context cMAB and single-context cMAB prediction algorithms will be elaborated on in more detail.

6.2.1 Hybrid cMAB Proactive Caching System

The topic of this subsection is to introduce the design of the proposed *Hybrid cMAB Proactive Caching System* (HCPC) used for proactive caching. The basic concept behind the hybrid system is that it implements a switching mechanism that allows an RSU to adaptively finalise its prediction between two cMAB-based prediction algorithms: ***Single-context cMAB*** and ***Dual-context cMAB*** algorithms. In general, the agents in cMAB problems use context to help choose which action to play in the current iteration. The context observed is actually an *N-dimensional* context, where each dimension is a source of side information that may or may not be the same type. Therefore, single-context cMAB is a *one-dimensional* cMAB problem where the agent only observes one source of information (e.g., previous RSU) to consider as context. The agent in dual-context cMAB, however, is able to detect information from

two sources (e.g., previous RSU and vehicle ID), together forming a *two-dimensional* context.

The single-context cMAB that makes use of the *previous RSU* as the context has been exploited in Chapter 5. As one of the underlying prediction algorithms in the HCPC system, it is enhanced in this chapter with a Win-or-Learn-Fast variable learning rate. The advantage of single-context cMAB is that it has sufficient learning opportunities for every related context s in the early stage of learning, but in some situations, it may still suffer from a similar dilemma as in the non-contextual MAB problem even though the previous RSU provides a good reference to a vehicle's incoming direction (which will be detailed in the next subsection). On the other hand, the dual-context cMAB designed in this chapter exploits a two-dimensional context that consists of *vehicle ID* and *previous RSU*. It reinforces the single-context cMAB and could result in a more explicit context for an agent RSU to distinguish different tasks. Nevertheless, its disadvantage is the shortage of learning samples in the early stages, since a vehicle passes through an RSU from a particular prior RSU only once a day. Therefore, the motivation behind the HCPC system is to combine the advantages of both in order to ensure the accuracy of the prediction as much as possible. The designed switching mechanism is the enabler of adaptive selection between single-context and dual-context, depending on the comparison of their historical prediction performance. In the meantime, it guarantees a lower bound on its prediction performance, i.e., single-context cMAB.

A complete procedure of an RSU selecting the next RSU as the proactive caching node in the HCPC system starts when a vehicular user connects to the RSU. It makes two predictions (performs two action selections) with dual-context and single-context cMAB algorithms, respectively, denoted as P_D and P_S. It then performs the switching mechanism to finalise its decision $P_F \in \{P_D, P_S\}$ and sends its proactive caching request to the predicted RSU (i.e., P_F). In other words, the final decision can also be seen as the result of either dual-context cMAB or single-context cMAB.

The key point in the switching mechanism is the way to compare the historical prediction accuracy of the two cMAB algorithms. One thing to consider in the comparison is whether the RSU extracts its past predictions made for all the vehicles that have connected to it or just the prediction data of the current vehicle, which corresponds to ***RSU-Centric*** and ***Vehicle-Centric***, respectively. In the HCPC RSU-Centric system, the RSU finalises its prediction (P_D or P_S) for all of the connecting vehicles, once it computes which cMAB algorithm may benefit its overall prediction performance in the current simulation cycle. On the other hand, the RSU in the HCPC Vehicle-Centric system does this on the vehicle level. It uses the past prediction performance of this particular vehicle to compute and determine what is the best option for the vehicle in the current cycle. The advantage of Vehicle-Centric system is that it allows "customization" for different vehicular users, which will intuitively benefit individual users because the best decision is customised for them. The two systems use different *window sizes* (WS) for backtracking length to calculate past prediction performance because for a Vehicle-Centric system, to obtain a similar past prediction sample size it needs longer backtracking length i.e., larger WS than RSU-Centric system. The switching mechanism of HCPC system is summarised in **Algorithm 3** and meanwhile, a comprehensive flow of the system in the flowchart is shown in Fig. 6.1.

In a proactive caching-enabled vehicular network, the objective is to realise seamless content delivery to vehicular users. This is achieved by a high cache hit ratio which relies on accurate mobility prediction. Therefore, achieving high prediction accuracy is the objective of the hybrid cMAB proactive caching system. In the following, the detailed implementation and design of the two parallel cMAB prediction algorithms will be discussed.

6.2.2 Parallel cMAB-based Mobility Prediction Algorithms

Finding the best RSU to pre-cache relevant content for a vehicular user is a matter of mobility prediction. It is crucial that the currently associated RSU is able to predict the

Algorithm 3: Switching mechanism in Hybrid cMAB Proactive Caching System

```
while not the end of the test do
    if Vehicle V_u connects to RSU m then
        Predictions by parallel algorithms:
        P_D ← Dual-context cMAB;
        P_S ← Single-context cMAB;
        Finalise prediction P_F - switching scheme:
        Vehicle-Centric System: Extract past predictions of V_u made by RSU m
         in the last WS tests;
        RSU-Centric System: Extract past predictions of all vehicles made by
         RSU m in the last WS tests;
        Compute cumulative average accuracy:
        Acc_D ← Dual-context cMAB;
        Acc_S ← Single-context cMAB;
        if Acc_D > Acc_S then
            P_F ← P_D;
        else
            P_F ← P_S ;
        end
    end
end
```

next possible RSU the vehicle is about to access, as accurately as possible. As discussed earlier, a cMAB problem is composed of an action set, context set, and rewards. By taking appropriate actions, the agent hopes to maximise its payoff eventually. In the next RSU proactive caching problem, the currently connected RSU helps a vehicle to continue the unfinished content transmission immediately when it reconnects to a new RSU, provided that the new RSU has the requested content. This completely depends on whether the last RSU predicts or selects the correct RSU from its neighbouring RSUs. If it was a correct prediction, positive feedback is given; otherwise, negative feedback is generated. From this point of view, they resemble each other in terms of action (RSU) selection and reward (feedback) generation. How the mobility prediction is modeled as a single-context cMAB problem has been elaborated in both Chapter 5 and [114]. However, the proposed dual-context cMAB algorithm differs in terms of the dimension of context. The remainder of this subsection will focus on the composition

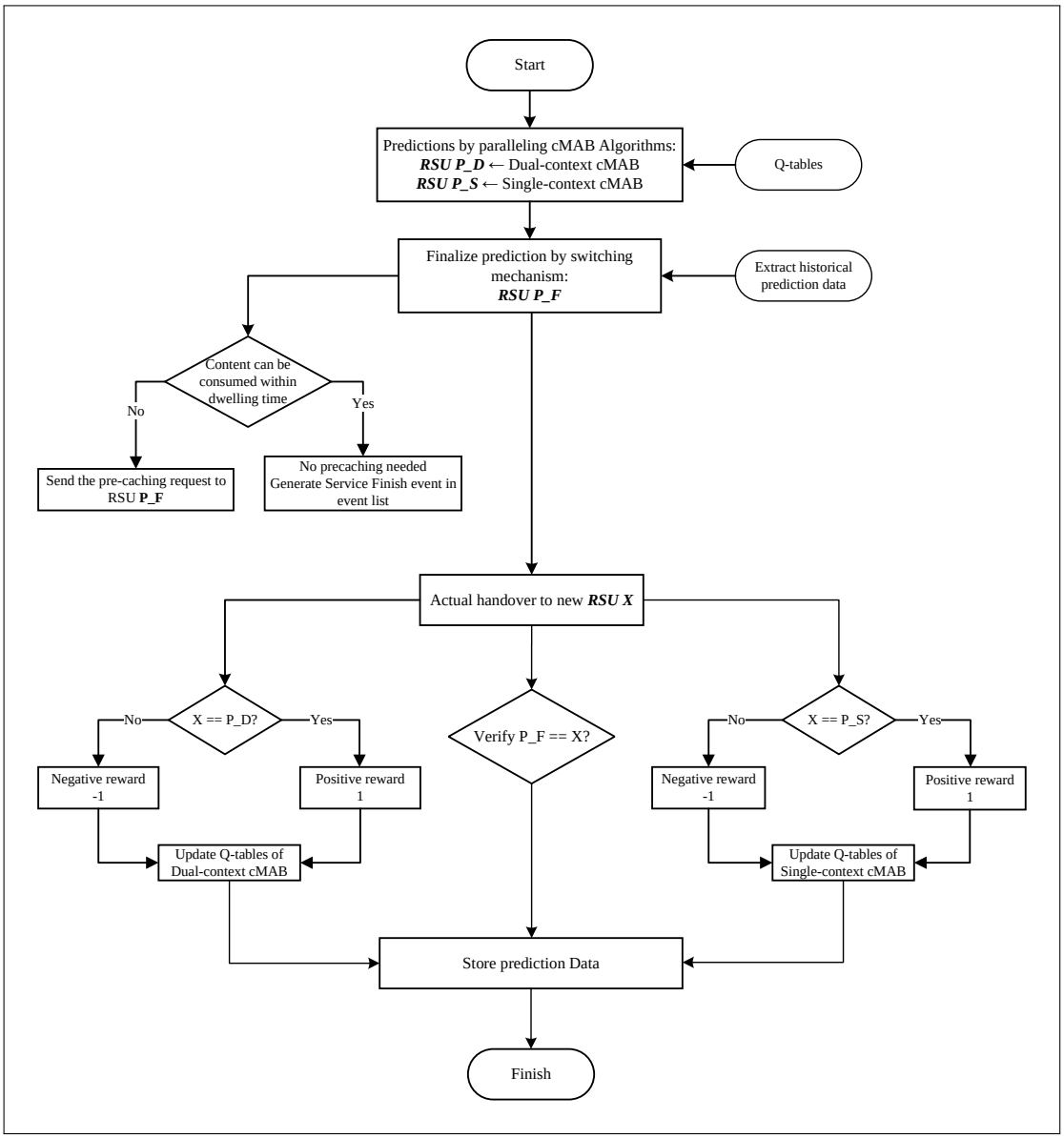


Figure 6.1 Flowchart of the Hybrid cMAB Proactive Caching System

This is a general cycle of an agent RSU serving a connecting vehicle, from *Start* when a vehicle connects to the RSU, to *Finish* when its action-value table is successfully updated with corresponding rewards and relevant prediction data is stored sufficiently.

of the context in the dual-context cMAB in contrast to the single-context cMAB and introduces how to solve them with Win-or-Learn-Fast variable learning rate.

1. **Context in cMAB** In cMAB problems, a specific Q-table that consists of multiple actions' quality values (Q-value) is associated with specific context

$s \in \mathcal{S}$. The agent aims to learn a Q-table of s. Generally, the purpose of introducing context is to help the agent make better decisions compared to a general MAB problem (i.e., non-contextual MAB). The effectiveness of single-context cMAB with the previous RSUs as the context has been proved in Chapter 5, since this information is a useful source to help RSUs distinguish the incoming directions of vehicles. Despite its excellent performance, there may still exist occasions where the RSU's actions have close Q-values, which results in high uncertainty and limits the prediction accuracy. Therefore, it is meaningful to investigate the performance of cMAB with additional context from a different dimension and this motivates the proposal of the Dual-context cMAB-based algorithm.

Specifically, the context in the ***Dual-context cMAB-based Mobility Prediction*** algorithm combines two-dimensional context i.e., *vehicle ID* and *previous RSU*. As in single-context cMAB, the information of previous RSUs is easily accessed and used as a reference to such directions compared to other sorts of information e.g., road information, vehicle angle, etc. Moreover, the use of vehicle IDs, sometimes referred to as OBU IDs in literature e.g., [12], as additional contextual information is also legitimate as the IDs are important and useful identifiers in the next generation vehicular networks. In both algorithms, when the agent RSU needs to predict the next RSU (action selection) for a newly connected vehicle, the vehicle's relevant context will first be identified, which corresponds respectively to vehicle ID plus previous RSU as dual context or previous RSU only as single context. The task of the agent RSU is to learn the action values associated with the identified context through trial and error. This enables the agent RSU to solve separate bandit tasks associated with them, thereby guaranteeing a more effective policy learned. Since the dual-context cMAB solution is tailored to a specific vehicle, in principle it is likely to provide more accurate predictions than single-context cMAB. The context of dual-context cMAB is summarised as follows:

- **Context** - *Previous RSU and Vehicle ID*: The previous RSU and the vehicle id together form the context in dual-context cMAB. The agent RSU can identify the previous RSU that a connected vehicle coming from and its ID. It then combines these two and retrieves the Q-table associated with combined context so that an action can be predicted properly according to the action selection algorithm. If there does not exist such Q-table, it will initialise one for the combined context and perform its decision.

2. **Mobility prediction** Mobility prediction (i.e., next RSU prediction) in the modelled cMAB-based prediction algorithms is essentially an action decision for an agent RSU. Action selection plays an important role in solving cMAB problems and is fundamentally based on the estimated true values of actions. In a cMAB problem, the learning agent learns its actions' quality values corresponding to a type of context through trial and error. $Q(a \mid s)$ is used to denote this value and name it Q-*value* as in Q-learning [22][94], where $a \in \mathcal{A}$ and $s \in \mathcal{S}$. The agent then uses the corresponding exploration-exploitation scheme (i.e., ϵ-greedy) to select the appropriate action based on their Q-values: the best action is selected with a probability of $1 - \epsilon$; Otherwise, with small probability ϵ, actions will be selected randomly with equal probability regardless of their Q-values.

$$A = \begin{cases} \arg\max_a Q(a \mid s), & 1 - \epsilon \\ random, & \epsilon \end{cases} \tag{6.1}$$

3. Q**-value update** For economy and clarity, the simplified term $Q(a)$ of $Q(a \mid s)$ is used to denote the Q-values of the actions under context s. In Subsection 3.5.2, the recursive action-value updating formula has been derived with *incremental implementation* [24]:

$$Q_{n+1} = Q_n + \frac{1}{n}(r_n - Q_n) \tag{6.2}$$

where Q_{n+1} is the value after the action a has been selected for n times.

Equation (6.2) is further generalised as follows by replacing the so-called *step-size* $\frac{1}{n}$ with a constant learning rate α. This is because vehicular networks are dynamic environments with varying traffic densities, which results in a *nonstationary* bandit problem. Therefore, recent rewards should be given more weight when updating action values.

$$Q(a) \leftarrow (1-\alpha)Q(a) + \alpha r \tag{6.3}$$

The Q-values of actions under a particular context $s \in \mathcal{S}$ are hence updated according to Equation (6.3).

The agent RSU accepts a reward after taking an action and observing its relevant outcome. The outcome is translated to a reward through the reward function R. In other words, given an action a taken at time step t and the observed outcome as b (which may or may not occur immediately), its reward can be computed with $r_t = R(b)$. In the cMAB-modelled mobility prediction problem, the outcome of an agent RSU predicting one of its neighbouring RSUs as the next possible RSU is either $b = True$ or $b = False$. Similar to Equation (5.3), the reward function R considered in this chapter is :

$$r = R(b) = \begin{cases} 1, & b = \text{True} \\ -1, & b = \text{False} \end{cases} \tag{6.4}$$

4. **Win-Or-Learn-Fast Variable Learning Rate** The learning rate α is a key parameter for any RL problems including cMAB. It has a significant influence on the dynamics of the learning process. A fixed learning rate for both positive outcomes and negative outcomes is often seen in the literature such as [112] and [115]. Bowling and Veloso proposed Win-Or-Learn-Fast (WoLF) method in [116] and provided the method to adapt different learning rates when different outcomes are observed. The principle behind this method is that the authors stated that the learning agent should learn faster when it is losing and more

slowly when winning. This principle of learning faster when unsuccessful or "cautiously" when successful is also relevant in dynamic vehicular environments, e.g., when a change in network topology or traffic distribution requires the RSUs to readjust their learned policies. Besides, this feature of WoLF also encourages exploration in the early stage of learning and is important in terms of avoiding rapid convergence towards a local optimum at the beginning of the learning process.

Therefore, a straightforward adaption of WoLF is to split the value of the learning rate α in Equation (6.3) into two cases, α_{win} and α_{lose}: the Q-value is updated with α_{win} if $r = 1$ and α_{lose} if $r = -1$. Therefore, the Equation (6.3) is rewritten using separate terms for Q-value estimates before ($Q(a)$) and after the update ($Q^{'}(a)$) as follows:

$$Q^{'}(a) = \begin{cases} (1 - \alpha_{win})Q(a) + \alpha_{win}, & r = 1 \\ (1 - \alpha_{lose})Q(a) - \alpha_{lose}, & r = -1 \end{cases} \tag{6.5}$$

Again, $Q^{'}(a)$ is still a simplified term of $Q^{'}(a \mid s)$ that omits the context s. The learning agent RSU updates Q-values of its actions for each independent context s using Equation (6.5).

As mentioned earlier, the single-context cMAB adopted in Chapter 5 is enhanced in this chapter to accommodate the WoLF method. To sum up, the two underlying parallel cMAB-based prediction algorithms in the HCPC Vehicle-Centric system are summarised in **Algorithm 4**. They are referred to as dual-context cMAB and single-context cMAB, respectively.

Algorithm 4: cMAB-based Next RSU Selection Algorithm

Initialisation (if not done): For RSU $m \in \mathcal{M}$ with the number of actions (RSU neighbours) $\mathcal{A}_m$, their Q-values are initialised to $Q(a) = 0$ for $a \in \mathcal{A}_m$;
while *not the end of the test* **do**
  if *A vehicle connects to RSU* m **then**
    Context detection:
    Dual-context cMAB:
    1. Detect context $s_1 \leftarrow$ previous RSU before m;
    2. Detect context $s_2 \leftarrow$ Vehicle ID ;
    3. Dual context $s_D \leftarrow s_1 + s_2$;
    Single-context cMAB:
    Single context $s_S \leftarrow$ previous RSU before m
    if s_* *(*s_D *or* s_S*) is a* ***new*** *detection* **then**
      Create an **entry** of s_* to its action values;
      Initialise $Q(a \mid s_*) = 0$, $\forall a \in \mathcal{A}_m$;
    end
    Predict the next RSU a_* (a_D **or** a_S) by:
    $(a_D \mid s_D) \leftarrow$ action taken based on Eq. (6.1);
    $(a_S \mid s_S) \leftarrow$ action taken based on Eq. (6.1);
  end
  if *Handover happens* **then**
    Reward r_* **(**r_D **or** r_S**) generation:**
    $r_D \leftarrow$ observe the reward of a_D according to Eq. (6.4);
    $r_S \leftarrow$ observe the reward of a_S according to Eq. (6.4);
    Update ***Q*-tables** of RSU m with r_D and r_S for ***Dual-context cMAB*** and ***Single-context cMAB*** by Eq. (6.5):
    if r_* *is 1* **then**
      $Q(a_* \mid s_*) \leftarrow (1 - \alpha_{win})Q(a_* \mid s_*) + \alpha_{win}$
    end
    if r_* *is -1* **then**
      $Q(a_* \mid s_*) \leftarrow (1 - \alpha_{lose})Q(a_* \mid s_*) - \alpha_{lose}$
    end
  end
end

6.3 Simulation and Test Scenarios

This section covers the introduction of simulation in the work. Compared to the work in Chapter 4 & 5, the significant changes are in traffic simulation with multiple scenarios. The network model and simulation are still identical to that in Section 5.5.

6.3.1 Test Scenarios

Three vehicular test scenarios are designed to simulate realistic traffic scenarios and the corresponding test data is generated by Simulation of Urban MObility (SUMO) [73]. They are summarised as the following:

- **Scenario I - Commuting traffic**:

  This scenario aims to simulate daily commuters in reality. Normally, such commuting vehicles depart and arrive from one area in a city to another (e.g., residential area to workplace). Two urban areas are interested, Las Vegas as the primary city and Manchester as the secondary city to generalise the application of the proposed HCPC Vehicle-Centric system on two cities with two very different road layouts. Five traffic zones (TAZs) are defined in SUMO to simulate realistic residential and workplace areas (assuming that a TAZ contains both areas) and each two of them form a TAZ pair, which results in 20 TAZ pairs. 10 vehicles commute between a TAZ pair, resulting in 200 vehicles in total. The distribution of the TAZs and RSUs in two cities are the same as Figure 4.4a and Figure 4.4b in Section 4.4.1.

  Another feature of commuting traffic is that commuters generally follow a point-to-point daily routine. Thus, to approximate this pattern, a specific vehicle travelling between two TAZs departs from a specific road in the originating TAZ as its home address and arrives at a specific road in the terminating TAZ as its workplace address, which is referred to as a "*departure trip*" and, conversely, as a "*return trip*". A "*departure test trace*" and a "*return test trace*" consist of 200 departure trips (i.e., vehicles) and 200 return trips, respectively. Furthermore, an individual vehicle is associated with an ID (ranging from 0 to 199 in this case) and its ID remains unchanged throughout all the test traces which reinforces the fact that they are commuters. Figure 6.2a and Figure 6.2b show an example of routes of all commuting vehicles in the two cities.

- **Scenario II - Random traffic**:

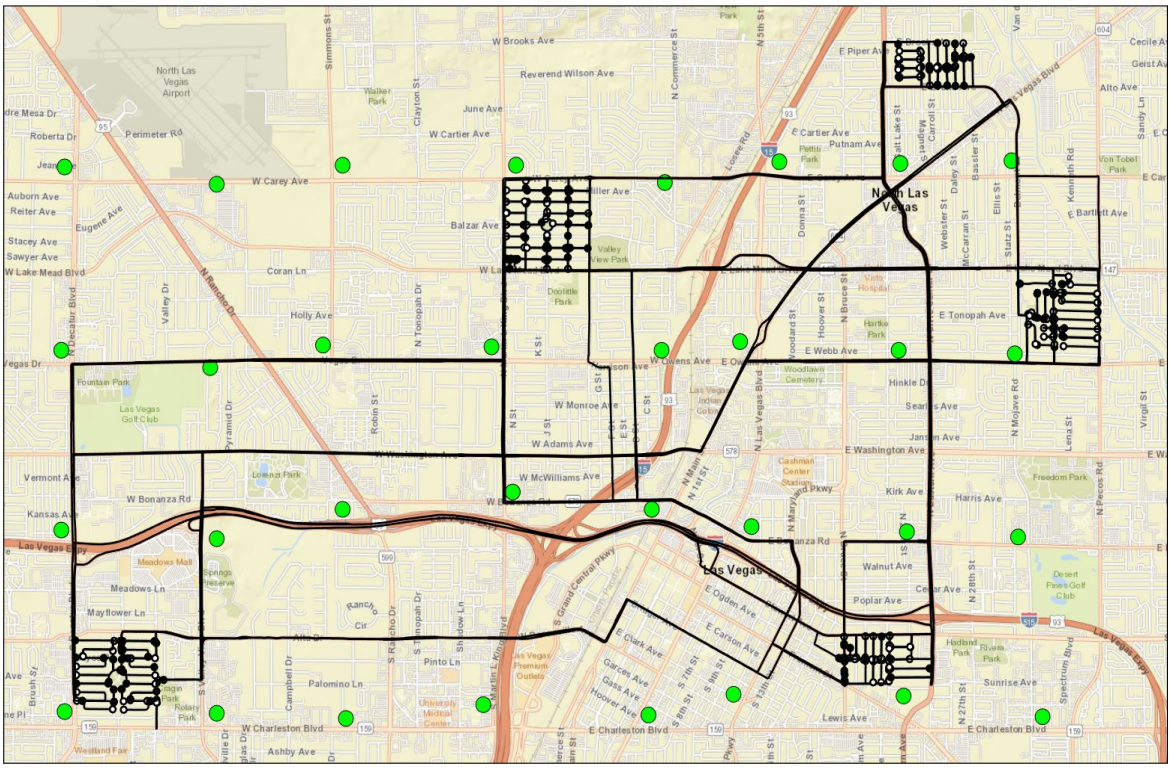
  This scenario is an extremely random scenario where vehicles randomly depart and arrive at locations on the map, independent of TAZs, but still follow the shortest path. Additionally, vehicle IDs in one test trace are different from those in another test trace (i.e., no duplicated IDs exist). This scenario may not be totally realistic but is meaningful to assess the performance of the proposed proactive caching system under such extreme circumstances. For consistency, there are also 200 random trips in each test trace of this scenario. Fig. 6.3a and Fig. 6.3b show an example of this scenario in the two cities. Trace generation in SUMO of such random scenario can be referred to as "Trip generation" in Appendix A.

- **Scenario III - Mixed traffic**:

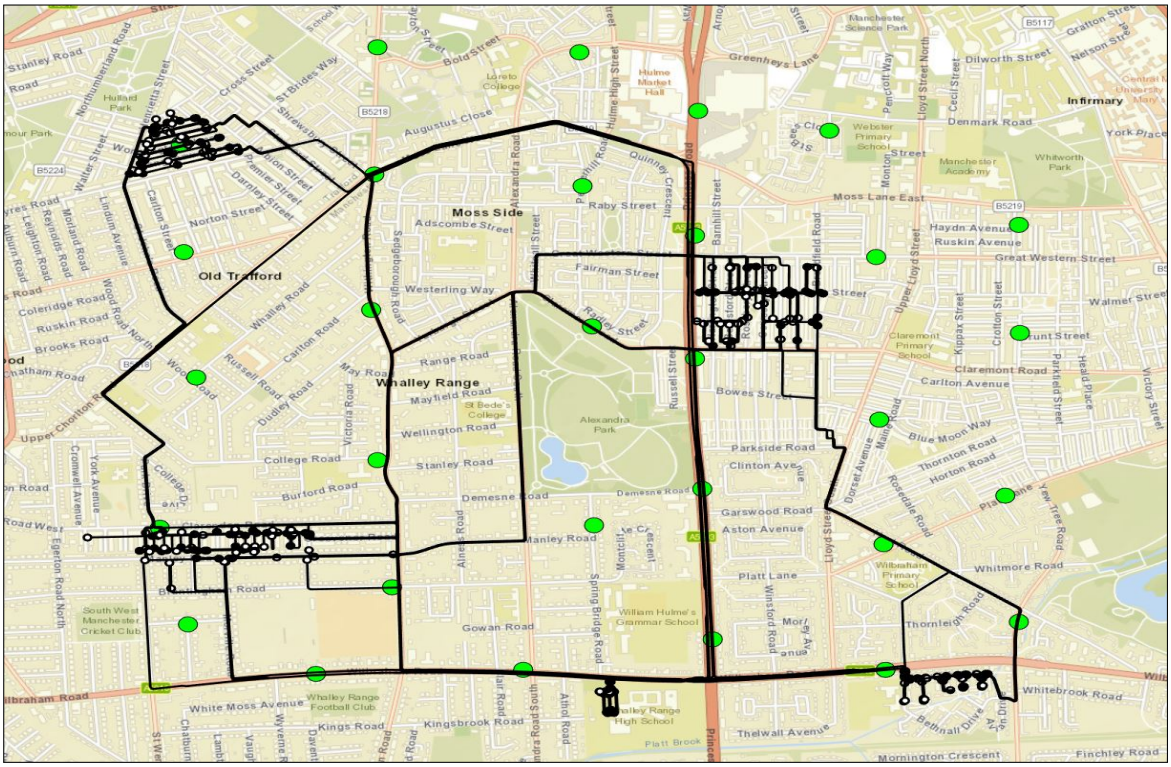
  In reality, it is very likely that the daily traffic in an urban area is mixed. In other words, it is composed of both commuting traffic and random traffic. The former is the commuters and the latter is generally new and random traffic going through the area. Therefore, the purpose of Scenario III is to simulate this more realistic scenario and is a mixture of Scenario I and II. For simplicity, traffic is mixed with an equal percentage of 50%, which results in two groups of vehicles: 200 commuting vehicles and 200 random vehicles, in each test trace of Scenario III. In addition to the mentioned traffic features in Scenario I and II, this test scenario also differentiates the two vehicle groups by their IDs (i.e., random vehicles do not use IDs ranging from 0 - 199). An example of this scenario can be referred to as the combination of Figure 6.2a and 6.3a or Figure 6.2b and 6.3b.

6.3.2 Traffic Simulation

Each of the above scenarios has 200 test traces including departure test traces and return test traces. These 200 test traces are organised in the order of *departure-return-*

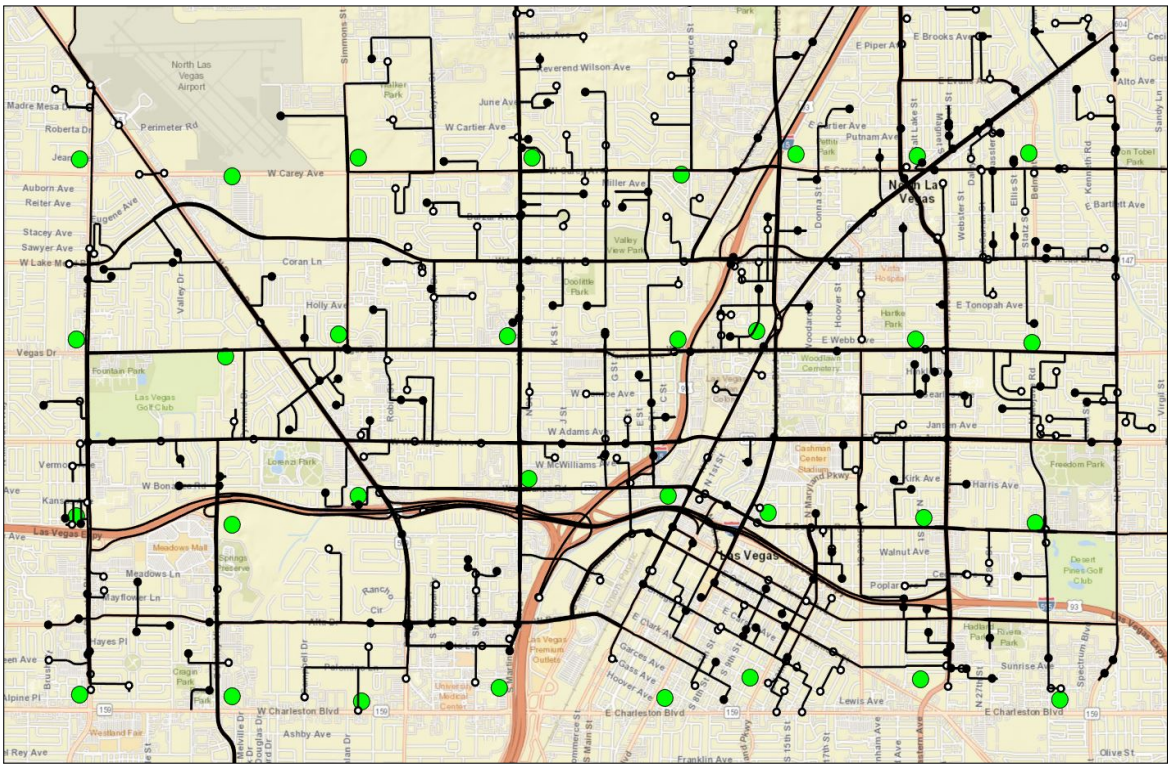


(a) An example of the commuting traffic routes in Las Vegas

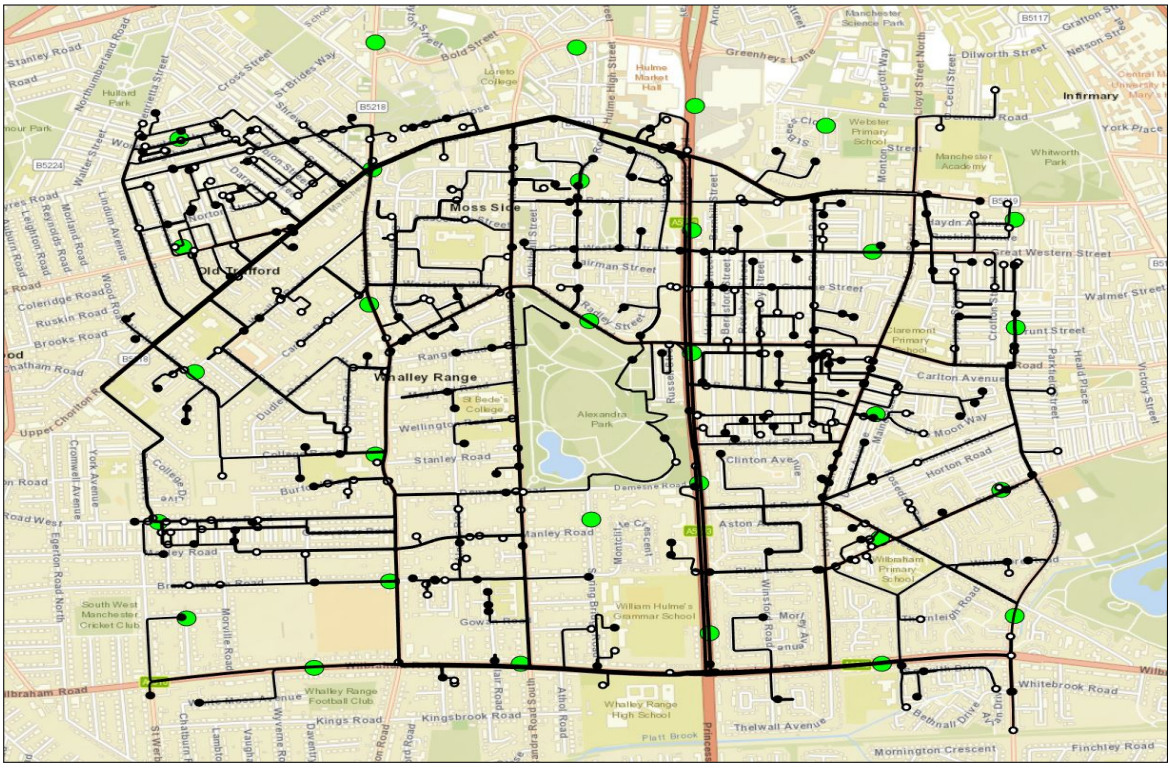


(b) An example of the commuting traffic routes in Manchester

Figure 6.2 Commuting traffic routes example in Las Vegas and Manchester. The blue points are RSUs and their IDs can be referred to Figure 4.4



(a) An example of the random traffic routes in Las Vegas



(b) An example of the random traffic routes in Manchester

Figure 6.3 Random traffic routes example in Las Vegas and Manchester. The blue points are RSUs and their IDs can be referred to Figure 4.4

...-departure-return during simulation for simulating a complete workday in an urban area, though this is not important for Scenario II which simulates completely random traffic.

On the other hand, 200 test traces also aim to simulate 200 workdays and the simulation period in SUMO is between 8 am to 9 am for departure trips and 5 pm to 6 pm for return trips. The vehicles' routes are defined by the tool *duarouter* and follow the Shortest or Optimal Path Routing rule. They depart at the *maxSpeed* and follow the default Car The Following Model is used to set the maximum the safe speed in the sense of being able to stop in time to avoid a collision. Other road behaviours apply as well such as lane changing, acceleration/deceleration, intersections, etc. Technical details about these settings can be found in SUMO documentation[1] or Appendix A. These test traces are fed into the event-driven network simulation module (described in Section 3.2.2) that have been applied to Chapters 4 & 5. Table 6.1 summarises the relevant parameters applied in both traffic simulation and network simulation.

Table 6.1 Simulation Parameters

Parameter	Description	Value
α_{win}	WoLF learning rate when winning	0.05
α_{lose}	WoLF learning rate when losing	0.5
ϵ	ϵ-greedy exploration-exploitation	0.05
N	No. of test traces	200
V_C	No. of Commuting Vehicles	200
V_R	No. of Random Vehicles	200
T_S	SUMO Simulation Time	1 hour
$\mathcal{M}$	No. of RSUs	32 (Las Vegas) 30 (Manchester)
ω	Backhaul Link Rate	$5Gbps$
e	Transmission rate	$50Mbps$
K	Size of content database	30
F_c	Fragment size	$100MB$

[1] https://sumo.dlr.de/docs/

6.4 Performance Evaluation

6.4.1 Comparing Systems

Five proactive caching systems are studied to evaluate their prediction performance:

- *HCPC Vehicle-Centric System*: The vehicle-centric variant of the hybrid cMAB system. It implements the switching mechanism at the vehicle level. The window size WS chosen for extracting the historical prediction data is 20 in order to obtain sufficient past prediction samples.

- *HCPC RSU-Centric System*: The RSU-centric variant of the hybrid cMAB system. Different from the HCPC Vehicle-Centric system, it focuses on the switching mechanism at the RSU level. The window size WS chosen for extracting the historical prediction data is 3 because it is sufficient to obtain a similar sample size with $WS = 20$ in the Vehicle-Centric system.

- *Previous-RSU cMAB-based Proactive Caching System*: This is the system that only uses the previous RSU as the context in cMAB. Its superiority has been tested and verified in the work in Chapter 5 and [114]. In this chapter, the WoLF variable learning rate is further implemented in order to maintain consistency with the HCPC system.

- *CPT+ based Proactive Caching System*: This system is based on the sequence prediction algorithm Compact Prediction Tree+ (CPT+). Different from the work in [8] and Chapter 4, the algorithm is adjusted to be used in an online mode. Briefly, an RSU trains its prediction tree model with all the available vehicles' data and when predicting the next RSU for a vehicle, it matches all the past RSUs this vehicle has connected and gives out the most possible RSU (highest score). To some extent, CPT+ also makes use of "context".

- *PPM-based Proactive Caching System*: This system implements the first-order Prediction by Partial Matching (PPM). It is a broadly used technique for context modelling and prediction as in [51]. Again, it has been adjusted to exploit online learning.

Remark: For clarity, the above five systems are referred to and denoted in the following figures as: ***HCPC Vehicle-Centric***, ***HCPC RSU-Centric***, ***PrevRSU-cMAB***, ***CPT+*** and ***PPM***, respectively.

6.4.2 Evaluation Metrics

The performance of the proactive caching system is assessed with *cache hit ratio*. For these systems, the cache hit ratio completely depends on how accurately a learning RSU can predict or select the correct next RSU. In other words, a selected action is considered correct if and only if it matches the actual RSU that a vehicle transits to. Therefore, the following metrics is defined for system evaluation:

- *Cumulative Prediction Accuracy with Sliding Window*: Denoting the total number of predictions as $Q_i^{prediction}$ and correct ones as $Q_i^{correct}$ of particular test trace $i \in N$. A fixed sliding window sw is applied to the cumulative accuracy. Thus, prediction accuracy PA_n up till test trace $n \in N$ is defined as:

$$PA_n = \begin{cases} \frac{\sum_{i=1}^{n} Q_i^{correct}}{\sum_{i=1}^{n} Q_i^{prediction}}, & n \leq sw \\ \frac{\sum_{i=n-sw+1}^{n} Q_i^{correct}}{\sum_{i=n-sw+1}^{n} Q_i^{prediction}}, & n > sw \end{cases}$$

6.4.3 Simulation Results

In the thesis, Las Vegas is treated as the primary city for simulation. Therefore, all three scenarios have been tested with the traffic data of Las Vegas. As the purpose of using Manchester city is to show the generalisation of the proposed system to different

road layouts, only the most detailed Scenario III is included to achieve this. In the following, results on a scenario basis will be demonstrated and analysed.

A) Scenario I - Commuting traffic

Fig. 6.4 demonstrates the prediction performance of the five proactive caching systems under Commuting traffic scenario in Las Vegas. As the traffic pattern of this scenario focuses on purely commuting traffic, their routes should be predictable. The accuracy of the two HCPC systems that reaches nearly 95% after convergence further validates this. The lost 5% accuracy results from ϵ-greedy exploration algorithm where 0.05 is adopted. The significant superiority of HCPC systems benefits from the switching mechanism which guarantees the best accurate action to be taken. It is obvious that the prediction accuracy of both HCPC systems does not show a clear difference and again, this is due to 1) the nature of the commuting traffic pattern in this scenario and 2) the introduction of vehicle ID in the dual-context cMAB algorithm. After a certain period of learning (approximately 20 test traces as depicted in Fig. 6.4), overall the RSUs in both HCPC Vehicle-Centric and HCPC RSU-Centric tend to finalise their decisions with the prediction of dual-context cMAB.

They outperform the PrevRSU-cMAB system by 20% and nearly 30% over the CPT+ system despite the fact that it is experiencing a slow-growing trend as the CPT+ model gets increasingly mature with more data being used to establish its model. With this trend, it could be inferred that CPT+ may reach a similar level of performance as HCPC systems perhaps after 1000 more test traces. Nevertheless, this is also its limitation in terms of adaptability and flexibility. The first-order PPM system performs the worst because essentially it is the same as the baseline *Probability-based Proactive Caching System* investigated in Chapter 5 and therefore cannot break the intrinsic limit of a certain scenario.

B) Scenario II - Random traffic

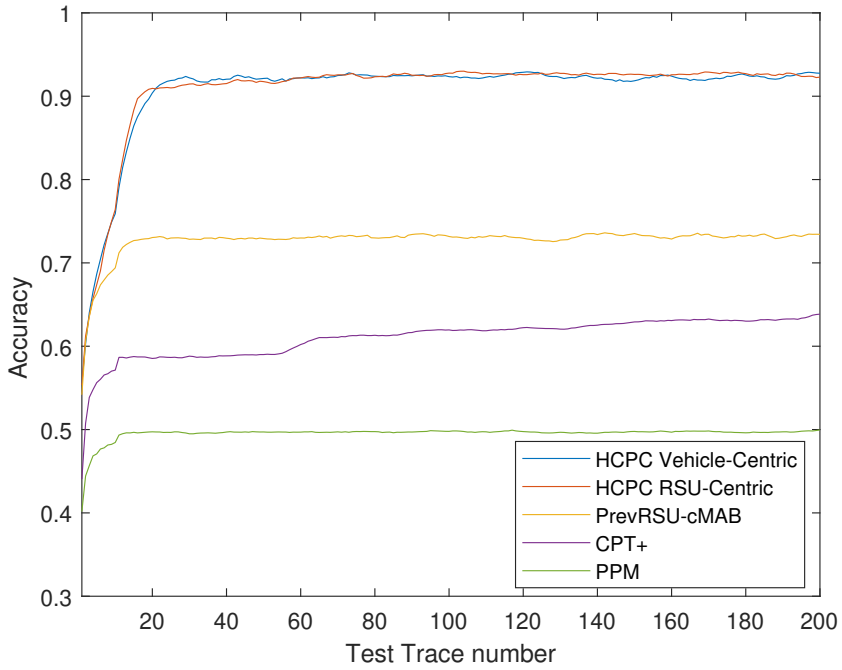


Figure 6.4 Overall Prediction Accuracy in Las Vegas - Commuting Traffic Scenario

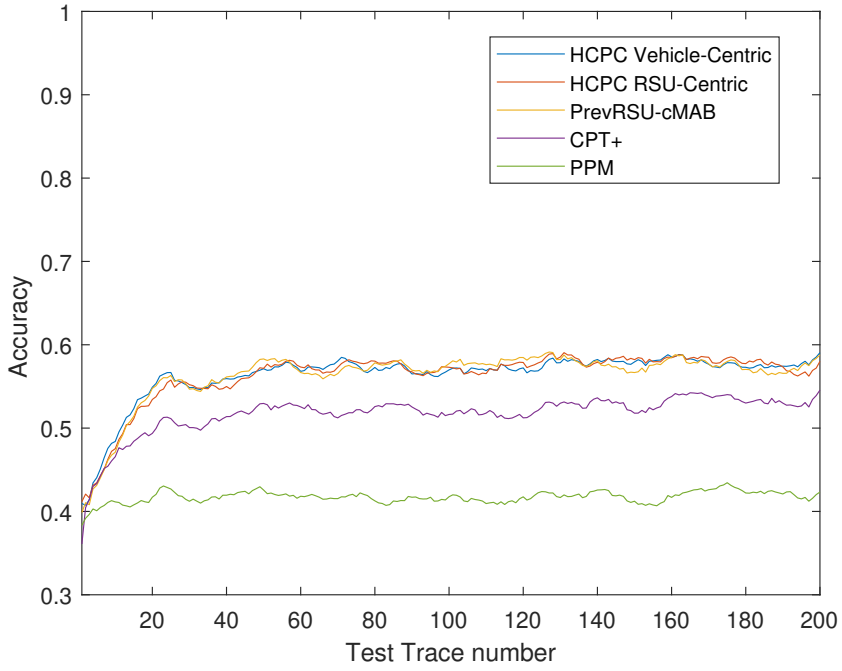


Figure 6.5 Overall Prediction Accuracy in Las Vegas - Random Traffic Scenario

The performance of the systems under extreme Random traffic in Las Vegas depicted in Fig. 6.5 shows obvious degradation especially for cMAB-based systems. Recall the traffic pattern in this is extremely random including both randomnesses in routes and vehicle IDs. Due to this nature, the HCPC systems always finalises their predictions with single-context cMAB because the accuracy of dual-context cMAB is constantly outperformed by single-context cMAB. This makes both systems identical to the PrevRSU-cMAB system that uses the previous RSU only as context. Despite this, they still outperform CPT+ and PPM-based ones. Such randomness in this scenario is also reflected in the oscillations of the result curves, unlike a much more smooth curve as in the purely commuting scenario.

C) Scenario III - Mixed traffic

Prediction performance of the proactive caching systems in Las Vegas and Manchester under the mixed scenario is shown in Fig. 6.6 and Fig. 6.7 respectively. HCPC Vehicle-Centric system outperforms the other four systems and shows the similar performance of nearly 80% accuracy in both cities. Therefore, the proposed HCPC Vehicle-Centric system can be generalised and applicable in various urban areas.

Compared to Commuting traffic and Random traffic in Scenario I and II, its accuracy falls in between. One reason for this is because of the co-existence of both commuting traffic and random traffic. On the other hand, it is in this relatively more realistic scenario that the proposed HCPC Vehicle-Centric system shows its superiority over its counterpart HCPC RSU-Centric system that has 70% of overall prediction accuracy. Thanks to its vehicle-centric feature, the most possible prediction is always made for an individual vehicular user (most likely a commuter vehicle) independent from other users. However, an RSU in the HCPC RSU-Centric system may make a less accurate prediction for a vehicle due to its RSU-centric feature. For instance, a vehicular user may benefit if the RSU finalises its prediction for this user with dual-context cMAB but for historical reasons, the

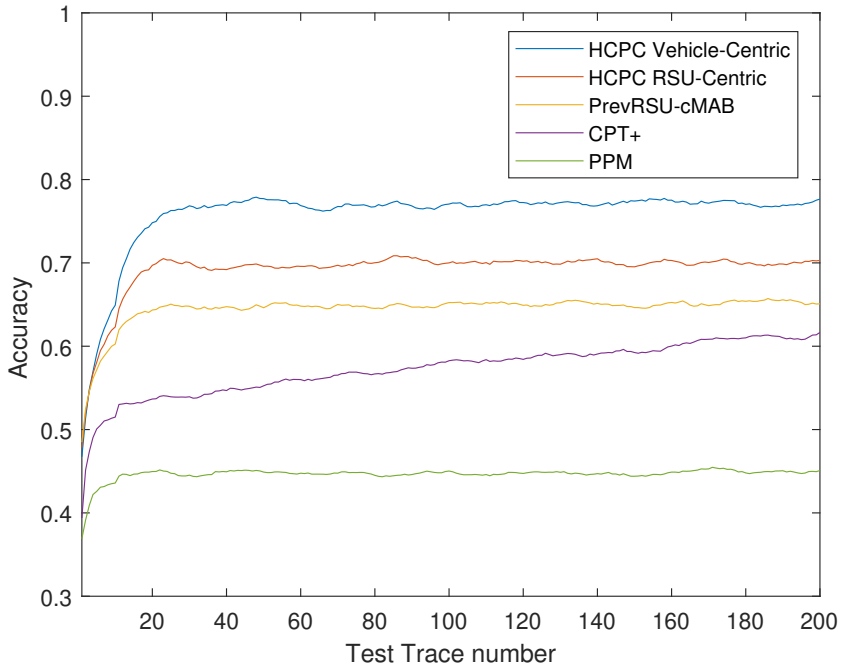


Figure 6.6 Overall Prediction Accuracy in Las Vegas - Mixed Traffic Scenario

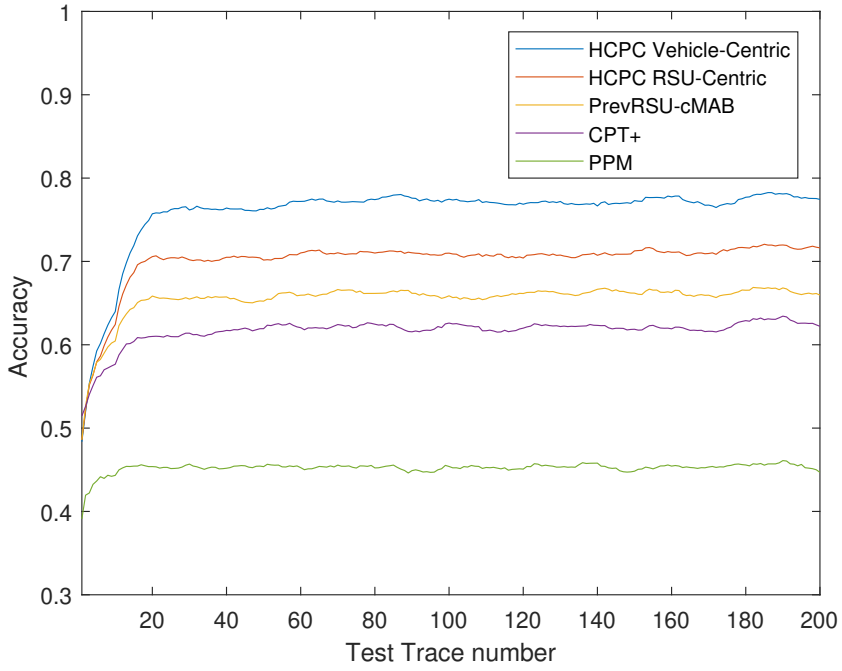


Figure 6.7 Overall Prediction Accuracy in Manchester - Mixed Traffic Scenario

RSU still believes the prediction of single-context cMAB can benefit most of the users connecting to it. This is when inaccurate predictions are made. In contrast, the HCPC Vehicle-Centric system avoids such situations by guaranteeing that the finalised prediction is vehicle-specific. To further validate this argument, Fig. 6.8 demonstrates the prediction accuracy of all the commuting vehicles in the two HCPC systems in Las Vegas and Manchester. For Las Vegas, the cumulative accuracy of these vehicles in the HCPC Vehicle-Centric system is the same as in the purely commuting scenario and is not affected by the random traffic, but they experience degradation in the HCPC RSU-Centric system. Although not shown, this is also a valid argument in the purely commuting traffic in Manchester.

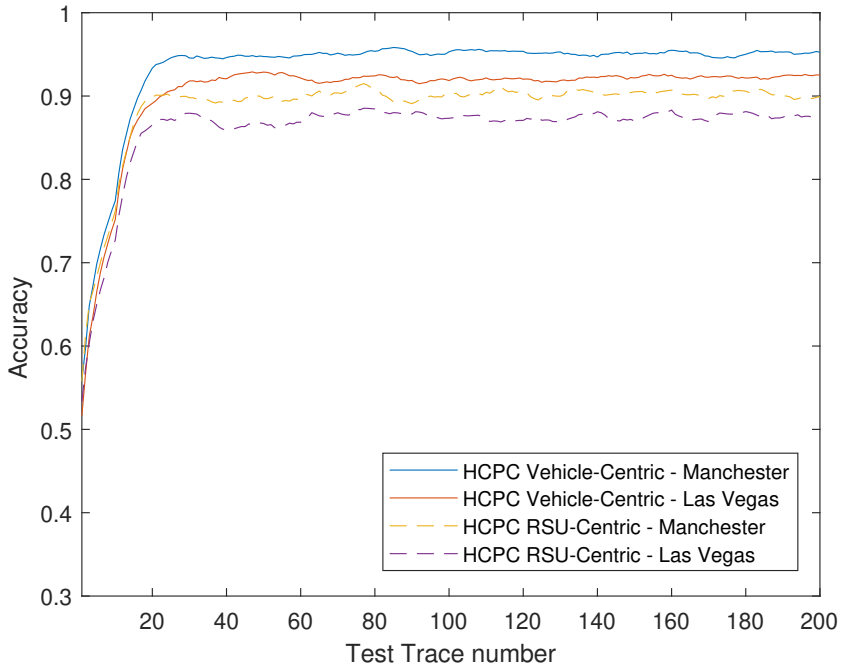


Figure 6.8 Prediction Accuracy of Commuting Vehicles Only in Two Cities - Mixed Traffic Scenario

6.5 Extended Study on An Alternative Commuting Traffic Scenario

This section aims to provide insights into situations that may impact the accuracy of dual-context cMAB, through analysis of individual vehicles and RSUs in a special commuting traffic scenario which is an intermediate between Scenario I and Scenario II in Las Vegas in Section 6.3.1. In fact, it is identical to that in Chapter 5, Section 5.5.1, except that Section 5.5.1 does not consider return trips of vehicles. In addition to showing the general prediction performance of the proactive caching systems, there will be a comprehensive comparison to the point-to-point commuting traffic scenario in Section 6.3.1. By analysing the unfavourable factors that limit the performance of dual-context cMAB, it also aims to conclude the common limitations of MAB-based algorithms.

The following is a detailed description of this scenario:

- **Scenario IV - Commuting traffic with random Origin-Destination (OD)** This is a special variant of Scenario I in Section 6.3.1. The only difference is that commuters in this scenario do not follow a fixed point-to-point daily routine. Instead, they may depart and arrive at random locations within the departing and arriving TAZs. Therefore, it is still called a commuting scenario and may exist in reality where people do not own fixed parking places and park anywhere nearby. Fig. 6.9 shows a concrete example of this scenario.

As depicted in Figure 6.10, while both HCPC systems still outperform other proactive caching systems, they experience a degradation in accuracy compared to Scenario I - Commuting traffic. This is mainly because of the randomness in origins and destinations within TAZs. To provide more insight into this, RSU 10 is selected for further analysis. Only its performance in the Vehicle-Centric system is analysed here. As shown in Figure 4.4a RSU 10 has four actions, {6, 9, 11, 15}, and it is very close to TAZ 2. However, its overall prediction accuracy in Scenario I - Commuting traffic

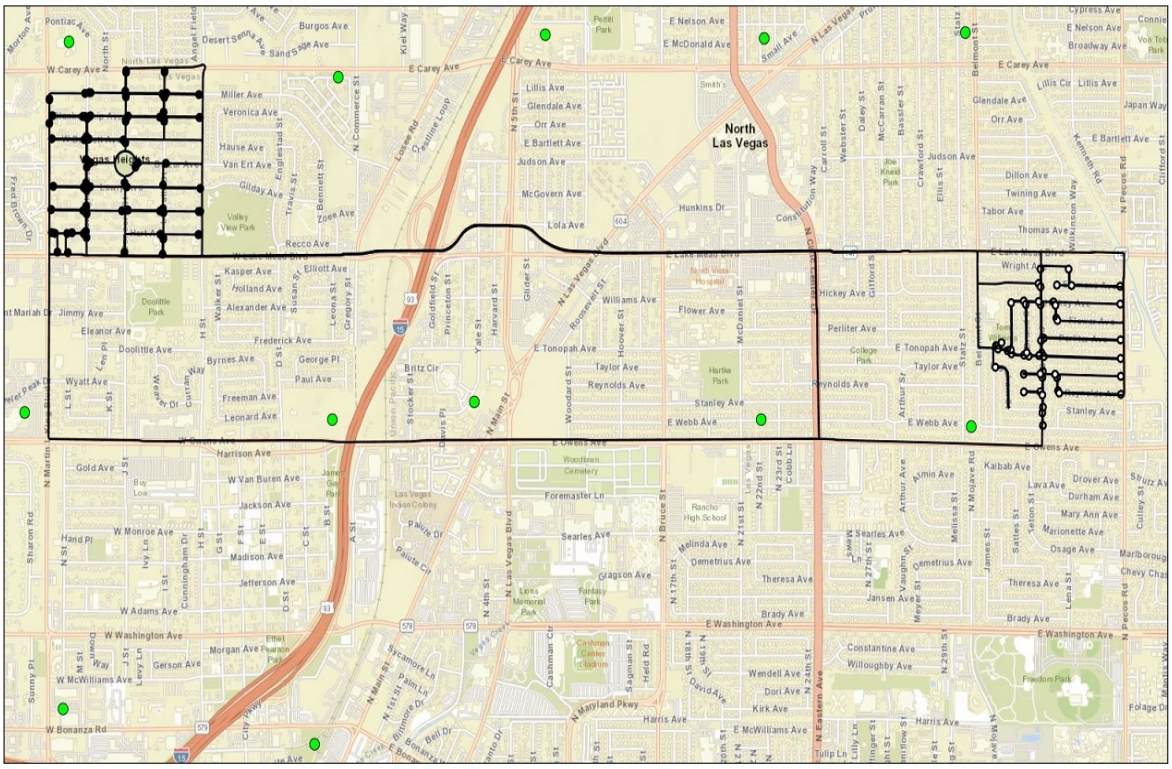


Figure 6.9 An illustration of a vehicle's departure routes in Scenario IV - Commuting traffic with random Origin-Destination (OD)

The figure shows all the 100 departure routes (overlapped) of a vehicle, where all hollow circles indicate the starting points and all solid circles indicate the ending points. All the starting or ending points are located in their own TAZ, which means that the vehicle follows its daily routine from one TAZ to another but varies in location.

and Scenario IV - Commuting traffic with random OD shows a disparity in Figure 6.11a. RSU 10 only predicts around 75% accurately in Scenario IV in contrast to 95% accuracy in Scenario I.

Figure 6.11b further disaggregates its overall performance into the separate performance of the two underlying cMAB algorithms. It is obvious that in both scenarios, dual-context cMAB dominates the performance at some point during the simulation, though this happens much later in Scenario IV than in Scenario I. Despite the notable oscillations of single-context cMAB in random OD scenario, the performance difference of single-context does not seem to be significant (both around 62%). Given the final overall accuracy, the gain brought by dual-context cMAB is considerable.

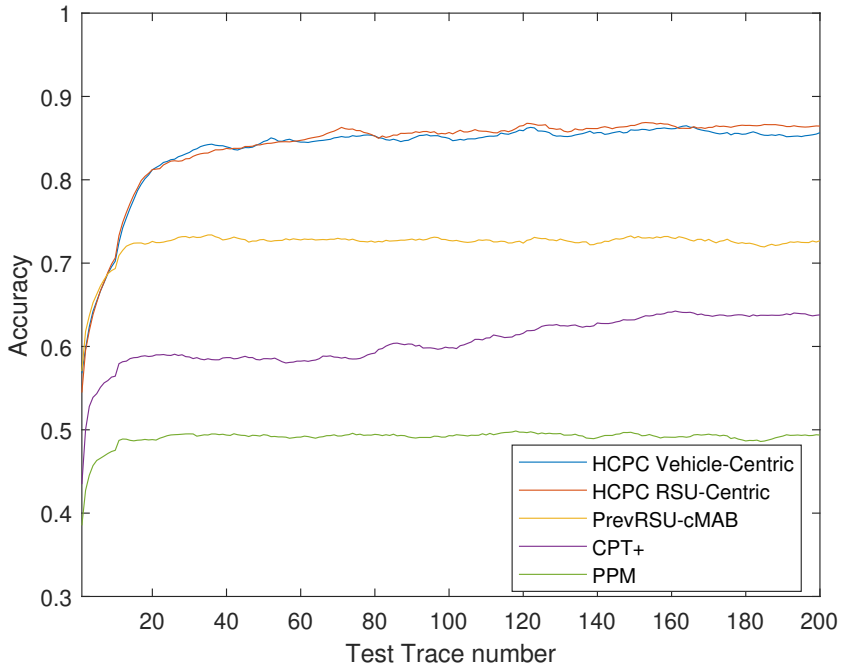
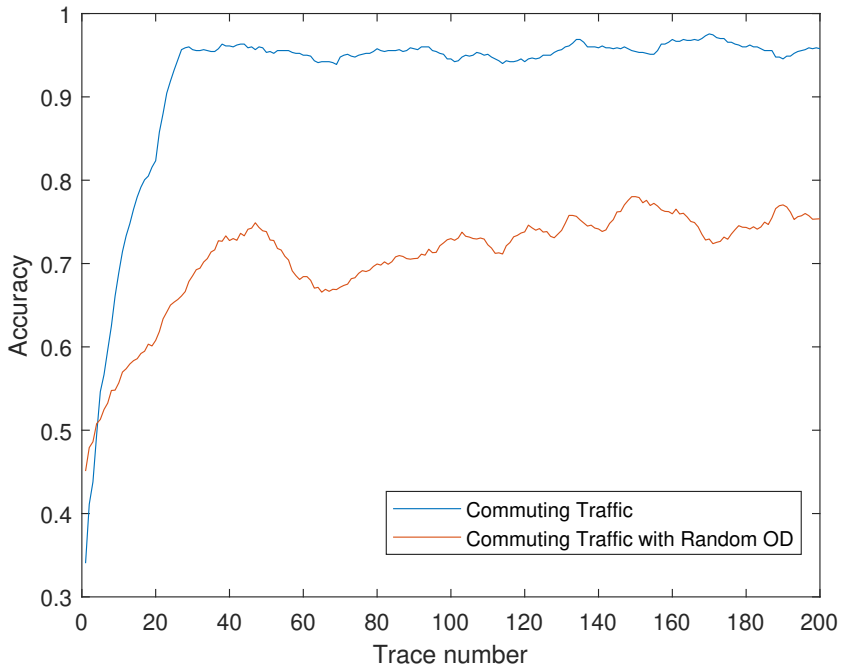


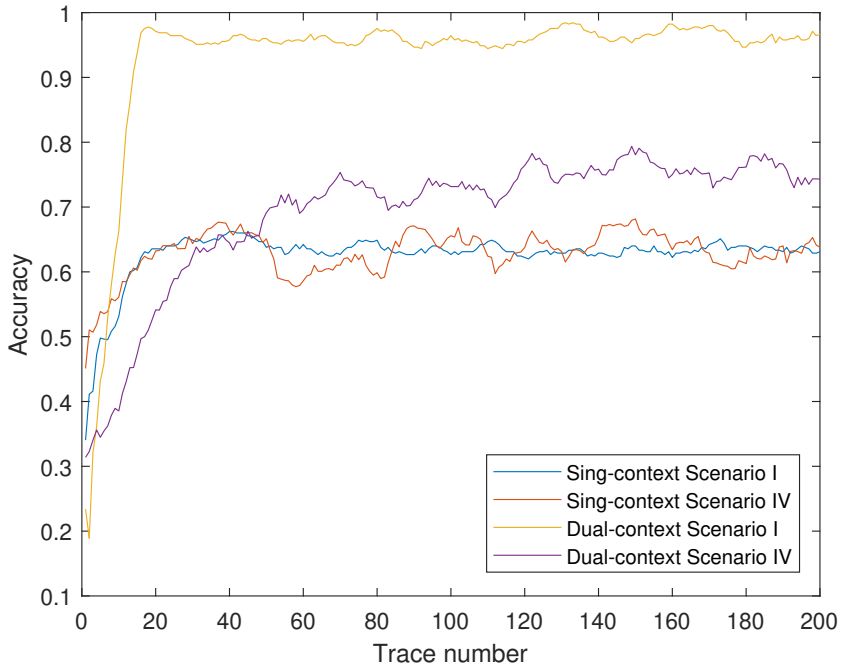
Figure 6.10 Overall prediction accuracy comparison in Scenario IV - Commuting traffic with random Origin-Destination (OD)

However, for some vehicles that RSU 10 predicts for in Scenario IV, dual-context cMAB does not work accurately and is even outperformed by single-context cMAB. Therefore, the problem now becomes what causes such a remarkable degradation of dual-context cMAB in the two scenarios. Take vehicle 90 as an example and consider the last 30 test traces, i.e., from trace 171 to 200. The followings are some observations based on the analysis of the data of vehicle 90:

- Prediction accuracy of the last 30 traces is 50%
- Dual-context combinations, (Vehicle ID, Previous RSU), used by RSU 10 to make a prediction for vehicle 90 are (90, 6), (90, 9), and (90, 15)
- Basically, all the wrong predictions happened in vehicle 90's departure trips, from TAZ 3 to TAZ 2 (referred to Figure 4.4a), under context (90, 15)
- The prediction accuracy under context (90, 15) is only 6.67%

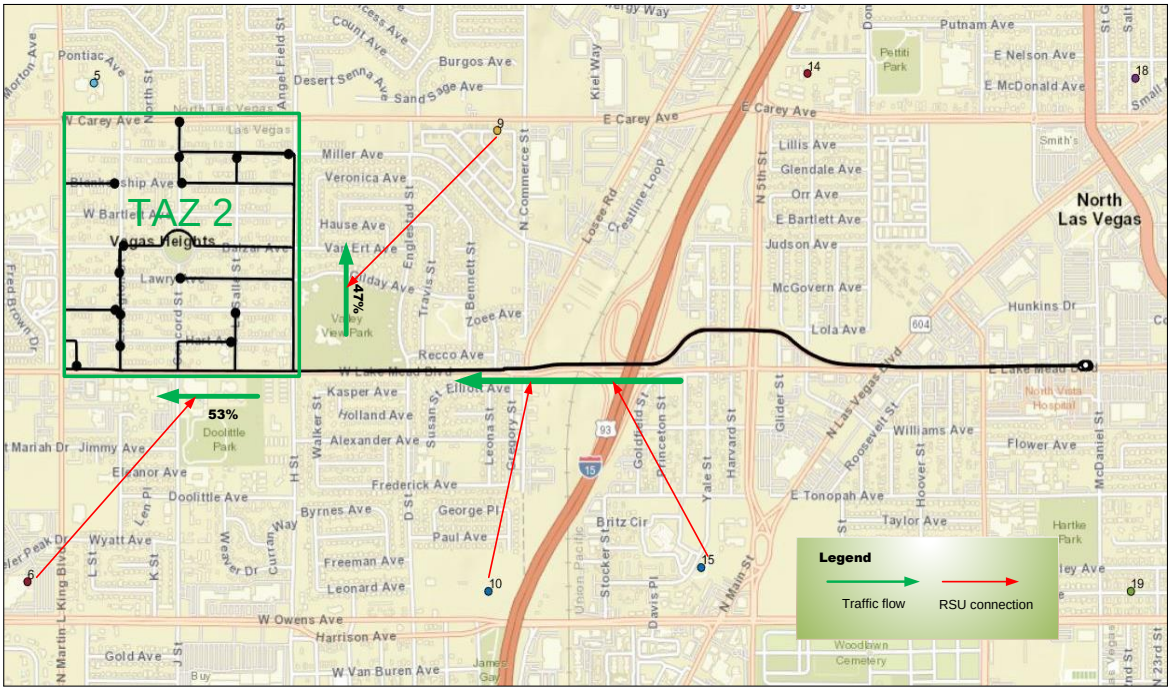


(a) Overall performance comparison

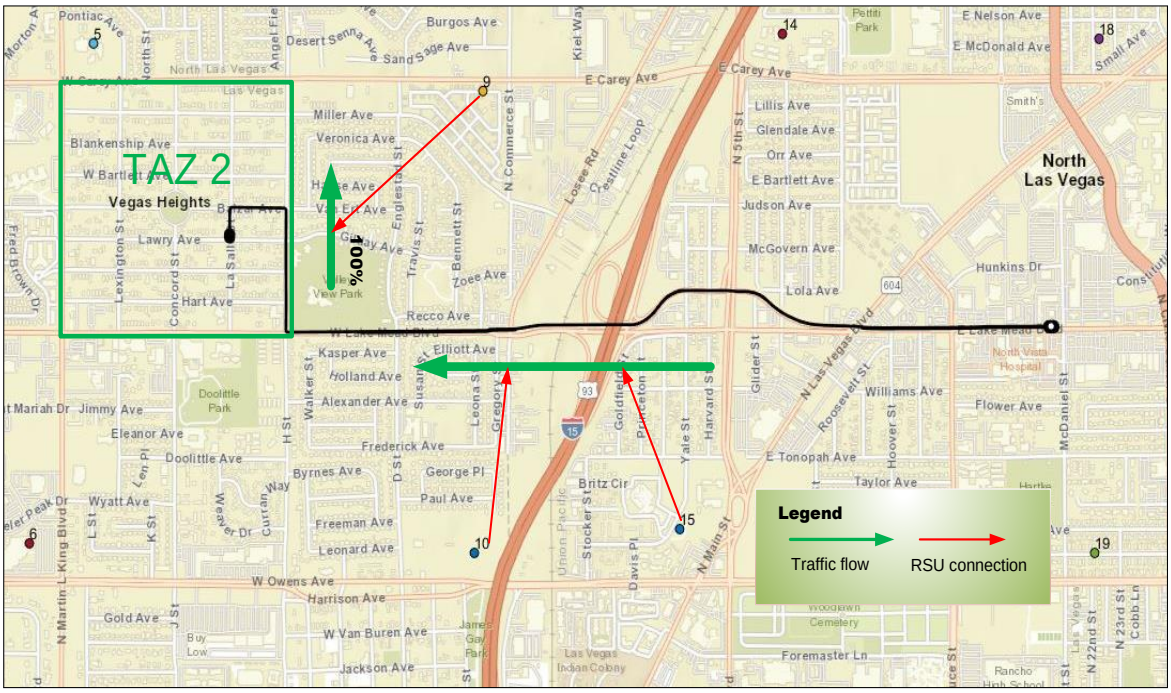


(b) Performance of the two underlying algorithms in two scenarios

Figure 6.11 Prediction performance comparison of RSU 10 in two commuting traffic scenarios in Las Vegas: Scenario I - Commuting Traffic vs Scenario IV - Commuting traffic with random OD.



(a) Scenario IV - Commuting traffic with Random OD



(b) Scenario I - Commuting Traffic (point to point)

Figure 6.12 Partial Departure routes of vehicle 90 in Las Vegas

As mentioned earlier, the main contributor to this inaccuracy is the randomness in the arrival TAZ, TAZ 2 in this case. Figure 6.12 illustrates some partial departure routes of vehicle 90 before it arrives TAZ 2. As shown in Figure 6.12a, vehicle 90 connects to RSU 6 or RSU 9 after RSU 10 because its destination is somewhere in TAZ 2. The proportions of such transitions to RSU 6 and RSU 9 in the last 30 test traces are 53% vs 47%, respectively. Consequently, the Q-values of context (90, 15) of RSU 10 end up converging to $\langle -0.9980, -0.9965, -0.9980, -0.9980 \rangle$. This means that RSU 10 believes that no convincing action exists and it is very easy to make inaccurate predictions with Q-values like these. In contrast, such a situation is rare in Scenario I as shown in Figure 6.12b, because it simulates point-to-point traffic and such randomness in TAZs is minimised. As a result, Q-values of $\langle -0.5000, 0.9927, -0.5000, -0.500 \rangle$ of context (90, 15) is achieved at the end of the simulation, which means that the second action i.e., RSU 9 is a convincing action to take to achieve accurate prediction.

To sum up, the above situation where Q-values are all negative or even close to -1 may happen in any MAB-based algorithms including dual-context cMAB, single-context cMAB as well as non-contextual MAB as studied in Chapter 5. Every dimension of context introduced is to help reduce the uncertainty of the agent RSU about its actions. Therefore, to resolve the above dilemma, the agent RSU may need further information on top of dual context, e.g., the lane in which the vehicle is currently positioned. This will be discussed in future works.

6.6 Conclusion

This chapter addresses the problem of proactive caching at the next RSU with a *Hybrid cMAB Proactive Caching System* that exploits two parallel underlying cMAB-based prediction algorithms: *Dual-context cMAB* and *Single-context cMAB*. The system allows RSUs to adaptively finalise its predictions between two algorithms. The hybrid system is further developed into two variants, Vehicle-Centric System and RSU-Centric

System, and their prediction performance is evaluated by comparing with three other systems, namely Previous-RSU cMAB, CPT+, and PPM, under three realistic-like traffic scenarios in two urban areas of Las Vegas, USA, and Manchester, UK. Simulation results have shown the excellent performance of the proposed hybrid proactive caching system. It has reached approximately 93% prediction accuracy under the Commuting traffic scenario and the Hybrid Vehicle-Centric System, in particular, still reaches nearly 80% accuracy in the Mixed traffic scenario while keeping the excellent prediction performance for commuting vehicles the same as in the Commuting traffic scenario. The results of the two cities demonstrate its superiority over the other three proactive caching systems, as well as its adaptability and applicability to different test scenarios and road layouts. In addition, this chapter also provides extended discussion and investigation on the performance in a special commuting traffic scenario, aiming to gain insights into what situations may limit cMAB-based algorithms.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

This thesis has focused on addressing the proactive edge caching problem in vehicular networks by mobility prediction. With this approach, effective proactive edge caching can be achieved by pre-storing relevant content in the most likely network location such that vehicles can continue unfinished transmission immediately when they reach the predicted location without having to request it from the original server, thereby addressing the intermittent connection challenge caused by the high-speed vehicular environment. From this perspective, accurately predicting the future location of a vehicle is closely related to the performance of proactive caching. The research work in this thesis has developed mobility-prediction algorithms based on machine learning techniques including sequence prediction and multi-armed bandit (MAB) learning, and these algorithms are designed to predict the next road-side unit (RSU) that a vehicle is likely to connect to in the future. Moreover, the proposed algorithms, as well as the corresponding systems, were evaluated in various traffic scenarios of different urban areas and their performance appeared to be independent of topology and topography. Furthermore, the thesis has been dedicated to exploring different approaches to improve the prediction accuracy of the system.

Specifically, a proactive caching system based on sequence prediction, SPPC system, was proposed in Chapter 4. It is based on a sequence prediction algorithm, named Compact Prediction Tree plus (CPT+). The work has verified the feasibility of modelling RSUs as symbols in sequences and demonstrated a good example of performing sequence prediction for the purpose of the next RSU prediction. The excellent performance superiority of the proposed system over the other comparing ones was shown by numerical results of simulation: over three times and twice better than the *non-proactive caching system* and *Baseline Proactive Caching system*, respectively, in terms of cache performance. This has also motivated further exploration in the following chapters for other machine learning techniques to achieve higher prediction accuracy of the next RSU.

One key constraint on the adaptability and applicability of the CPT+ based SPPC system in Chapter 4 is its requirement for offline training and its increasingly complex tree model. This has become the motivation for developing online model-free *reinforcement learning* (RL) based proactive caching system. RL is one of the most popular and powerful approaches in both wireless networks and other artificial domains due to its self-organising and self-co-ordination abilities. Moreover, it eliminates the need for manual intervention that is potentially challenging and time-consuming to keep up with dynamic varying traffic load and network topology. Nevertheless, the classical full RL model that requires modelling environment states may not be efficient for wireless networks especially vehicular networks which are highly dynamic. Therefore, Chapter 5 proposed to use MAB learning model, which is a stateless model-free RL technique. The chapter designed two distributed MAB-based proactive caching systems: one is based on *non-contextual MAB* and the other is based on *contextual MAB* (cMAB). Simulation results demonstrated that the cMAB-based system reached 75% and 80% prediction accuracy in scenarios of Las Vegas, USA and Manchester, UK, respectively and this system also outperformed the variant of SPPC system by over 10% in Las Vegas. Such an advantage in prediction accuracy was also reflected in the higher proportion of content fragments transmitted by edge caches, where approximately 80% of the content in the cMAB-based system was served by caches directly in both cities. In addition to the performance in accuracy, the work Chapter 5 has also proved the practicability and superiority of the MAB-based systems analytically through uncertainty evaluation with an extended Subjective Logic framework.

Motivated by the excellent prediction performance of the cMAB algorithm of Chapter 5, Chapter 6 provided a further extensive study on this. It aimed to improve mobility-prediction accuracy by proposing a *Hybrid cMAB Proactive Caching System* (HCPC) which implemented a switching mechanism between *Single-context (one-dimensional) cMAB* and *Dual-context (two-dimensional) cMAB*. Different from the two-dimensional cMAB algorithm investigated in Section 5.6, which used the previous RSU and second-to-last RSU as combined context, this chapter designed the dual-context from different

fields, i.e., previous RSU and vehicle ID. The HCPC system was further developed into two variants: the *Vehicle-Centric* System that realises vehicle-level switching and the *RSU-Centric* System with RSU-level switching for comprehensive performance comparison, such that the benefit of single-context and dual-context can be fully exploited. Moreover, the chapter also compared the two hybrid systems with other systems presented in the previous chapters applying them to three simulated realistic scenarios in Las Vegas and Manchester, so that the adaptability of the proposed systems in different road layouts can be fully investigated. Performance results demonstrated that the hybrid Vehicle-Centric system can reach nearly 95% cumulative prediction accuracy in the *Commuting traffic* scenario and outperform the other systems considered for comparison by reaching nearly 80% accuracy in *Mixed traffic* scenario. Even in the completely Random traffic scenario, it also guaranteed a minimum accuracy of nearly 60%.

7.1.1 Original Contributions

The original contributions are listed as follows based on descending order of their importance.

Multi-armed Bandit Learning Approach for Proactive Caching

Chapter 5 proposes non-contextual MAB (MAB for short) based and contextual MAB (cMAB) based algorithms in an online learning way to address proactive caching at the next RSU. The problem is formulated as a multi-agent MAB problem by modelling RSUs as individual agents and their neighbours as actions such that each agent is able to learn action values in the MAB scheme or policies in the contextual MAB scheme. Reinforcement learning (RL) techniques e.g., Q-learning has been applied in various aspects of wireless communications such as Dynamic Spectrum Access [22]. MAB, as a special RL technique, can be more computationally efficient thanks to its single-state and model-free features, compared to the classical RL. Chapter 5 shows how MAB can

be applied to mobility-prediction based proactive caching in vehicular networks for the first time. Additionally, in contrast to the work in [19, 20] which requires a massive amount of offline training for a *Long Short-Term Memory* (LSTM) model to infer the next RSU, the online MAB learning approaches do not require offline data and can be highly adaptable to fast-changing vehicular networks. This work has been submitted to *IEEE Transactions on Mobile Computing*, with the aim of attracting more attention from the research community to use model-free MAB learning in mobility prediction.

Extended Subjective Logic Framework for Uncertainty Analysis

Another major contribution of the research work in Chapter 5 is the specifically extended subjective logic framework for proactive caching systems. The extended framework enables analytical evaluation of the overall uncertainty behind the MAB and cMAB algorithm based systems as well as two other baseline systems, using entropy. By doing this, it provides an insight into the uncertainty behind the learning-based proactive caching systems including uncertainty variation and correlation with prediction accuracy. As far as is known, no work in the literature has done this.

Hybrid cMAB Proactive Caching System

Chapter 6 proposes a novel *Hybrid cMAB Proactive Caching System* (HCPC) with a specifically designed switching mechanism to allow RSUs to adaptively finalise their predictions between the *Dual-context* (two-dimensional) and *Single-context* (one-dimensional) cMAB algorithms. Particularly, the dual-context cMAB is creatively designed with vehicle ID plus the previous RSU as combined context because the dual-context cMAB which uses the previous RSU plus the second-to-last RSU discussed in Section 5.6 has shown limited improvement in accuracy, due to limited situations that can benefit from it. Additionally, the novel hybrid mechanism designed in this chapter is dependent on the real-time online learning performance of the underlying algorithms in comparison to the hybrid scheme in [12] that depends on offline dataset quality for

determining the first-order or second-order Markov chain model. Another enhancement in the HCPC system in contrast to the work in Chapter 5 is the implementation of the *Win-or-Learn-Fast* (WoLF) variable learning rate to protect against the risk of significant changes in the vehicular environment. This work is planned to be submitted to *IEEE Transactions on Vehicular Technology* for publication.

Sequence Prediction Approach for Proactive Caching

The SPPC system proposed in Chapter 4 is an original system that utilises a sequence prediction algorithm (CPT+) to address mobility-prediction based proactive edge caching. It has shown, for the first time, the feasibility and superiority of using sequence prediction in vehicular networks to solve such problems. The original CPT+ algorithm is properly tuned to cope with repetitive training sequences and implemented through an interface to the network simulation module. The advantage of using sequence prediction for predicting the next RSU is that it can be more convenient and straightforward for realistic mobile network operators, compared to the works in [19, 20] which used deep learning for direction prediction and then inferred the next RSU. This work has been published in [8] in *IEEE 3rd Connected and Automated Vehicles Symposium*.

Distributed Independent Multi-agent Learning System

The proposed systems in Chapter 5 as well as Chapter 6 implement MAB and cMAB techniques in a distributed way on individual RSUs to realise instant learning and prediction, whilst previous similar works (e.g., [12, 19, 20]) were based on centralised approaches and offline training. The agent RSUs are capable of learning and making decisions independently and form an independent Multi-agent reinforcement learning (MARL) system. An advantage of such a system is that it significantly increases the breadth of potential applications with different information availability constraints in wireless networks because it does not require a learning agent's awareness of the

actions performed by the other agents in contrast to classic MARL such as in [117] and [118].

Extensive Test Scenarios with SUMO

This thesis has created a variety of traffic scenarios simulated by SUMO in two modern cities with significantly different characteristics and layouts, Las Vegas and Manchester, from USA and UK respectively. The aim is to test the applicability and adaptability of proactive caching systems. In particular, Chapter 6 has designed three more realistic traffic scenarios: *Commuting traffic, Random traffic, Mixed traffic* in the two cities to evaluate the system performance in a more comprehensive way. By contrast, the closest previous work in [20] only considered a highway and a single intersection scenario simulated in SUMO. The work in this thesis has provided more comprehensive traffic scenarios for evaluating the proposed algorithms and systems.

7.1.2 Hypothesis Revisited

The following hypothesis stated in Chapter 1 has guided the research work in this thesis:

"*Mobility-prediction techniques can effectively enable proactive edge caching, and the degree of improvement depends on the prediction accuracy.*"

It emphasises the effectiveness of mobility prediction techniques to proactive edge caching and the need for improving prediction accuracy. Therefore, the key contributions of this thesis listed in Subsection 7.1.1 can be summarised in the context of the above hypothesis as follows:

- The effectiveness of the mobility prediction of the next RSU for proactive caching has been demonstrated in Chapters 4 and 5. Particularly, Chapter 4 proposed sequence-prediction based algorithm (SPPC system), and the effectiveness of this approach was verified and evaluated with metrics including the served content

fragments proportion distribution, proactive caching gain, cache utilisation, and network delay by comparing to other benchmarking systems. Chapter 5 further developed two online multi-armed bandit learning algorithms, whose feasibility was demonstrated by the average percentage of content fragments satisfied by proactive caching and the average network delay. Additionally, the cMAB-based system outperformed the online variant of the SPPC system.

In addition, continuous efforts have been made to improve the accuracy of mobility prediction in Chapters 5 and 6. Chapter 5 also demonstrated the association between prediction accuracy and proactive caching performance. More specifically,

- The two MAB learning based algorithms proposed in Chapter 5 have shown excellent accuracy superiority over other benchmarking systems. Take the performance comparison in Las Vegas as an example. The prediction accuracy of cMAB-based system has outperformed CPT+ based system (a variant of SPPC system in Chapter 4) by 10%. These two systems both need context but cMAB only requires the previous RSU and leads to a much more compact prediction model than CPT+. The non-contextual MAB, on the other hand, outperformed the other two non-contextual systems by 10% and 20%. Additionally, the extended work in Section 5.6 has investigated the dual-context cMAB scheme on top of the proposed cMAB algorithm and showed limited improvement in terms of prediction accuracy. Figures 5.8 and 5.9 have illustrated the link between prediction accuracy and proactive caching performance by showing the percentage of average content fragments satisfied by proactive caching and the average network delay on a test trace basis: systems with higher accuracy transmit higher proportions of fragments by caches, hence less network delay and better proactive caching performance.

- Motivated by the results in Chapter 5, the primary focus of Chapter 6 is to develop more advanced algorithms to continuously improve the accuracy of mobility prediction. The innovations in the HCPC systems in Chapter 6 have

resulted in outstanding prediction performance. The Vehicle-Centric HCPC system, for example, has reached nearly 93% of accuracy in the point-to-point commuting traffic scenario, and 85% of accuracy with commuting traffic with random Origin-Destination compared to the 78% of the cMAB in the same traffic scenario in Chapter 5.

Last but not least, a variety of traffic scenarios in urban city areas have been designed to validate the performance and adaptability of the proposed mobility-prediction based proactive caching systems:

- Four traffic scenarios in Las Vegas and Manchester have been designed for comprehensive evaluation: *Commuting traffic with random Origin-Destination*, *Commuting Traffic*, *Random Traffic* and *Mixed Traffic*. The first one has been evaluated with all the proposed systems and the prediction accuracy has been improved progressively under such scenario.

This thesis has conducted massive empirical and in some cases analytical, evaluations of these contributions. The simulation results have shown considerable improvement in mobility prediction accuracy as well as network performance, thus, proving the hypothesis of this thesis.

7.2 Recommendations for Future Work

This section aims to give a number of recommendations for future work on the areas studied in this thesis, including the extension of ideas and potential refinements.

Introducing Three-dimensional Context

The research work presented in Chapters 5 & 6 has already demonstrated the effectiveness and improvement of contextual MAB with one-dimensional context (last RSU),

two-dimensional context (last and second-to-last RSU) as well as two-dimensional context (last RSU and vehicle ID). Nevertheless, they all have limitations in certain situations as discussed in the end of both chapters. Therefore, the third dimension of context is necessary in order to further improve the prediction accuracy of the next RSU. A potential idea is road position of the vehicle, i.e., the lane. This is particularly effective in Scenario IV in Chapter 6, where vehicles do not have fixed arrival, because the vehicle's lane position is likely to decide whether it is turning right, left, or going straight on. Such additional context is especially helpful for an RSU to make a decision when a vehicle is about to reach its destination area (a TAZ). The challenge is the accessibility of such information from the perspective of the RSU but it is worth some investigation given the increasingly intelligent vehicles.

Dynamic Hybrid cMAB System Based on Scenario Detection

Motivated by the work on different dimensions of cMAB and different combinations of dual-context cMAB, it may be meaningful to design a more sophisticated dynamic hybrid system than the one in Chapter 6. However, instead of comparing historical performance to realise the switch, it can be more innovative to realise this on a scenario basis. For example, depending on the traffic patterns of an agent RSU, single-context cMAB may be good enough when there is a dominant traffic route. Depending on its geo-location whether it is close to a TAZ, dual-context cMAB (last and second-to-last RSU) may perform better than single-context. In a more complex scenario, dual-context cMAB that considers vehicle ID or even three-dimensional context is a better option. All of these are common situations in a dynamic vehicular network and they are useful to fully understand what scheme can best benefit the network and vehicular users.

Effect of the Action-selection Strategy on Performance

ϵ-greedy action-selection strategy has been adopted in the MAB-based algorithms throughout the thesis. This strategy constantly simulates exploration with a persistent

small ϵ value. In other words, the learning agent selects a greedy action (the action with the highest Q-value) with probability $1 - \epsilon$ while exploring other non-greedy actions with the probability ϵ. This is a rather straightforward and widely used approach. However, one disadvantage of using a constant ϵ is that the learning agent keeps exploring even when it has converged to an adequately good policy, which may lead to a sub-optimal action. Therefore, it is meaningful to understand the influence of action-selection strategy on the performance of the learning algorithms. Potential candidates may include greedy exploration, random exploration, decaying ϵ-greedy exploration where ϵ is decaying as the learning process goes, and Thompson sampling that builds up a probability model (commonly Beta model) from the obtained rewards and then samples from this to choose an action. The metrics to compare these approaches can be the general prediction accuracy of the learning agent, action convergence, etc.

Effect of Learning Rate α

Similar to the action-selection strategy and reward function, the impact of the learning rate α on the the performance of learning agent is worth some investigation. Although in Chapter 6 the WoLF method has been used to realise differing learning rates for positive and negative rewards instead of a constant learning rate (0.5), they are not adequate to fully study its influence on system performance. As discussed in the thesis, a constant learning rate is important to the non-stationary environment, in particular, the vehicular environment and it is related to how many recent action values are considered for updating. An interesting direction is to customise the learning rate to individual RSU agents depending on their traffic patterns, or even individual vehicles based on their past performance.

Exploring Inverse Reinforcement Learning to Reward Function

The reward function used in the MAB-based algorithm presented in this thesis applied a constant reward value: $r = \pm 1$. However, such a empirical value may not be sufficient

to identify an optimal reward function. Inverse reinforcement learning (IRL) is a feasible technique to extract or optimise the reward function based on the observed optimal behaviour. In state-of-the-art MAB learning, the aim is to learn an optimal strategy of actions or context-action pairs to maximise the overall reward. Nonetheless, the aim of IRL is to recover the learned reward values for each action and use these values to produce a desired behaviour.

O-RAN Driven Vehicular Networks

Open Radio Access Network (Open RAN or O-RAN) has been a promising solution to virtualisation and disaggregation in future RAN architecture. By 2022, there are already over 300 members and contributors in the O-RAN Alliance since its establishment in 2018. Its specifications are expected to drive 50% of RAN-based revenues by 2028 [119]. There are two important RAN Intelligent Controllers (RIC) in the O-RAN architecture: one is the near-real-time (near-RT) RIC and the other is the non-real-time (non-RT) RIC. The two RICs aggregate and process the data from the network including network status, load, throughput, handovers, etc and leverage machine learning (ML) algorithms to determine control policies and actions on the RAN. The O-RAN architecture also makes distributed ML possible e.g., federated learning for more advanced prediction tasks. For example, the non-RT RIC can send out the hyperparameters of a deep neural network model to multiple near-RT RICs deployed in the network edge. These near-RT RICs then can collect useful data from their connected RAN nodes, train the models and send them back to the non-RT RIC. The non-RT RIC will aggregate the parameters and distribute the aggregated model to near-RT RICs. The latter will control RAN nodes through the specific *xApp* hosted in it. As highly mobile and dynamic wireless networks, vehicular networks can benefit from O-RAN, especially with mobility management service. The rich data and context information in vehicular networks are good sources to obtain sophisticated ML models in RICs, which assists RAN nodes to manage handover and allocate radio resources through *xApps*.

Appendices

A SUMO Tutorial

A.1 Network Building

The road network for SUMO is one of the essential inputs for traffic simulation. Figure 1 shows a SUMO network file opened in the SUMO user interface, which is the map of Las Vegas, USA. Such network file describes the traffic-related part of a map, the roads and intersections the simulated vehicles run along or across [73]. It is essentially a directed graph where in SUMO context, nodes are *junctions* and edges are *streets*. Every street (edge) is a collection of lanes, including the position, shape, and speed limit of every lane and every junction includes its right way of regulation and connections between lanes. Other information such as traffic light logic and roundabout descriptions is also contained in such network files.

Figure 1 A screenshot of the road network of Las Vegas shown in SUMO

A SUMO network file is usually in a **XML** format (`.net.xml`) and complies with XML schema `net_file.xsd` [1]. Although one can build a simple network by hand, a very common way to build a large SUMO network is by converting an existing map from other formats using *netconvert* tool. *OpenStreetMap* [2] is a free editable map of the whole world and is a good source for SUMO network. Once an area of a city map is extracted, a SUMO network can be generated by the following command:

```
netconvert --osm-files LasVegas.osm.xml -o LasVegas.net.xml
```

where it imports the road network stored in `LasVegas.osm.xml` and stores the SUMO-network generated from this data into `LasVegas.net.xml`.

A.2 Demand Modelling

Once the above network is generated, it is available to check with `sumo-gui` which is a graphical user interface. Apparently, only the network structure is shown and no vehicles would be driving around. This is the second important element for SUMO simulation, named *traffic demand*. Two important concepts for demand modelling are:

- **Trip**: a vehicle movement from one place to another defined by the starting edge (street), the destination edge, and the departure time.
- **Route**: an expanded trip i.e., a route of a vehicle that contains not only the first and the last edge but all edges the vehicle will pass.

In the rest of the thesis, a file that stores trip information is referred to as trip file and a file that contains route information is referred to as a route file. It is possible to define the trip file and route file manually [3] but this can be very inefficient for large networks. There are a variety of applications available for defining vehicular demands

[1] https://sumo.dlr.de/xsd/net_file.xsd
[2] http://www.openstreetmap.org/
[3] https://sumo.dlr.de/docs/

(i.e., trips) for SUMO from existing input data. In the following, two main methods that will be used in the research work in later chapters will be introduced.

Trip generation

`randomTrips.py` is a Python tool provided in SUMO package that allows the generation of a set of random trips for a given network. The output in an XML file contains the trip information defined for each individual vehicle. This is a convenient and fast way to have user-defined number of vehicular demands in seconds. It is also the method that will be used for the mobility simulation for the random trip traffic scenario in Chapter 6. An example call is given as below:

```
randomTrips.py -n <net-file> -b t0 -e t1 -p ((t1 - t0) / n)
min-distance min -o <trip-file>
```

This will result in some random trips stored in an XML file with option `-o` for the network given by option `-n`. The trips are distributed evenly in an interval defined by `-b t0` and `-e t1` in seconds and the number of trips is determined by the repetition rate given by option `-p`. For n vehicles departing between `t0` and `t1`, the option is set to `((t1 - t0) / n)`.

Another important method that is actually the primary method for commuting trip generation in the thesis is through `od2trips`. This tool converts O/D (origin/destination) matrices to trips, hence the name. These matrices defines amounts of vehicles that start from one *traffic assignment zone* (TAZ) to another within a certain time period. Therefore, the concept of TAZ should be explained first. A TAZ is described by its id (an arbitrary name) and lists of source and destination edges and TAZs can be defined by the user by drawing polygons in the road network with `netedit` and processing these polygons with `edgesInDistricts.py`. A simple example of five resulting TAZs of a network is shown below, where each TAZ may include dozens of edges.

```
<tazs>
```

```
    <taz id="1" edges="-14294334#0 -14294334#1 -14294334#2 ..."/>
    <taz id="2" edges="-14298714#10 -14298714#11 -14298714#12 ..."/>
    <taz id="3" edges="-131120994#0 -131120994#1 -131120994#2 ... "/>
    <taz id="4" edges="-14291746#0 -14291746#1 -14291746#2 ..."/>
    <taz id="5" edges="-14296364 -14297841 -14299279#10 ..."/>
</tazs>
```

Next, with a well defined TAZ file, the O/D matrices (refered to as OD file) can be generated. There are a number of supported OD-formats but in this thesis, the O-format is adopted. Based on the above TAZ file, an example of the OD file can look like the following. The file defines 10 vehicles drive between 4 pairs of TAZs, in total 40 vehicles, during the period of 8 (hour) to 9.

```
$OR;D2
*From-Time To-Time
8.00 9.00
*Factor
1.00


* some
* additional
* comments
 1      2       10
 2      3       10
 3      4       10
 4      5       10
```

Finally, the trip file can then be generated by calling `od2trips` with the following command:

```
od2trips -n <taz-file> -d <od-file> -o <trip-file>
```

where the TAZ file, OD file, and the output file name are passes to options `-n`, `-d`, and `-o`, respectively. An example of trips in a trip file is shown below:

```
<trip id="74" depart="28808.9" from="14306149#0" to="-14299279#8"
fromTaz="2" toTaz="5" departLane="free" departSpeed="max" />
<trip id="83" depart="28820.03" from="291786314#9" to="-14294334#2"
fromTaz="3" toTaz="1" departLane="free" departSpeed="max" />
<trip id="21" depart="28831.3" from="-14298382#4" to="14309880#2"
fromTaz="1" toTaz="4" departLane="free" departSpeed="max" />
```

Route generation

For route file generation, a common tool provided by SUMO package is `duarouter`. This tool generates routes based on Shortest or Optimal Path Routing rules. The trip files from the previous two methods for trip generation are suitable and can be handled by `duarouter`. For `randomTrips.py`, `duarouter` is actually embedded in it and can be called automatically by setting the relevant option `-r` with as below:

```
randomTrips.py -n <net-file> -b t0 -e t1 -p ((t1 - t0) / n)
min-distance min -o <trip-file> -r <route-file>
```

For trip files generated with `od2trips`, a direct call of `duarouter` is needed:

```
duarouter -n <net-file> -r <trip-file> -o <route_file>
```

An example of routes in a route file is shown below. Note that the actual number of route edges may be in the dozens.

```
<vehicle id="74" depart="28808.9">
    <route edges="-14318523#1 14318523#1 -14308341#3 -14308341#2 ..."/>
</vehicle>
```

```
<vehicle id="83" depart="28820.03">
    <route edges="601629490#9 -601629490#9 -601629490#8 ..."/>
</vehicle>
<vehicle id="21" depart="28831.3">
    <route edges="-179988106#6 14324945#3 14324945#4 ..."/>
</vehicle>
```

A.3 Traffic Simulation and Outputs

In the phase of the actual SUMO simulation, two important inputs are the network file built with `netconvert` and the route file generated with `duarouter`. The road network defines the area the simulation is based on and the route file defines the traffic demands i.e., the vehicles running in the road network. To run the simulation, `sumo` is called together with the passed input files as below:

```
sumo -n <net-file> -r <route-file>
--fcd-output <fcd-trace-file> --fcd-output.geo true
```

where, similarly, the option `-n` and `-r` is followed by the corresponding road network file and route file.

The option `--fcd-output` is to set the format of the output file. The documentation[4] in SUMO has listed several available types of outputs for different purposes. This thesis has chosen the *fcd output* that contains floating car data including name, position, angle and type for every vehicle, because such level of details is essential for discrete event driven simulation (DES). An example of vehicle' information of a particular second in the trace file is shown in Figure 2.

The fcd output files then needs to be converted to "trace files" in a specific format using `traceExporter.py`. Different applications that read these traces may require different formats. In this thesis, GPSDAT csv format that contains information e.g.,

[4]https://sumo.dlr.de/docs/Simulation/Output/index.html

```
<timestep time="29248.00">
    <vehicle id="101" x="-115.205998" y="36.182399" angle="358.027707"
    type="DEFAULT_VEHTYPE" speed="24.916031" pos="149.289341" lane="258995176#12_0" slope="0.000000"/>
    <vehicle id="102" x="-115.205896" y="36.169889" angle="358.434341"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="19.857912" lane="600526665#11_0" slope="0.000000"/>
    <vehicle id="122" x="-115.195527" y="36.190879" angle="87.711409"
    type="DEFAULT_VEHTYPE" speed="14.648517" pos="61.889360" lane="-14324220#7_0" slope="0.000000"/>
    <vehicle id="126" x="-115.204876" y="36.161133" angle="359.512432"
    type="DEFAULT_VEHTYPE" speed="7.136647" pos="332.815389" lane="14294812_0" slope="0.000000"/>
    <vehicle id="133" x="-115.205932" y="36.169957" angle="358.471402"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="27.358986" lane="600526665#11_1" slope="0.000000"/>
    <vehicle id="142" x="-115.193397" y="36.162903" angle="88.640248"
    type="DEFAULT_VEHTYPE" speed="9.931990" pos="187.753779" lane="-14298382#2_0" slope="0.000000"/>
    <vehicle id="144" x="-115.205849" y="36.172090" angle="2.899268"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="40.918971" lane="650732522#4_1" slope="0.000000"/>
    <vehicle id="3" x="-115.198843" y="36.161854" angle="88.994960"
    type="DEFAULT_VEHTYPE" speed="14.819693" pos="159.064519" lane="-14297929_0" slope="0.000000"/>
    <vehicle id="33" x="-115.192554" y="36.173505" angle="358.643031"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="102.878973" lane="601630089#2_0" slope="0.000000"/>
    <vehicle id="34" x="-115.202353" y="36.160539" angle="269.520476"
    type="DEFAULT_VEHTYPE" speed="11.759977" pos="46.846750" lane="14294812_0" slope="0.000000"/>
    <vehicle id="48" x="-115.159869" y="36.174887" angle="103.580275"
    type="DEFAULT_VEHTYPE" speed="21.963718" pos="51.094463" lane="145132868_0" slope="0.000000"/>
    <vehicle id="49" x="-115.192590" y="36.173504" angle="358.643031"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="102.878955" lane="601630089#2_1" slope="0.000000"/>
    <vehicle id="52" x="-115.205897" y="36.169957" angle="358.471402"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="27.358998" lane="600526665#11_0" slope="0.000000"/>
    <vehicle id="60" x="-115.205813" y="36.172088" angle="2.899268"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="40.918950" lane="650732522#4_0" slope="0.000000"/>
    <vehicle id="71" x="-115.200663" y="36.166757" angle="89.265381"
    type="DEFAULT_VEHTYPE" speed="7.996374" pos="11.231389" lane="650732528#4_0" slope="0.000000"/>
    <vehicle id="79" x="-115.186069" y="36.175575" angle="102.346410"
    type="DEFAULT_VEHTYPE" speed="30.748225" pos="245.212244" lane="124183712#2_1" slope="0.000000"/>
    <vehicle id="81" x="-115.160968" y="36.166198" angle="81.805314"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="12.808982" lane=":5434499061_4_0" slope="0.000000"/>
    <vehicle id="85" x="-115.205802" y="36.164687" angle="269.320812"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="63.118926" lane="14294335#2_0" slope="0.000000"/>
    <vehicle id="88" x="-115.183545" y="36.166248" angle="90.034502"
    type="DEFAULT_VEHTYPE" speed="0.000000" pos="68.138997" lane="-493377035#5_0" slope="0.000000"/>
</timestep>
```

Figure 2 A screenshot of fcd output trace

```
119   08:01:18   -115.103167 36.193878   70   27.830
21    08:01:18   -115.192708 36.162884   70   0.000
51    08:01:18   -115.156271 36.201024   70   47.040
56    08:01:18   -115.152335 36.200284   70   38.087
74    08:01:18   -115.147389 36.191423   70   81.282
83    08:01:18   -115.106392 36.188338   70   43.329
119   08:01:19   -115.103169 36.193806   70   28.933
21    08:01:19   -115.192708 36.162884   70   0.000
51    08:01:19   -115.15627  36.201102   70   31.049
56    08:01:19   -115.152462 36.200284   70   41.157
74    08:01:19   -115.147378 36.191218   70   82.036
83    08:01:19   -115.106553 36.188342   70   52.130
118   08:01:20   -115.105651 36.192081   70   52.295
119   08:01:20   -115.103168 36.193714   70   36.447
21    08:01:20   -115.192708 36.162884   70   0.000
51    08:01:20   -115.156267 36.201156   70   21.428
56    08:01:20   -115.152576 36.200285   70   36.958
74    08:01:20   -115.147367 36.191008   70   83.898
83    08:01:20   -115.106731 36.188346   70   57.780
```

Figure 3 A screen shot of a test trace

id, date, x, y, status, speedKMH, etc is selected as the format for the DES network simulation implemented in MATLAB. A usage example of this tool is as below and a screenshot of the final output test trace is shown in Figure 3, where the leftmost column to the rightmost column represent vehicle ID, timestamp, longitude, latitude, status, and speed, respectively.

```
traceExporter.py --fcd-input <fcd-trace-file>
--gpsdat-output <test-trace-file>
```

Abbreviations

BS Base Station

CPT+ Compact Prediction Tree plus

CPT Compact Prediction Tree

CP Content Provider

D2D Device-to-Device

DASH Dynamic Adaptive Streaming over HTTP

DSRC Dedicated Short Range Communications

HCPC Hybrid cMAB Proactive Caching System

HTTP Hypertext Transfer Protocol

HetNet Heterogeneous Network

ICN Information Centric Networking

LSTM Long Short-Term Memory

LTE Long Term Evolution

MAB Multi-armed Bandit

MANETs Mobile Ad hoc Networks

MCC Mobile Cloud Computing

MDP Markov Decision Process

MEC Mobile Edge Computing

ML Machine Learning

MNO Mobile Network Operator

NDN Named Data Networking

NFV Network Function Virtualisation

OBU On-board Unit

OSN Online Social Network

PPM Prediction based on Partial Matching

QL Q-Learning

QoE Quality of Experience

QoS Quality of Service

RAN Radio Access Network

RAT Radio Access Technology

RL Reinforcement Learning

RSU Road-side Unit

SCN Small Cell Network

SDN Software Defined Network

SL Subjective Logic

SUMO Simulator of Urban MObility

UE User Equipment

V2I Vehicle-to-Infrastructure

V2R Vehicle-to-Roadside units

V2V Vehicle-to-Vehicle

VANETs Vehicular Ad hoc Networks

VCCN Vehicular Content-Centric Network

VLC Visible Light Communication

WAVE Wireless in Vehicular Environments

WoLF Win-or-Learning-Fast

cMAB Contextual multi-armed bandit

References

[1] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[2] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 320–351, 2020.

[3] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: an overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[4] T. Gueniche, P. Fournier-Viger, R. Raman, and V. S. Tseng, "CPT+: Decreasing the time/space complexity of the compact prediction tree," in *Advances in Knowledge Discovery and Data Mining*, T. Cao, E.-P. Lim, Z.-H. Zhou, T.-B. Ho, D. Cheung, and H. Motoda, Eds. Cham: Springer International Publishing, 2015, pp. 625–636.

[5] T. Gueniche, P. Fournier Viger, and V. Tseng, "Compact prediction tree: A lossless model for accurate sequence prediction," vol. 8347, 12 2013.

[6] Q. Wang and D. Grace, "Proactive Edge Caching in Vehicular Networks: An Online Bandit Learning Approach," July 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Proactive_Edge_Caching_in_Vehicular_Networks_An_Online_Bandit_Learning_Approach/20290953

[7] Q. Wang and D. Grace, "A hybrid proactive caching system in vehicular networks based on contextual multi-armed bandit learning," July 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/A_Hybrid_Proactive_Caching_System_in_Vehicular_Networks_based_on_Contextual_Multi-armed_Bandit_Learning/20291673

[8] Q. Wang and D. Grace, "Sequence prediction-based proactive caching in vehicular content networks," in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*, 2020, pp. 1–6.

[9] Z. Su, Y. Hui, T. H. Luan, Q. Liu, and R. Xing, *The Next Generation Vehicular Networks, Modeling, Algorithm and Applications*. Springer.

[10] S. Zhang, J. Chen, F. Lyu, N. Cheng, W. Shi, and X. Shen, "Vehicular communication networks in the automated driving era," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 26–32, 2018.

[11] P. Dai, K. Liu, X. Wu, Y. Liao, V. C. S. Lee, and S. H. Son, "Bandwidth efficiency and service adaptiveness oriented data dissemination in heterogeneous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6585–6598, 2018.

[12] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted over-the-top edge prefetching for hierarchical vanets," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1786–1801, 2018.

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[14] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2020.

[15] Z. Su, Y. Hui, Q. Xu, T. Yang, J. Liu, and Y. Jia, "An edge caching scheme to distribute content in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5346–5356, 2018.

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

[17] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017.

[18] N. Nakamura, Y. Niimi, and S. Ishihara, "Live vanet cdn: Adaptive data dissemination scheme for location-dependent data in vanets," in *2013 IEEE Vehicular Networking Conference*, 2013, pp. 95–102.

[19] L. Hou, L. Lei, K. Zheng, and X. Wang, "A Q-learning-based proactive caching strategy for non-safety related services in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4512–4520, 2019.

[20] H. Khelifi, S. Luo, B. Nour, A. Sellami, H. Moungla, and F. Naït-Abdesselam, "An optimized proactive caching scheme based on mobility prediction for vehicular networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997.

[22] N. Morozs, T. Clarke, and D. Grace, "Distributed heuristically accelerated Q-learning for robust cognitive spectrum management in lte cellular systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 4, pp. 817–825, 2016.

[23] A. Mahajan and D. Teneketzis, *Multi-Armed Bandit Problems*. Boston, MA: Springer US, 2008, pp. 121–151.

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

[25] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE-Acm Transactions on Networking*, vol. 22, no. 5, pp. 1444–1462, October 2014.

[26] Z. Chang, L. Lei, Z. Y. Zhou, S. W. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 28–35, June 2018.

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

[28] E. Baştuğ, M. Bennis, and M. Debbah, "Cache-enabled small cell networks: Modeling and tradeoffs," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, 2014, pp. 649–653.

[29] B. Bai, L. Wang, Z. Han, W. Chen, and T. Svensson, "Caching based socially-aware D2D communications in wireless content delivery networks: a hypergraph framework," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 74–81, 2016.

[30] B. Q. Chen, C. Y. Yang, and G. Wang, "High-throughput opportunistic cooperative device-to-device communications with caching," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7527–7539, August 2017.

[31] 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*, vol. 5, no. 1, pp. 1–20, 2014.

[32] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, p. 5–12, nov 2013.

[33] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, May 2014.

[34] B. Chen and C. Yang, "Caching policy optimization for D2D communications by learning user preference," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1–6.

[35] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[36] J. Jiao, X. Hong, and J. Shi, "Proactive content delivery for vehicles over cellular networks: The fundamental benefits of computing and caching," *China Communications*, vol. 15, no. 7, pp. 88–97, 2018.

[37] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. Mini, and A. A. Loureiro, "Data communication in vanets: Protocols, applications and challenges," *Ad Hoc Networks*, vol. 44, pp. 90–103, 2016.

[38] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.

[39] Y. Y. Nasrallah, I. Al-Anbagi, and H. T. Mouftah, "Enhanced algorithms for the IEEE 802.11p deployment in vehicular ad hoc networks," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–5.

[40] M. Laroui, A. Sellami, B. Nour, H. Moungla, H. Afifi, and S. B. Hacene, "Driving path stability in vanets," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[41] M. Balfaqih, M. Ismail, R. Nordin, and Z. A. Balfaqih, "802.21-assisted distributed mobility management solution in vehicular networks," *IEEE Access*, vol. 5, pp. 9518–9532, 2017.

[42] N. Gupta, A. Prakash, and R. Tripathi, "Medium access control protocols for safety applications in vehicular ad-hoc network: A classification and comprehensive survey," *Vehicular Communications*, vol. 2, no. 4, pp. 223–237, 2015.

[43] N. M. Huong, A. Benslimane, and D. J. Deng, "Reliable broadcasting using polling scheme based receiver for safety applications in vehicular networks," *Vehicular Communications*, vol. 4, pp. 1–14, April 2016.

[44] S. Ucar, S. C. Ergen, and O. Ozkasap, "Multihop-cluster-based IEEE 802.11p and LTE hybrid architecture for vanet safety message dissemination," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2621–2636, 2016.

[45] S. Uçar, S. Ergen, and Özkasap, "Visible light communication in vehicular ad-hoc networks," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, 2016, pp. 881–884.

[46] R. Hussain, S. Kim, and H. Oh, "Traffic information dissemination system: Extending cooperative awareness among smart vehicles with only single-hop beacons in VANET," *Wireless Personal Communications*, vol. 88, no. 2, pp. 151–172, May 2016.

[47] Q. Zhang, H. Zheng, J. H. Lan, J. W. An, and H. Peng, "An autonomous information collection and dissemination model for large-scale urban road networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1085–1095, April 2016.

[48] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu, "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, 2015, pp. 1207–1212.

[49] Z. W. Hu, Z. J. Zheng, T. Wang, L. Y. Song, and X. M. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE*

Transactions on Wireless Communications, vol. 16, no. 10, pp. 6321–6334, October 2017.

[50] T. Liu, S. Zhou, and Z. Niu, "Joint optimization of cache allocation and content placement in urban vehicular networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[51] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, Jun. 2018.

[52] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H.-C. Chao, "An efficient cache strategy in information centric networking vehicle-to-vehicle scenario," *IEEE Access*, vol. 5, pp. 12 657–12 667, 2017.

[53] D. D. Van, Q. S. Ai, Q. Liu, and D. T. Huynh, "Efficient caching strategy in content-centric networking for vehicular ad-hoc network applications," *IET Intelligent Transport Systems*, vol. 12, no. 7, pp. 703–711, September 2018.

[54] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.

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

[56] K. N. Doan, T. Van Nguyen, T. Q. S. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3128–3140, 2018.

[57] 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*, vol. 36, no. 6, pp. 1331–1344, 2018.

[58] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: Strategies, challenges, and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1443–1474, 2018.

[59] D. Grewe, M. Wagner, and H. Frey, "Perceive: Proactive caching in ICN-based VANETs," in *2016 IEEE Vehicular Networking Conference (VNC)*, 2016, pp. 1–8.

[60] D. Zhao, Y. Gao, Z. Zhang, Y. Zhang, and T. Luo, "Prediction of vehicle motion based on Markov model," in *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, 2017, pp. 205–209.

[61] S. Parija, R. K. Ranjan, and P. K. Sahu, "Location prediction of mobility management using neural network techniques in cellular network," in *2013 International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT)*, 2013, pp. 1–4.

[62] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[63] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 3, p. 22–36, Jul 1996.

[64] J. Pitkow and P. Pirolli, "Mining longest repeated subsequences to predict world wide web surfing," in *Second USENIX Symposium on Internet Technologies & Systems (USITS 99)*. Boulder, CO: USENIX Association, Oct 1999.

[65] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27 699–27 719, 2019.

[66] Z. Wang, W. Zhu, X. Chen, L. Sun, J. Liu, M. Chen, P. Cui, and S. Yang, "Propagation-based social-aware multimedia content distribution," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1s, pp. 1–20, 2013.

[67] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting video server selection strategies in the youtube CDN," in *2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 248–257.

[68] Z. Xiaoqiang, Z. Min, and W. Muqing, "An in-network caching scheme based on betweenness and content popularity prediction in content-centric networking,"

in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–6.

[69] Y. Zhang, X. Tan, and W. Li, "PPC: Popularity prediction caching in ICN," *IEEE Communications Letters*, vol. 22, no. 1, pp. 5–8, 2018.

[70] N. Ben Hassine, D. Marinca, P. Minet, and D. Barth, "Popularity prediction in content delivery networks," in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2015, pp. 2083–2088.

[71] W. Hoiles, O. N. Gharehshiran, V. Krishnamurthy, N.-D. Ngoc-Dung-Dào, and H. Zhang, "Adaptive caching in the youtube content distribution network: A revealed preference game-theoretic learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 1, no. 1, pp. 71–85, 2015.

[72] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[73] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[74] J. E. D. Krajzewicz, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, 2012.

[75] S. A. Elsayed, S. Abdelhamid, and H. S. Hassanein, "Proactive caching at parked vehicles for social networking," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[76] D. Grewe, M. Wagner, S. Schildt, M. Arumaithurai, and H. Frey, "Caching-as-a-service in virtualized caches for information-centric connected vehicle environments," in *2018 IEEE Vehicular Networking Conference (VNC)*, 2018, pp. 1–8.

[77] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular mobility simulation for VANETs," in *40th Annual Simulation Symposium (ANSS'07)*. IEEE, 2007, pp. 301–309.

[78] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[79] G. A. Wainer and P. J. Mosterman, *Discrete-event modeling and simulation: theory and applications*. CRC press, 2018.

[80] J. Banks, I. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*. Pearson, 2005.

[81] MATLAB, *version 9.4.0 (R2018a)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[82] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–10.

[83] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[84] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018.

[85] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[86] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[87] W. Wang, A. Kwasinski, D. Niyato, and Z. Han, "A survey on applications of model-free strategy learning in cognitive wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1717–1757, 2016.

[88] D. Russo, B. V. Roy, A. Kazerouni, and I. Osband, "A tutorial on thompson sampling," *CoRR*, vol. abs/1707.02038, 2017. [Online]. Available: http://arxiv.org/abs/1707.02038

[89] S. J. Russell, *Artificial intelligence: a modern approach.* Pearson Education, Inc., 2010.

[90] M. A. Wiering, "Reinforcement learning in dynamic environments using instantiated information," in *Machine Learning: Proceedings of the Eighteenth International Conference (ICML2001)*, 2001, pp. 585–592.

[91] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[92] N. Morozs, "Accelerating reinforcement learning for dynamic spectrum access in cognitive wireless networks," 2015.

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

[94] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[95] P. Dai, Z. Hang, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. S. Lee, "Multi-armed bandit learning for computation-intensive services in MEC-empowered vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7821–7834, 2020.

[96] Y. Miao, Y. Hao, M. Chen, H. Gharavi, and K. Hwang, "Intelligent task caching in edge cloud via bandit learning," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 625–637, 2020.

[97] S. Maghsudi and E. Hossain, "Multi-armed bandits with application to 5G small cells," *IEEE Wireless Communications*, vol. 23, no. 3, pp. 64–73, 2016.

[98] X. Xu, M. Tao, and C. Shen, "Collaborative multi-agent multi-armed bandit learning for small-cell caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2570–2585, 2020.

[99] F. M. Zennaro and A. Jøsang, "Using subjective logic to estimate uncertainty in multi-armed bandit problems," *arXiv preprint arXiv:2008.07386*, 2020.

[100] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *arXiv preprint arXiv:1204.5721*, 2012.

[101] T. Stockhammer, "Dynamic adaptive streaming over HTTP–standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[102] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., and V. S. Tseng, "SPMF: A Java open-source pattern mining library," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 3389–3393, 2014. [Online]. Available: http://www.philippe-fournier-viger.com/spmf/

[103] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[104] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction," *CoRR*, vol. abs/1910.09457, 2019. [Online]. Available: http://arxiv.org/abs/1910.09457

[105] A. Jøsang, "Artificial reasoning with subjective logic," in *Proceedings of the second Australian workshop on commonsense reasoning*, vol. 48. Citeseer, 1997, p. 34.

[106] T. Muller, "An unforeseen equivalence between uncertainty and entropy," in *IFIP International Conference on Trust Management*. Springer, 2019, pp. 57–72.

[107] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *arXiv preprint arXiv:1806.01768*, 2018.

[108] L. M. Kaplan, F. Cerutti, M. Sensoy, A. D. Preece, and P. Sullivan, "Uncertainty aware AI ML: why and how," *CoRR*, vol. abs/1809.07882, 2018. [Online]. Available: http://arxiv.org/abs/1809.07882

[109] T. Jiang, D. Grace, and P. D. Mitchell, "Efficient exploration in reinforcement learning-based cognitive radio spectrum sharing," *IET communications*, vol. 5, no. 10, pp. 1309–1317, 2011.

[110] A. Josang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2016. [Online]. Available: https://books.google.no/books?id=bJkJkAEACAAJ

[111] B. Xin, H. Yu, Y. Qin, Q. Tang, and Z. Zhu, "Exploration entropy for reinforcement learning," *Mathematical Problems in Engineering*, 2020.

[112] M. Bennis and D. Niyato, "A Q-learning based approach to interference avoidance in self-organized femtocell networks," in *2010 IEEE Globecom Workshops*, 2010, pp. 706–710.

[113] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[114] Q. Wang and D. Grace, "Proactive edge caching in vehicular networks: An online bandit learning approach," *Manuscript submitted to IEEE Transactions on Mobile Computing*, 2022.

[115] A. H. Ko, R. Sabourin, and F. Gagnon, "Performance of distributed multi-agent multi-state reinforcement spectrum management using different exploration schemes," *Expert systems with applications*, vol. 40, no. 10, pp. 4115–4126, 2013.

[116] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[117] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[118] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[119] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: architecture, interfaces, algorithms, security, and research challenges," *CoRR*, vol. abs/2202.01032, 2022. [Online]. Available: https://arxiv.org/abs/2202.01032