Edge Cloud and Network Optimization for Mobile Augmented Reality

Huang, Zhaohui

Awarding institution:
King's College London

Edge Cloud and Network Optimization for Mobile Augmented Reality

Zhaohui Huang

Supervisor: Prof. Vasilis Friderikos
Prof. Osvaldo Simeone

The Department of Engineering
King's College London

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2023

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Zhaohui Huang
July 2023

Acknowledgements

Studying Phd at King's College London is quite challenging, especially confronted with Covid-19. I would like to first appreciate much support from my supervisor Dr Vasilis Friderikos. With lots of deep discussions and his wise suggestions, I manage to instruct my research and pursue my phd degree.

I also need to thank all my my colleagues and friends in the Centre for Telecommunications Research that helped me with my research and cared me about my life. I would like to thank to Dr Jongyul Lee and Dr Yantong Wang for guiding me and spending an unforgettable time together.

Finally, I should thank my family, my father Guangxiang Huang and my mother Fenghong Yao, for their consistent support and unconditional love. Without their encouragement, I could not overcome nervousness, loneliness and all other challenges.

Abstract

Mobile augmented reality (MAR) applications are still in a rather embryonic state but they are currently attracting significant attention from both academic and industry stemming from increased capabilities from the network side as well as the terminals. The ability of augmented reality services to overlay digital content over the physical space and surroundings result in an immersive metaverse type of environment which opens up a plethora of potential use cases and applications. As with respect to mobile wireless networks, mobility has been proved to be an important factor in service latency of MAR systems. Previous research mainly focused on various techniques of offloading processing without considering the decomposition of MAR functions and the effect of mobility in an explicit manner.

To this end, our works in chapter 3 combine the mobile edge clouds (MEC) and MAR to propose an optimization framework with a rich set of MAR specific constraints. In chapter 4, we then consider the content aware aspect and allow the proactive caching of high probability 2D field of views (FoVs) of AR Objects (AROs) to be stored instead of caching the significantly larger and complex 3D original AROs. In chapter 5, multiple view streams could also be brought in as another solution to proactive caching. In this case, popular view streams are precached with preferred 3D AROs to improve the overall user experience during different mobility events. In chapter 6, MAR is incorporated with metaverse but such an extension seeks reliable and high quality support for the foreground interactions and background contents from these applications, which intensifies their consumption of energy, caching and computing resources. To tackle these challenges, a more flexible request assignment and resource allocation with more efficient processing are proposed through anchoring decomposed metaverse AR services at different edge nodes and proactively caching background metaverse region models embedded with target AROs. Advanced terminals capabilities are also considered to further reduce service delay at an acceptable cost of energy consumption.

As illustrated by a broad set of numerical investigations, the proposed set of solutions can efficiently decompose the MAR service to available edge cloud resources and hence increase decision making quality compared to other previously proposed and baseline schemes.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Mobile Augmented Reality (MAR) Applications in Metaverse

Augmented reality (AR) attracts lots of attention on 5G and beyond networks due to its promising growth in marketing and practice. Different from virtual reality (VR) that separates the user from reality completely, AR represents a typical medium where overlay virtual content on the physical world [80]. Different from traditional AR, MAR applications enable the further alignment between virtual content and physical world that improves the user perception during a mobility event. MAR could be extended and enhanced in wireless edge supported metaverse by today's available technologies like digital twin and head-mounted display (HMD) rendering [112][28]. Users could have mixed experience seamlessly between metaverse and physical world through various metaverse AR applications like massively multiplayer online (MMO) video games and virtual concerts [112]. Users equipped with MAR devices can upload and analyze their environment through AR customization to achieve appropriate AR objects (AROs) and access the metaverse in mobile edge networks [111]. Rendering 3-dimensional (3D) AR objects (AROs) with the background virtual environment and updating in metaverse consume lots of energy and can be quite demanding for caching and computing resources [111][62]. Hence, it proposes a series of higher requirements for AR data storage, processing and communication at the same time.

In metaverse, the user can build interactions through the corresponding virtual image, known as avatar [28]. Such foreground interactions may activate related MAR applications and a 3D model of metaverse with target AROs at a defined amalgamated virtual and physical location should be presented to the user according to his view port [33][57]. Although the background scene itself could be regarded as static or slowly changing, the AROs in foreground interactions have to be updated more frequently. However, rendering delicate 3D AROs and models based on user mobility are quite demanding in time and energy while mobile devices suffer from a limited computing and battery capacity [111]. Hence, it could be more reasonable and flexible to leave some complex tasks to a more powerful server but this causes more communication cost at the same time. According to Fig. (1.1), the edge nodes are able to execute most tasks with low extra latency cost while the far-away cloud data centers could store all necessary data for metaverse and provide a final backup support. In addition, service decomposition has widely been accepted as an effective method breaking a complex service into multiple chained microsevices to better support the task assignment and resource allocation. In this way, decomposed MAR microservices could be rearranged to suitable edge servers according to current computing and caching resources.

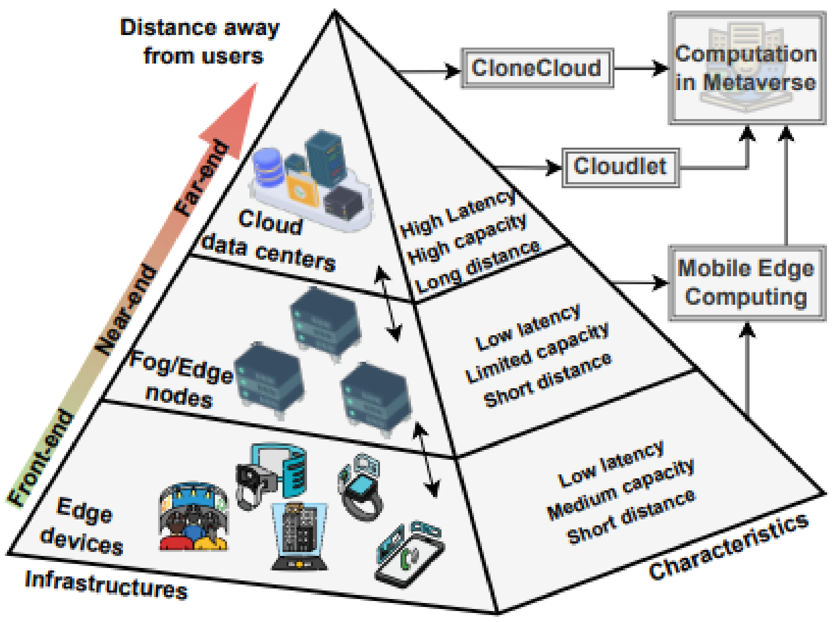


Fig. 1.1 Characteristics and Distances to Users of Different Network Infrastructures supporting Metaverse

Increased capabilities of mobile devices (e.g. over 1000 devices/km^2 [111]) brings about challenges in processing their target AROs according to different user mobility events. Also, the addition of AR scanners on mobile phones further accelerates the immersion of AROs in the physical world as they will ask for added flexibility and accuracy in placing AROs due to their inherent capability of creating a depth map of the surrounding space. In that case, the provision of an intelligent edge cloud is required that 'follows' the users based on their mobility whilst taking into account resource requirements in different areas of the network due to the proliferation of AROs. As illustrated in Figure (1.2), until now network provision and resource management policies have considered hot spot areas in terms of number of users, however the envisioned virtual immersion scenario might create a new form of hot spot area where there is significant virtual content to be overlaid on the physical space requiring in that respect new form of proactive network and edge cloud resource management [42]. Noticing that rendering 3D AROs itself is still time consuming and seriously affected by users' view ports [62], replacing some of them by 2D images with an acceptable level of loss in quality might be a way to further cut down the latency and energy cost. Existing techniques allow them to be generated from 3D AROs according to user view ports and even reconstructed or conversed back [41][77][102]. If more focusing on view ports, creating multiple view streams embedded with AROs might be another solution. In this case, instead of directly manipulating AROs, view streams are allocated and proactively cached on edge servers.

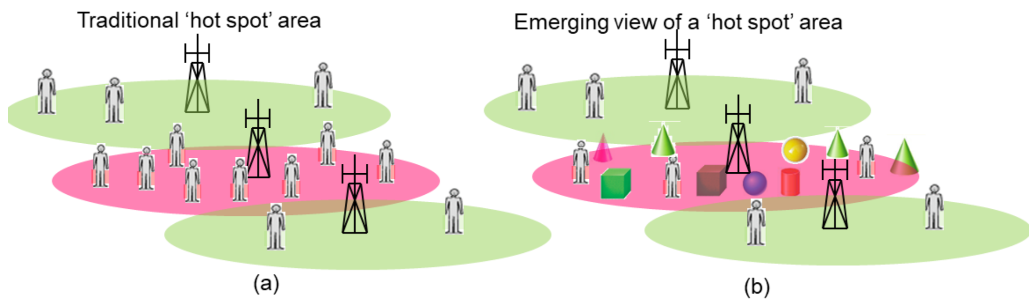

Traditional 'hot spot' area
Emerging view of a 'hot spot' area
(a)
(b)

Fig. 1.2 (a) Traditional view of a hot spot area in terms of the number of users and (b) an emerging view where a hot spot area might relate to an increased amount of virtual objects to be overlaid to the physical,world and not necessarily to the number of mobile users, requiring in that respect the provision of significant amount of network resources [42].

Thus, MAR in metaverse needs to process and present AROs in a timely manner without consuming much energy or sacrificing much quality. Clearly, it is not practical to leave either the whole MAR service or required AROs executed at terminals which is a heavy burden and hinders the synchronization between terminals and the metaverse. In our works, a series of methods like decomposition, 2D Field of View, Multi-streaming and heuristic algorithms based on optimal solutions are explored to provide efficient and high quality solutions for MAR service on an edge cloud (EC) supported network.

1.2 Motivation and Objectives

Our works are motivated by a series of objects in this area and they are listed and explained below.

Consideration of User Physical Mobility: Fig. (1.3) displays a toy example regarding EC support for an AR application and the effect of mobility on the communication between the end device and the selected EC. From the figure, changing the point of attachment

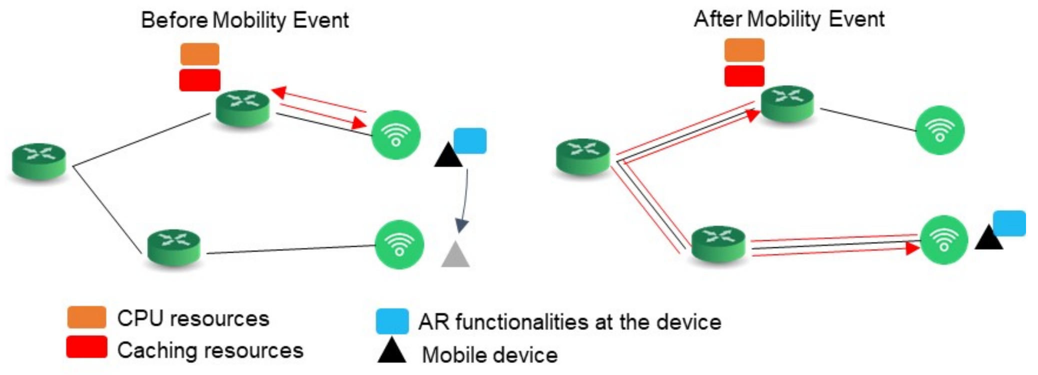


Fig. 1.3 Effect of user mobility on the selection of EC for supporting mobile augmented reality applications. Latency significantly increases when changing the point of attachment. [42]

during an MAR session might have a detrimental effect in terms of overall latency since the new point of attachment is not optimized in terms of the EC support for such latency sensitive application.

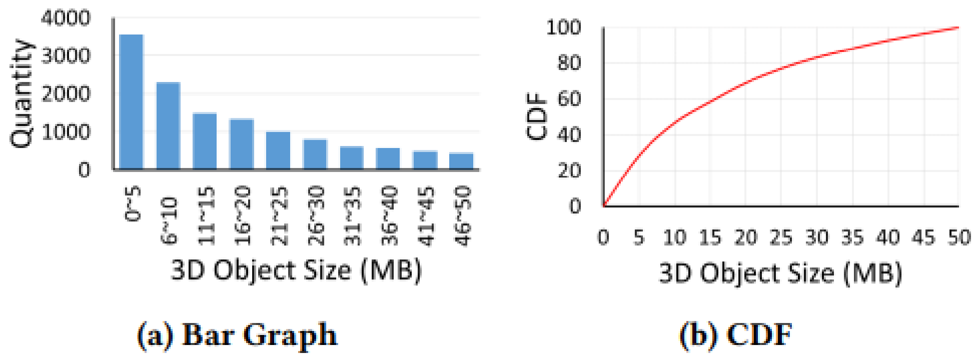


Fig. 1.4 3D AROs: Size vs Quantity (a) and Size vs Cumulative Distribution Function (CDF) [117]

Proactive caching and processing of 3D AROs: As shown by (1.4), the size of 3D AROs ranges from several MBytes to tens of MBytes. Although most AROs are within 20 MBytes, it is still much larger than 2D images which are usually tens or hundreds of KBytes. Hence, it leaves a heavy burden of computing and storage for ECs.

Service Deployment and resource allocation: It is about placing decomposed MAR functions and allocating sufficient resources to meet QoS requirements. Even for a static scenario of this problem (without user mobility), the complexity grows in a rather (high degree) polynomial manner. The underlying complexity of this problem is illustrated by Fig. (1.5); this toy example illustrates the number of possible configurations (in this case 24) in the case of 4 CPU cores, 3 possible utilization levels and 2 memory allocations

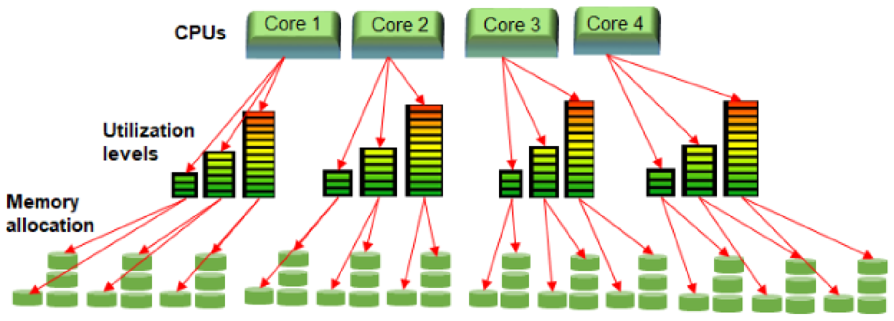


Fig. 1.5 Complexity of decomposed MAR functions (Microservices) Problem via a toy example

(4x3x2). It is evident that as the level of granularity increases the complexity increases substantially (cubic degree of polynomial) and therefore suitable mechanisms should be in place to allow for an autonomous operation that can allow efficient decision policies to be implemented.

Maintaining Interactions: It not only means the communication among decomposed MAR functions but also includes clients. Too many requests from users may lead to a serious congestion in the network. Thus, a balance should be achieved between the simplicity of the client-side software and extensive implementation of decomposed MAR functions at edge nodes. In addition, the entry module and the workload of decomposed MAR functions should be carefully considered to avoid potential risks of overloading. Similarly, the path-planning problem is closely related to service deployment and resource allocation. Machine learning based designs seem to have advantages in finding good-quality paths approaching to optimal ones within the service latency limit. In our works, long short term memory and simulated annealing are brought in to provide high quality predictions efficiently.

Fault tolerance Mechanism: The MAR should keep consistency of transactions during the recovery from failures within milliseconds. A proper mechanism including appropriate warning messages and redirection is required to handle partial failures to avoid the ripple effect (disabling the whole system). The far-away cloud and a Feasibility Check stage are brought in our works to help handle potential risks.

1.3 Contributions

This thesis is based on our main works including 4 conference papers, 3 journals and a book chapter. These works are listed below with brief explanation of their contributions.

Decomposition granularity and Machine Learning in MAR service optimization: Through a review of emerging advanced services, challenges and typical solutions in

5G and beyond mobile communication networks, especially MAR, we shed light into the architectural aspects of the above envisioned virtualized mobile network ecosystem. More specifically, and as an example of advanced services, we discuss how AR applications can be decomposed into several service components that can be located either at the end-user terminal or at the edge cloud. Furthermore, we examine how this inherent service decomposition can be considered under the lenses of VNF based microservices architectures. Key areas where data-driven machine learning techniques can ease the network orchestration are also outlined. These contributions are included in "Granular vnf-based microservices: Advanced service decomposition and the role of machine learning techniques" [49].

Mobility-aware optimization with EC support: Through explicitly considering the decomposition of MAR functions and the effect of mobility, an optimization framework is proposed that also includes a rich set of AR specific constraints, provides service decomposition and allows a trade off between service latency and frame accuracy to improve the communication efficiency. In addition, based on this optimal scheme, a simulated annealing based mobility aware AR (SAMAR) algorithm and a long short term memory based scheme (LSTM) neural network for predicting edge clouds for anchoring MAR functions are designed for achieving high-quality solutions whilst being amenable for real-time implementation. These contributions are included in "Proactive edge cloud optimization for mobile augmented reality applications" [42] and "Optimal service decomposition for mobile augmented reality with edge cloud support" [48].

Optimized replacement for complex 3D AROs using 2D FoVs: This idea considers the novel content-aware aspect of the proposed solution allows for proactive caching of high probability 2D FoVs of the AROs to be stored instead of caching the significantly larger and more complex 3D original AROs. To this end, a joint optimization scheme (Optim) considering mobility and the trade-off between delay, storage capacity and FoV allocation is proposed. A nominal LSTM deep neural network is further explored to provide efficient pro-active decision making in real-time. These contributions are included in "Field of view aware proactive caching for mobile augmented reality applications" [43].

AROs embedded in multi view streams: TO achieve better performance in ARO sharing and synchronization, we explicitly utilize the notion of content popularity not only to synthetic objects but also video view streams. In this case, popular view streams are cached in a proactive manner, together with preferred AROs, in selected edge caching locations to improve the overall user experience during different mobility events. To achieve that, a joint optimization problem considering mobility, service decomposition, and the balance between service delay and the preference of view streams and embedded AROs is proposed. To tackle the curse of dimensionality of the optimization problem, a nominal Long Short Term Memory (LSTM) neural network is proposed, which is trained offline with optimal solutions and provide high quality real-time decision making during inference. These contributions are included in "Network resource optimization for multi-view streaming mobile augmented reality" [45] and "Optimal proactive caching for multi-view streaming mobile augmented reality" [46].

Terminal aware extension of MAR into metaverse: The metaverse MAR service is decomposed and anchored at suitable edge caching/computing nodes in 5G and beyond networks to enable efficient processing of background metaverse region models embedded with target AROs. To achieve that, a joint optimization problem is proposed, which explicitly considers the user physical mobility, service decomposition, and the balance between service delay, user perception quality and power consumption. In addition, the terminals are brought in and hence get another optimized framework for further comparison. These contributions are included in "Mobility aware optimization in the metaverse" [44] and "Optimal mobility aware wireless edge cloud support for the metaverse" [47].

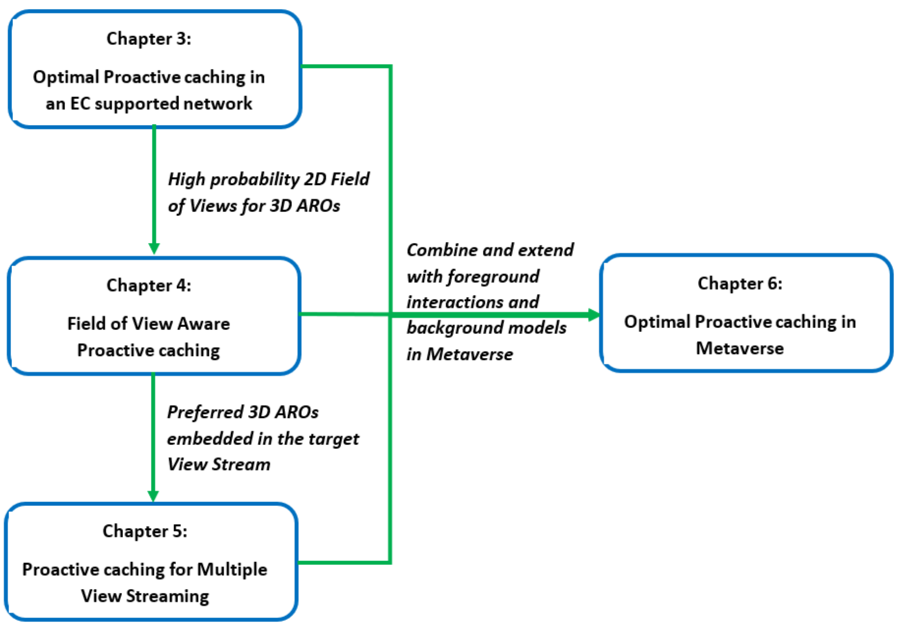

Chapter 3:
Optimal Proactive caching in an EC supported network
High probability 2D Field of Views for 3D AROs
Chapter 4:
Field of View Aware Proactive caching
Preferred 3D AROs embedded in the target View Stream
Chapter 5:
Proactive caching for Multiple View Streaming
Combine and extend with foreground interactions and background models in Metaverse
Chapter 6:
Optimal Proactive caching in Metaverse

Fig. 1.6 Links of Technical Chapters

Fig. (1.6) reveals the relation between following technical chapters. In Chapter 3, we start with a nominal scenario where proactive caching is considered under mobility events in an EC supported network. Noticing that 3D AROs are large for caching and complex for processing, we then replace them by popular 2D Field of Views and hence achieve the Field of View Aware proactive caching in Chapter 4. Undoubtedly, enabling such 2D replacements for 3D AROs brings down the general frame quality. To avoid degradation of user perception experience and unnecessary duplicate caching of same AROs required by different users, 3D AROs are set in multiple view streams and preferred ones are proactively cached; those techniques are detailed in Chapter 5. Finally, we study a scenario that integrates also foreground interactions and background models to capture a nominal metaverse type of environment. Thus, the extensions for typical metaverse type of AR applications are proposed in chapter 6.

Chapter 2

Related Work

In this chapter, related works are grouped in a top to bottom manner that starts with the general 5G/B5G networks, then focuses on internal microservice architectures requiring service decomposition and finally dives into metaverse supported MAR applications with their resource allocation issues. Through comparison, we identify technical gaps and explicitly show similarities and differences between our works and cited references.

2.1 5G and B5G Network Architecture

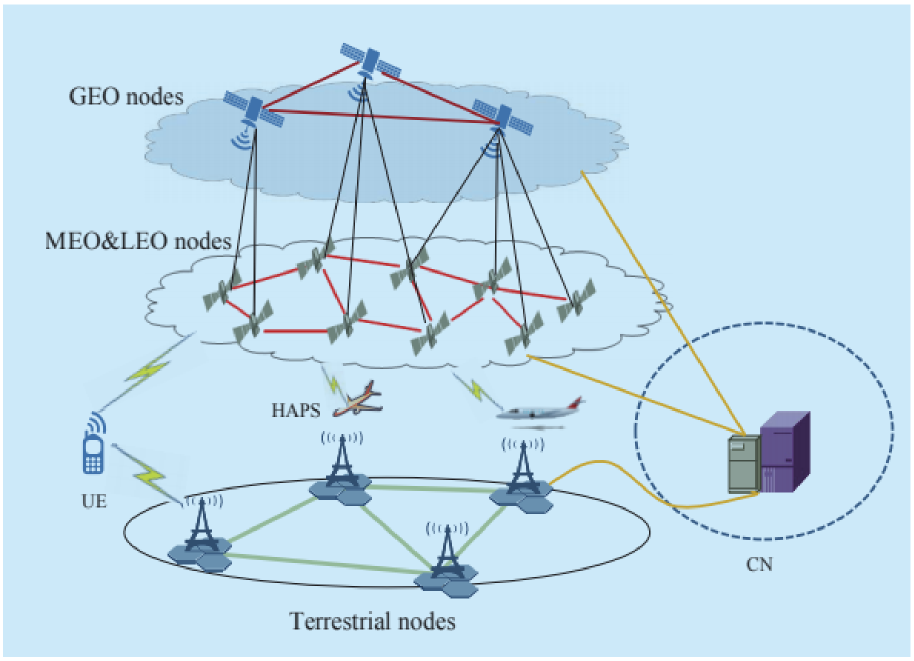


Fig. 2.1 A general 5G/B5G Architecture [17]

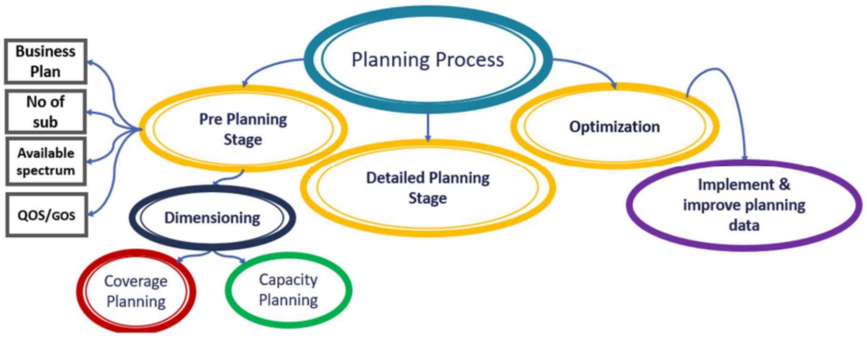


Fig. 2.2 The General Architectural design of B5G/6G Network Planning [29]

We have been experiencing the new era of 5G and beyond 5G networks (B5G) that brings about amazing elevation in service communication and quality. Naturally, we take 5G/B5G networks as basic research background where MAR applications should be deployed and executed. A general 5G and B5G architecture is shown by Fig. (2.1). Clearly, it is a multi-layer system including at least users equipped with mobile terminals, core networks, terrestrial stations and satellite stations. A series of researches focused on different components in the B5G/6G system have been proposed to improve their capabilities and related resource allocation. As shown by Fig. (2.2), aforementioned resource allocation and request assignment problem could be regarded as a typical type of network planning process consists of three main stages: pre-planning, detailed planning and optimization [29]. In the pre-planning stage, the number of network elements and their capacity are investigated or estimated through dimensioning. The later detailed planning stage further analyses the initial size and configuration of the given 5G/B5G network and provides automatic network planning through parameterization and prioritization, in which the planner minimizes co-channel inference and adjacent channel noise. Finally, the optimization stage operates measurements of network level and field test to filter and handle problematic planning data before implementation [29].

In this theis, we expect that the proposed set of algorithms to be embedded into an edge computing 5G and beyond platform. Native support of edge computing is being supported even from Release 17 within 3GPP. According to Rel-17 3Gpp, the edge data network provides the service to application clients and consists of edge application servers, edge enabler servers and edge configuration servers. The proposed schemes could run by a given module, known as Edge Application Server Discovery Function (EASDF), which acts as a Domain Name System (DNS) resolver and complements the DNS queries with user location and related information. At the same time, the enhancements in the User Equipment route-selection policy (URSP) could better support the distributed anchoring of applications and the relocation of the server. Thus, the EASDF module with the URSP policy enables the utilization of advanced allocation schemes in an EC supported network. The implementation of approaches further requires the dynamic availability (e.g. deployment changes and user mobility), network capability exposure (e.g.Network Exposure Function) and the support for seamless service continuity.

2.2 Service Decomposition

A series of recent researches provides an immersion into key technologies related to microservices and how advanced services can be decomposed in order to acquire the full benefits of such architectures. This section revolves around the case of MAR as an example of a rich service offering where the above-mentioned aspects need to be amalgamated in order to provide efficient service delivery; however, the overarching aspects are equally applicable to other rich service offerings beyond MAR.

Network Functions Virtualization (NFV) is one technique that contributes to the so called "softwarization" of the network. In essence, NFV decouples processing and controlling software from specialized hardware (e.g., load balancer, firewalls, TCP optimizer) [21][70], thereby leading to the replacement of physical network functions (PNFs) by monolithic ones [21][90]. In addition to that, NFV allows that each service can be represented as an ordered set of different distinct virtual network functions (which might be co-located or not) [61]. Based on the characteristics of a service request (for example deploying a MAR service or any other application), network operators can flexibly deploy a set of virtual network functions (VNFs) in the appropriate edge or core servers and chain them. In this case, the data flow is steered to go through all the VNFs in a pre-defined order.

It is worthy to point out that MAR applications can be decomposed into a set of atomic VNFs and apply a web-scale deployment to achieve better efficiency [25][1]. Broadly speaking, it can be split into four basic functions: Object Detection, Feature Extraction, Object Recognition and Object Matching [117][81]. Firstly, frames recorded by camera are detected and those with target objects are selected and uploaded. Then, feature points are extracted and provided for recognizing the original image. The work by [66], proposes an optimization engine for the increased precision on Object Detection with functions of model selection and resource mapping deployed at an edge cloud. Finally, it goes to the database to search and find an appropriate match. These functions can be extended or further decomposed to achieve a better granularity and satisfy the different requirements and constraints of the MAR system. For example, AR quality monitoring and AR adapting are executed on a user device for better presentation on screen. In a slightly different approach from [117] and [66], researchers focus on tailoring computationally intensive functions apart and regroup them on mobile devices [51][114] or edge/cloud servers

[67][92]. The MAR system designed for socialization extends the "Feature Extraction" with an offline probabilistic model that encodes feature descriptors to better compress them [117]. It also splits "Object Matching" into three minor steps: Template Matching, Object Tracking and Annotation Rendering. Specially focused on 3D objects, authors not only find a match for recognized objects but also calculate the position of the target in Template Matching [117].

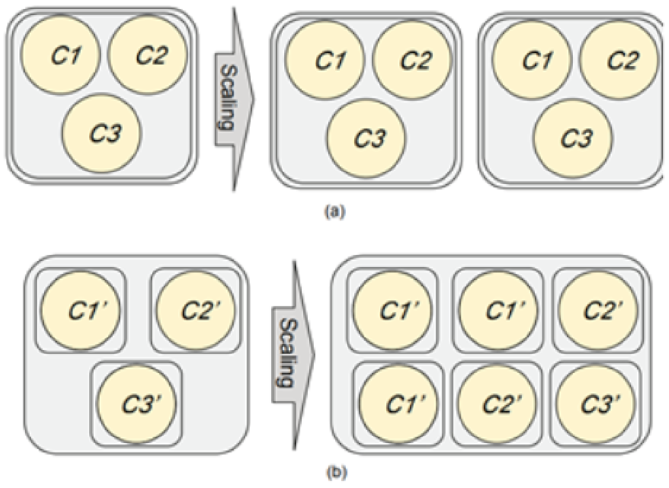


Fig. 2.3 (a) Monolithic Architecture (MA) & (b) Microservice Software Architecture (MSA) [21]

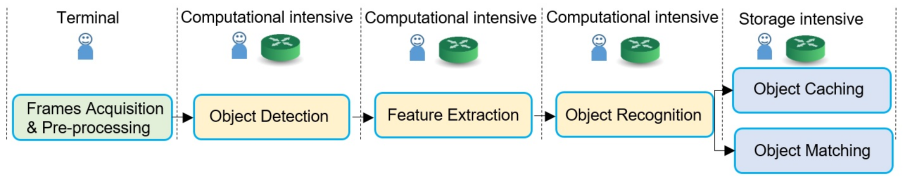


Fig. 2.4 An example chain of decomposed MAR functions applying MSA [42]

From the above discussion it becomes apparent that advanced services such as mobile AR can be decomposed since the service per se builds by several different complex functions. Therefore, the use of microservices, as explained in the sequel, is a natural fit to explore the inherent decomposition of emerging complex services and applications that embed multi-modal information and require various complex image/vision processing algorithms to run efficiently. After decomposition, the chaining and arranging of VNFs becomes an important issue. According to [21] and [27], monolithic architecture (MA) of VNFs allow lots of repeated functionalities and the corresponding structure grows, which eventually becomes a heavy burden for developing, managing, enabling efficient resource allocation and causes extra processing latency, especially when networks operators need to steer traffic in the sequences of chained VNFs in the so-called Service Function Chains

(SFC). The main difference between MA and Microservice Software Architecture (MSA) is illustrated by Fig. (2.3) [21]. MA ties the functions together while MSA decouples them and enables a more flexible arrangement. In other words, managing of the different functions can take place in a more granular manner using the MSA architecture whereas in MA any decision about resource usage will reflect all the functions that are bundled together. As a result, researchers like [21] and [56] adopt MSA as a better modular design with a more flexible service composition. Noticing that a predefined sequence also exists in MAR as explained earlier, the decomposed MAR microservices could also be assigned to different servers instead of being tied together as shown by Fig. (2.4).

2.3 Metaverse supported AR Applications

The problem of efficient resource allocation for supporting metaverse type of applications also starts to attract significant amount of attention and various aspects have been considered. In [36], the emphasis is placed in the synchronization of Internet of Things (IoT) services, in which they employ IoT devices to collect real world data for virtual service providers. Through calculating maximum awards, users could select the ideal virtual service provider. Researchers then proposes the game framework considers such reward allocation scheme and general metaverse sensing model [36]. In Fig. (2.5) the authors also adopt a game theoretic framework by considering tasks offloading between mobile devices based on Coded Distributed Computing (CDC) in a proposed vehicular metaverse environment. It integrates AR reality and cars' mobility data seamlessly to mix virtual and real world entertainment experiences for passengers [54]. Another framework proposed by [22] manages and allocates different types of metaverse applications so that common resources among them could be shared through a semi-Markov decision process and an optimal admission control scheme.

Although the above mentioned research works all provide an optimization framework related to resource allocation in metaverse like ours, they mainly handle the general metaverse applications without diving into a specific one. The work in [74] applies a set of proposed resource optimization schemes in a virtual education metaverse. More specifically, a stochastic optimal resource allocation scheme is developed with the aim to reduce the overall cost of incurred by a service provider. Similar to the service decomposition in this paper, they only upload and cache some parts of data or services to achieve reduced levels of delay and offer better privacy [74]. The work shown by Fig. (2.6) is closely relevant since in that paper not only latency but energy consumption is also considered as is the case of our proposed model that uses a multi-objective optimization approach [28]. For ultra-reliable and low-latency communication (URLLC) services, researchers bring in

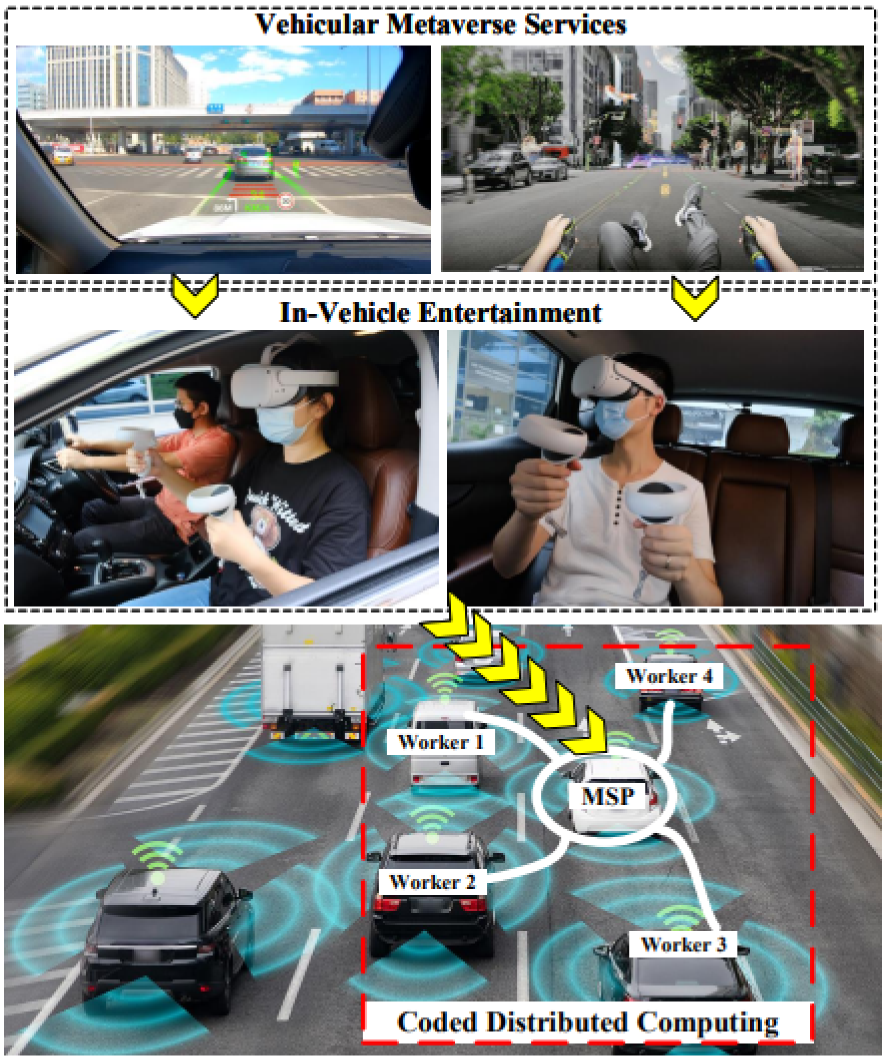


Fig. 2.5 An example of Coded Distributed Computing (CDC) in Vehicular Metaverse [54]

digital twins and deploy a mobility management entity for each access point to determine probabilities of resource allocation and data offloading [28]. Then, by applying a deep learning neuro network the proposed scheme tries to identify a suitable user association and an optimized resource allocation scheme for this association. While, in this paper, the core idea is to decompose he service and allow a flexible allocation across edge clouds by taking also into account user mobility. The work in[112] considers virtual reality applications in the metaverse and regard the service delivery as a series of events in the market, in which users are buyers and service provides are sellers. Hence, they apply Double Dutch Auction (DDA) to achieve common price through asynchronous and iterative biding stages [112]. They emphasize the quality of user perception experience by Structural SIMilarity (SSIM) and Video Multi-Method Assessment Fusion (VMAF). In our proposed framework, we also utilize the SSIM metric to determine the frame quality after integrating background

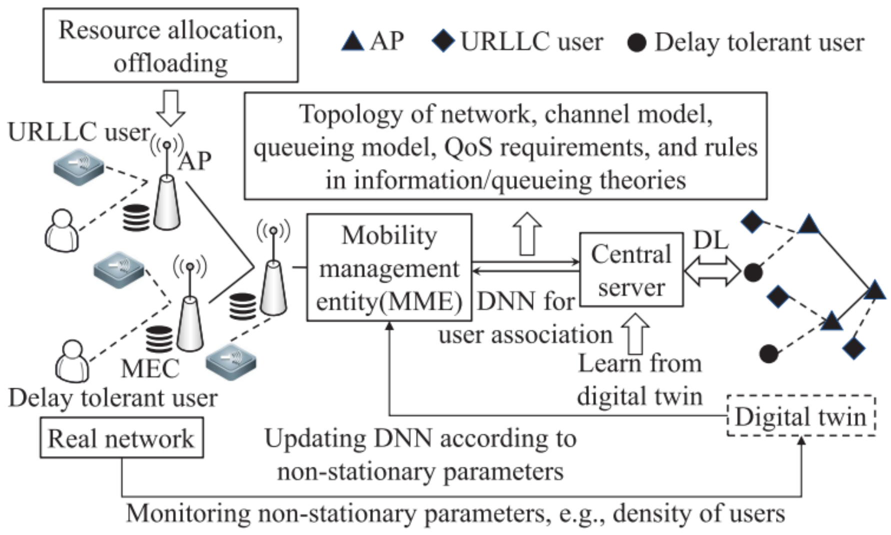

Resource allocation, offloading
▲AP ◆URLLC user ● Delay tolerant user
Topology of network, channel model, queueing model, QoS requirements, and rules in information/queueing theories
URLLC user
AP
Mobility management entity(MME)
Central server
DL
DNN for user association
Learn from digital twin
MEC
Delay tolerant user
Real network
Updating DNN according to non-stationary parameters
Digital twin
Monitoring non-stationary parameters, e.g., density of users

Fig. 2.6 applying Digital Twin for ultra-reliable and low-latency communications (URLLC) [28]

scene and AR contents [112]. The work in [112] further brings in a deep reinforcement learning-based auctioneer to reduce the information exchanging cost. While in this paper, a multi-objective optimization approach is taken where we aim to balance across different objectives using scalarization and considering user mobility in an explicit manner.

2.4 MAR Resource Allocation

The proposed Optim scheme in our works provides, in essence, an EC-supported MAR system whereas MAR service decomposition and user mobility is explicitly taken into account whilst balancing augmented reality objects caching, video frame length and accuracy. In that frontier, there has been various recent research efforts to improve overall MAR systems. The work in [34] proposes a trade-off model between workload and service latency, whereas the research in [66] proposes a dynamic adaptive protocol called DARE over the edge for an MAR system. As shown by Fig. (2.7), the DARE protocol achieves optimized Model Selection, Resource Allocation, Frame Rate and Frame Size by designing three hey components: Quality of Augmentation (QoA) Monitor, AR adaptation and Optimization Engine. MAR clients first send requests with measurements of network conditions to the edge server. Then the optimization engine determines the suitable size of frames with computation models for Object Detection. Requests are mapped for better computational resource allocation. Such optimized configuration data are sent back to clients so that the AR Adaption module will adapt rate and size of frames accordingly. Afterwards, suitable frames containing AR Objects are selected and transmitted for computing purposes. Finally, virtual content will be sent back to augments the target objects on screen [66]. In a similar setting, the work in [50] considers a cloud-based architecture to improve the performance of MAR applications in development, deployment and maintenance. The above mentioned work considered the MAR system as a whole, however some other relevant works considered specific aspect or a particular functionality of a MAR application. The work in [64] improves object detection, while the work in [52] improves object recognition. On the other hand, Wu et al. focus on finding a better local cache management strategy [109]. Furthermore, Li et al. reduce the amount of initially loaded model data to minimize computational pressure [62]. Among existing works, mobility of AR applications is mentioned but its effect on service latency is not considered explicitly as is the case in this work.

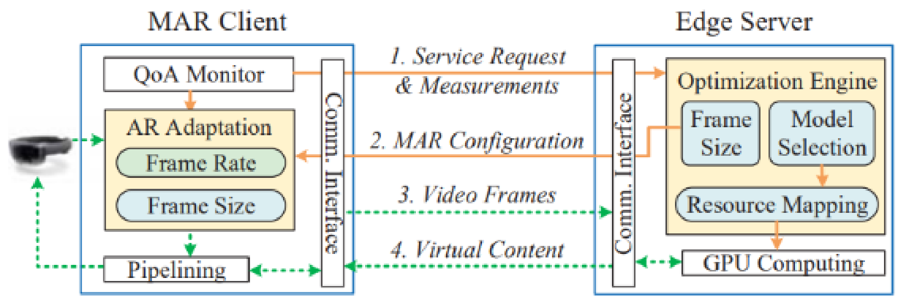


Fig. 2.7 Dynamic Adaptive Augmented Reality (DARE) protocol overview [66]

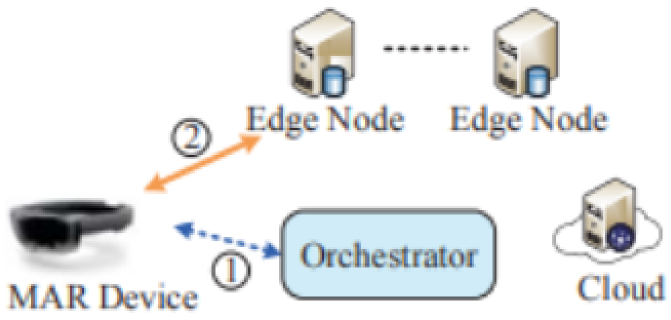


Fig. 2.8 An example scenario combining MEC and MAR [67]

In a closely related work [67] shown by Fig. (2.9), authors handling the relationship between latency and frame resolution through the combination of MEC and MAR. After given a similar mobile edge scenario (as shown in Fig. (2.8)), both algorithms consider the service latency as the main driving factor and then propose a trade-off model to solve the problem. According to the results provided by [67], FACT can achieve a low latency with an acceptable frame accuracy. However, as they point out, diverse workloads with heterogeneous servers lead to the problem that the minimization of latency may not

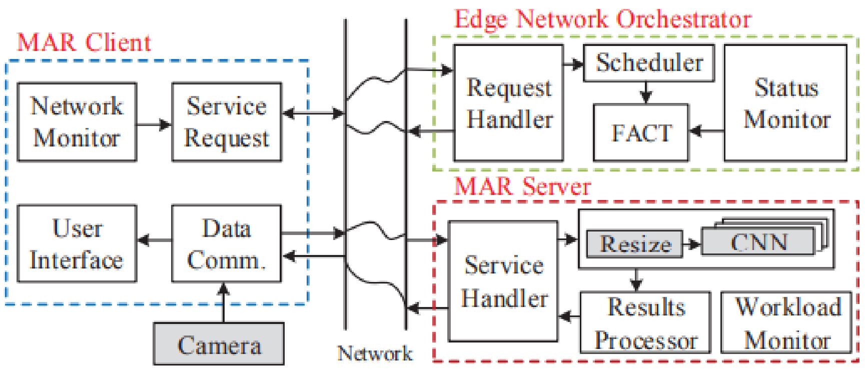


Fig. 2.9 Fast and Accurate Object Analysis (FACT) MAR system overview [67]

happen for network latency and computational latency simultaneously. In that sense, not all available EC can be made a full use in that scheme. They provide an edge network orchestrator for EC-based MAR systems that considers the trade-off between service latency and analytic accuracy [67]. This is similar to the proposed optimization trade-off balancing model for EC-based MAR systems, however there are some significant differences. The most important one is the inherent user mobility. In [67], user's mobility is not taken into account which means that the model considers only the path between user's initial location to assigned edge cloud. However, in this work, mobility is explicitly taken into account which relates to the path between candidate user locations and different edge clouds. Furthermore, another key difference is that in the so-called FACT scheme proposed in [67], the MAR service is regarded as a whole while in this work it is decomposed into two types of functions, i.e., computational and memory intensive. This allows us to have a more flexible assignment, compared to other previous monolithic solutions, according to cache size and CPU frequency. In addition, the work in [67] considers a convex type of function between frame length and computational complexity which is only affected by the frame length. However, this work takes into account differences between EC CPU frequencies, i.e., considers heterogeneous edge clouds. Finally, the authors in [67] considered three main constraints: minimum accuracy requirement, once service per request and range of the decision variable. However, in this paper a more realistic scenario is considered where additional constraints such the available capacity of the edge clouds, the cache size and the cache miss penalty are explicitly taken into account. A more detailed comparison in results between the previously proposed FACT algorithm and the proposed schemes in this paper is presented in numerical investigation section.

In [110], the caching resource sharing problem is considered separately in a Radio Access Network (RAN) and the Core Network (CN) under the view of fully 5G virtualization. Similar to our work, a probability of requesting a set of popular content is assumed, and this information is used to formulate the actual cache hit ratio. More specifically,

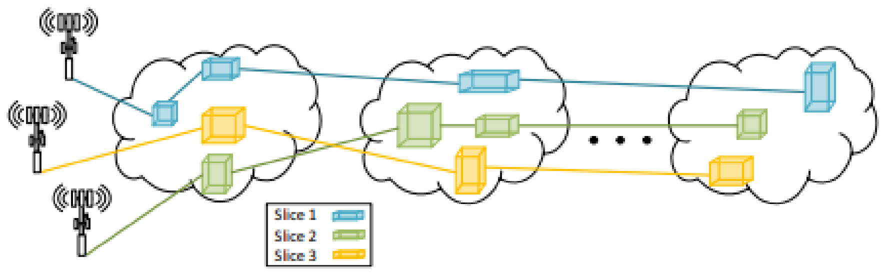


Fig. 2.10 Centralized System model for VNF placement and resource allocation on Network Slices [35]

in [110], a dynamic Cournot game in RAN is firstly solved, and this provides a probability sorting algorithm to achieve the local optima for the CN. Then, a global equilibrium algorithm is proposed to achieve the balance between the given local optimal solutions [110]. Although the demand and competition in mobile virtual network operators are carefully researched in [110], the authors in [35] point out that the chaining of virtual network functions is also a key feature of 5G and beyond virtual networks and should be considered. In Fig. (2.10), a set of network slices is applied to generate a centralized optimization problem that maximizes system utility [35]. A trade-off between efficiency and fairness could be achieved through selecting the utility function for their optimization problem [35]. In [34], another trade-off targeting on workload and service latency is provided. In their later work, two trade-offs are designed for the MAR system, where instead of choosing different target functions under different conditions, a weighting parameter is brought in to directly achieve the balance between latency and preference. Such an idea is also utilized in this paper. However, it has to be noted that most of the previously proposed schemes for MAR do not explicitly take into account wireless transmission, user mobility and service decomposition jointly.

There are other relevant works proposing improved frameworks and networking with the edge cloud support. Fig. (2.11) shows the architecture of a framework based on decomposed functions [117]. In the application layer, the Recognition Proxy serves as a bridge that links both layers. With the support of the Visual Tracker and the Annotation Renderer, the application can track the object's position in continuous camera frames and augments objects based on virtual content sent from the Recognition Proxy. In the Base Layer, a list of decomposed MAR functions handles frames uploaded from the application layer. The Scene Matcher ensures a suitable match between AR objects and local resources with the support from a Cloud Manager. The Motion Detector monitors the status of recognized objects to determine whether they satisfy the current camera frames. The Peer Manager together with the Annotation Synchronizer enable the simultaneous sharing and interactions between different devices. As shown by arrows in Fig. (2.11), the flow

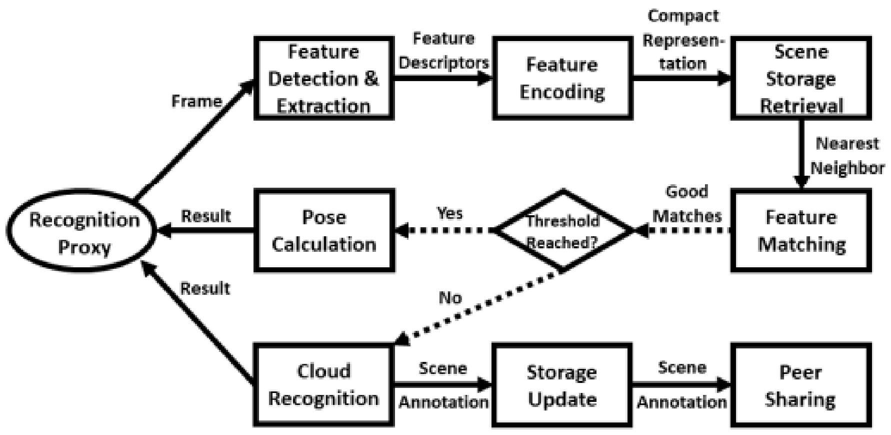


Fig. 2.11 Collaborative Augmented Reality for Socialization (CARS) Framework architecture [117]

has a pre-defined order among main microservices while some other modules provide functionalities to satisfy extra requirements like detecting required poses and enabling socializing. In [104], a tensor-based cloud-edge computing framework is designed for cyber–physical–social systems, which can be separated to the cloud and edge plane. In their framework, the cloud manages large-scale, long-term and global data, while the edge servers react to real-time situations by processing small-scale, short-term and local data [104]. The proposed framework in this paper also tends to allocate decomposed services at edge servers while it turns to the further cloud when exceeding the caching or computing boundary. Another optimization framework in [106], alternating direction method of multipliers (ADMM), considers a general task and regards terminals, edge nodes and remote clouds as a three-tier network. Similar to our framework, this one also jointly optimizes offloading decisions and computing resource allocation to minimize the task delay [106]. It then reduces problem complexity by decomposing the large-scale problem into smaller sub-problems and finally provides near-optimal solutions subjecting to the battery capacity [106]. The deep learning framework in [82] is specialized for AR applications and considers trade-offs between accuracy and latency. However, they accept a more powerful local "helper" (e.g., home servers), and hence, the AR application is processed either locally or in a remote cloud. MAR with Viable Energy and Latency (MARVEL) also accepts a simplified network structure including mobile AR devices and clouds but integrates local inertial tracking, local optical flow and visual tracking into a complete system [16].

There are relevant research works focusing on resource allocation problems for different video view streams. In [11], multiple views of a live video in a cloud-based system are considered in a similar fashion with this work. Similarly, they also try to optimize the overall users' satisfaction by formulating a Mixed Integer Linear Problem (MILP) with computational and bandwidth constraints. In addition, the paper proposes a Fairness-Based Representation Selection (FBRS) heuristic algorithm to achieve near-optimal results [11]. However, the focus in that work is mostly on the application layer quality of experience and received utility without emphasis on the wireless transmission process [11]. In [120], an interactive multi-view HTTP adaptive streaming is proposed with the emphasis in exploring the optimal subset of views under channel constraints. Due to the NP-hardness of the corresponding optimization problem, the paper offers the design of a sub-optimal low-complexity greedy algorithm with focus on network congestion episodes in hot spot areas. The work in [18] anchors its contribution in an optimization framework for the allocation of power and bandwidth in multi-user video streaming to improve the quality of user experience (QoE). Their work tracks requests in the unmanned aerial vehicle (UAV) relay networks and proposes a decentralized allocation scheme to derive the optimal solution [18]. The work in [121] shares a similar target with the current work. More specifically, in order to solve the pro-caching problem for vehicular multi-view 3D video streaming, the authors explicitly consider the mobility and construct a user mobility model that includes predicted moving direction and destinations. The difference is that they treat it as a Markov decision problem, and such a joint optimization problem concentrates on the overall reward and cost in 5G small-cell networks [121]. Due to the complexity of finding optimal decisions, a dynamic K-Nearest Neighbor algorithm is embedded into their proposed deep learning approach to provide the allocation for cache memory and the selection for the views set [121]. In [24], an object detection model of AR applications in an EC-supported network is considered, and the authors provide effective task placement and offloading strategies achieving the balance between latency and energy. Furthermore, the work in [24] directly identifies three latency classes and achieves the task execution time through historical data, while in this paper, a more detailed communication process is focused and calculated from the given formula.

2.5 Utilized Heuristic Algorithms

As already eluded in the previous section, tackling network complexity issues for provisioning granular advanced services under microservices architectures raises several challenges. Leveraging heuristic algorithms on the MAR system after decomposition and chaining, such as Simulated Annealing, Neural Networks and Reinforcement Learning, are considered to have the potential to solve problems in this area. Till now, a series of heuristic algorithms including machine learning ones are proposed with improved prediction capabilities, flow path management and resource allocation features [10]. Compared to optimized algorithms, they manage to achieve acceptable solutions within an acceptable

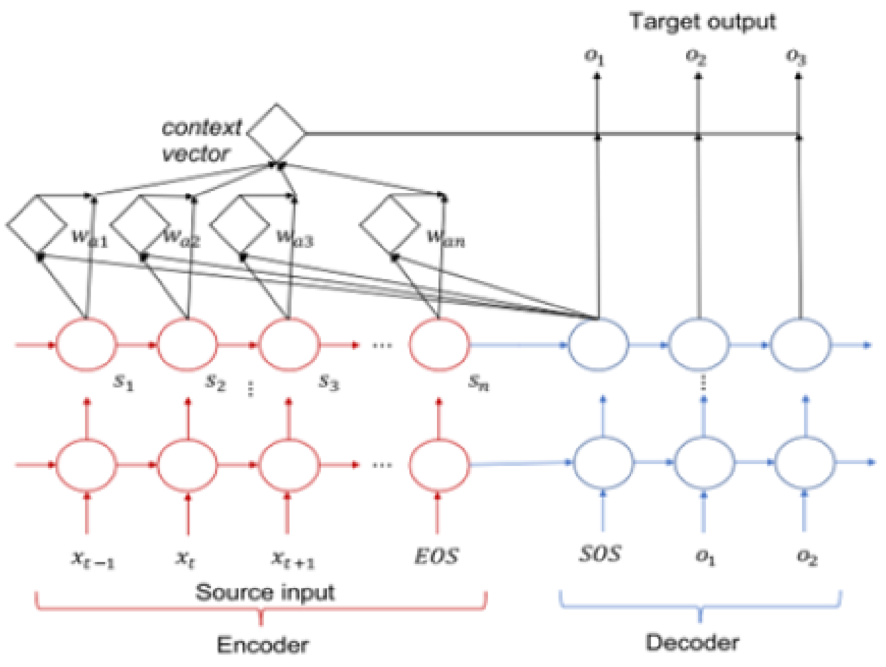


Fig. 2.12 Schematic structural view of the sequence-to-sequence model [125]

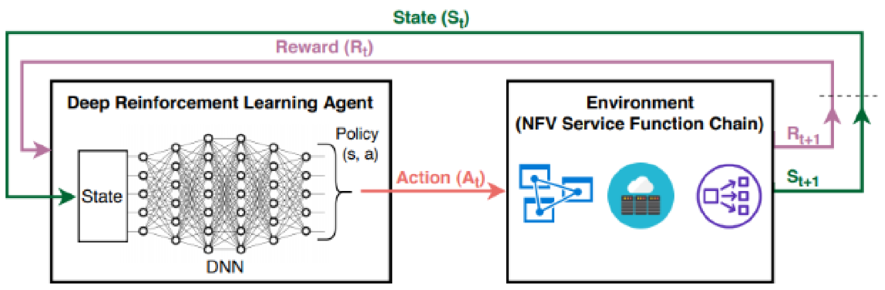


Fig. 2.13 Deep Reinforcement Learning Architecture [53]

space and time cost. Direct integration of heuristic algorithms on container or controller does simplify the overall structure and hence decreases the cost of communication and deployment. A series of machine learning techniques like examples from (2.12) and [53] are utilized as isolated modules to predict network states and provides suggestions for the decision-making stage. According to Fig. (2.12), network paths are treated as sequential data to construct a sequence-to-sequence model, in which source input and target paths are treated as a sequence of nodes [125]. An attention mechanism oversees the data collection to compute a relevant path from input sequence to target sequence based on empirical records (summarized as context vector). The proposed beam search can avoid the sinking in a local optimal solution and further explore global optimal solutions. With the support of deep learning techniques for traffic engineering purposes (neural networks and Natural Language Processing sentences analysis), the proposed technique manages to achieve an efficient network-level path. Furthermore, numerical simulations reveal that such data-driven path planning techniques shows advantages in delay and throughput in both congested and non-congested conditions. Thus, predictions can be generally accurate and useful for network operation with reliable empirical data [125]. The other example is shown by Fig. (2.13) [53]. A model applying Deep Reinforcement Learning (Deep Neural Network, DNN) can also provide resource predictions in dynamic NFV environments. Multiple agents spread over the network and provide their asynchronous observations of the network to reinforce their learned policy and hence can achieve optimized prediction output. According to the design shown in Figure 13, DNN algorithms in the Deep Reinforcement Learning (DRL) agent work as a non-linear approximators that provides an automated dynamic scaling solution for resource allocation purposes and has advantages in dealing with large numbers of outcomes and impacts over time. Based on these predictions, their model then moves to an optimum learned stage that optimizes provisioned resources and takes scaling action on the actual 5G infrastructure [53]. Clearly, through applying an isolated module when processing algorithms like in our works, such modules do not directly interfere with the deployment, allocation and configuration of NFVs, which is computationally friendly and enhances the overall independency and flexibility. However, because these algorithms generate solutions based on records and work separately from the SFC, they are not able to validate their solutions immediately; in addition to that, this might cause potential risks like path non-connectivity and low generalization accuracy [125][53]. Extra prevention or re-computation mechanism are applied to increase stability and fault tolerance. Hence, in our works, the predictions given from the machine learning module are further validated by the Feasibility Check stage before putting into practice.

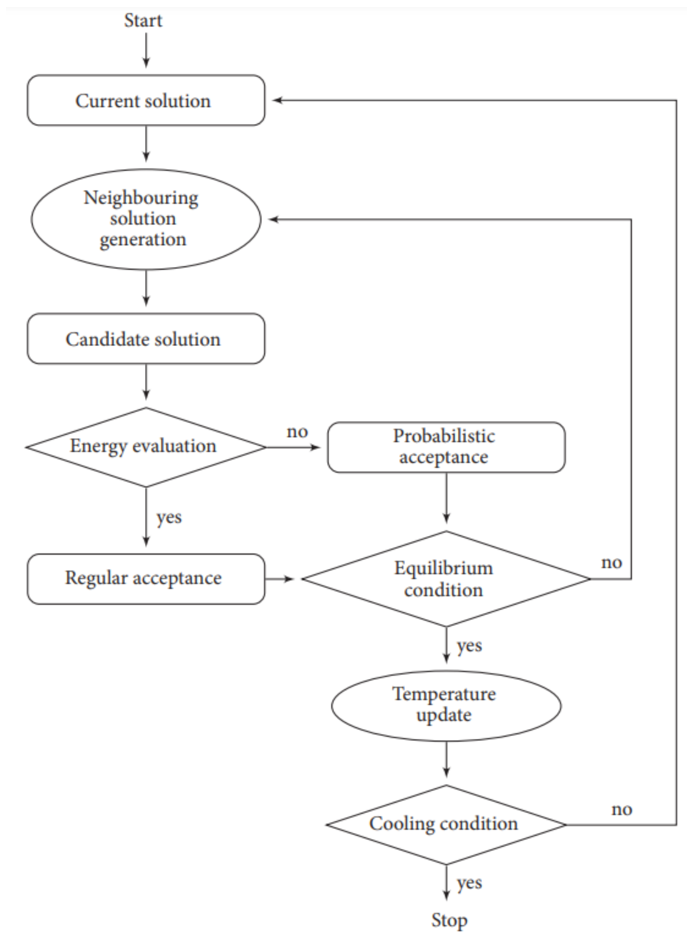


Fig. 2.14 Example flow chart of Simulated Annealing Algorithm [4]

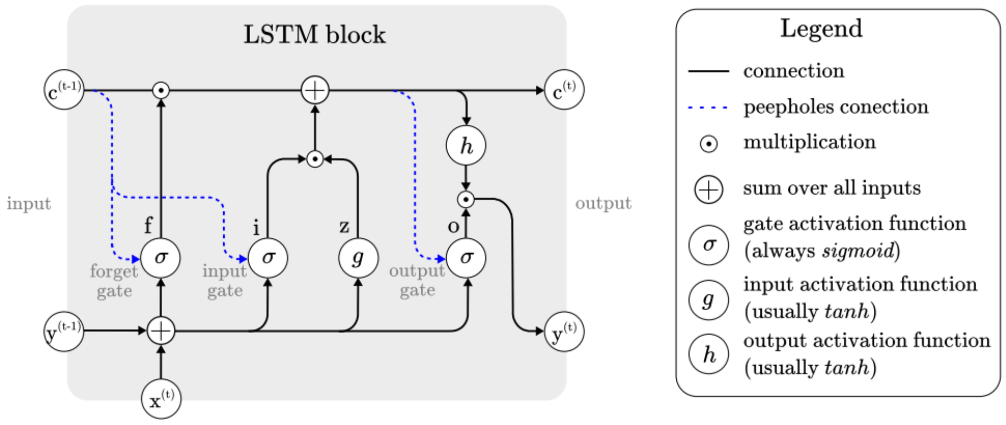


Fig. 2.15 Example block architecture of Long Short Term Memory [103]

In our works, two typical heuristic algorithms, Simulated Annealing (SA) and Long Short Term Memory (LSTM), are utilized to improve overall computing efficiency. Fig. (2.14) shows a typical example flow chart of Simulated Annealing Algorithm. It is regarded as an extension of hill climbing algorithm and improves a certain energetic objective function during each iteration until approaching the global optimum in an acceptable level [4]. It consists of a outer loop representing the cooling scheme and a inner loop representing the solution exploring scheme. SA algorithm brings in an occasional acceptance mechanism that accepts worse solution at a decreasing range by a decreasing probability to avoid sinking into the local optima. Fig. (2.15) shows a typical example block architecture of Long Short Term Memory Algorithm. LSTM consists of a series of memory blocks and each block is defined as a input gate, an output gate and a forget gate [103]. Key features of information are maintained over time while inferences are filtered and forgotten in a short period.

2.6 Technical Gaps

In this chapter, we go through related works about MAR in 5G/B5G networks and a series of key technical gaps in this topic can be concluded as below.

In this topic, at the network level, the collocation between terminals, edge nodes, far-away clouds remains as a big challenge including its planning, optimization and implementation. When moving to the application level, serving as a whole adds extra burden to terminals while the suitable application-specific decomposition granularity and allocation strategy usually appear as non-polynomial problems and are hard to solve. Although some references provide AI supported algorithms but their feasibility, reliability and general quality might still be not satisfying. This thesis selects MAR as an example to explore ideal network optimization. When turn into this area, few works explicitly handle the user physical mobility with a series of realistic constraints. The balancing and Object embedding of AR-specific applications also have not been fully discussed. Metaverse integrates VR and AR into mixed reality while how it interacts with internal objects, virtual scenes and users remains as a challenge.

To target with aforementioned technical gaps in this area, in following chapters, we start with explicit consideration of user physical mobility in a nominal MAR scenario in Chapter 3. In Chapter 4, we further consider the 2D field of view substitution to cut down caching burden. The ARO embedding, sharing and synchronizing issue is considered through view streams in Chapter 5. And finally, we extend MAR into metaverse managing to embed AROs into background scenes and sychronize all updates to users in the same region.

Chapter 3

Optimal Service Decomposition for Mobile Augmented Reality with Edge Cloud Support

The chapter is based on [42] and [48]. In this chapter, we contribute in the MAR service decomposition, categorization and explicitly consideration of user physical mobility with EC support. Heuristic algorithms are explored to identify their potential in elevating computing efficiency.

3.1 Introduction

As mentioned earlier, the utilization of mobile augmented reality (MAR) applications is expected to further increase by the proliferation of enhanced mobile devices and enhanced capabilities offered by the current deployment of 5G networks. Whilst it is already widely accepted in different industries (entertainment, advertisement and manufacturing) and on different devices (i.e. smart glasses) it is well expected that the next frontier of AR applications would be on mobile networks [14]. To enable lighter, highly mobile and more technical advanced AR technologies and devices,we are facing a series of challenges including platform diversity, computational efficiency and caching limitations [80][62]. According to [62], a significant volume of memory and CPU resources size is required when rendering 3D objects which naturally emphasizes the insufficiency of computing and caching resource when depending solely on the mobile terminal capabilities. Noticing that MAR service not necessarily updates each frame, hereafter we assume a nominal frame rate as 15 frames/second and the rendering happens at every other frame ($\tilde{1}$33.2ms interval) [24][75][72]. Thus, the service delay of the aforementioned work flow within the interval could be regarded as acceptable.

Mobile edge cloud (MEC) architectures introduced earlier can better support MAR applications because it is possible to anchor computational and memory intensive functions of the service at computing-efficient nodes at the edge of the network in close proximity to mobile devices [80][62][67]. Despite the gains that can be attained by offloading such complex tasks to edge clouds (ECs) attention should be placed on the required volume of data transmission that need to take place between end devices and the edge cloud. In addition, during congestion episodes parts of the network might be overloaded resulting in communication delays between the end terminal and edge clouds which might affect the service quality [81]. Therefore, MAR system's performance in response time can be increased through enhancing transmission efficiency and optimizing the edge cloud servers allocation and utilization [62][67][81][83]. For example, during congestion episodes reducing the uploaded video frame size might ease the burden of data transmission and computation supported by the MEC. Hence, frame selection and target area/object selection could be used to reduce the amount of total uploaded data [67][50]. However, the analytic accuracy of frames deteriorates with a smaller uploaded size. To this end, it is necessary to achieve, inter alia, a balance between frame accuracy and service latency.

In the sequel, the different functions in which a typical MAR application can be decomposed are explained. It is worth pointing out from the outset that the MAR functions can serve as a network function virtualization (NFV) function chain. NFV service chaining is a well known network architecture paradigm and focuses on decomposed functions' placement, load balancing and availability [38][58]. It enables flexible and economical implementation of applications in 5G networks and beyond [99][58]. In this work, the decomposed MAR functions are categorized into two types: computational intensive functions that consumes mainly computing resources and storage intensive functions that requires significant amount of local cache memory. Firstly, the terminal is responsible for selecting the video frames that have the potential to be augmented with AR objects (ARO). Then, after a pre-processing those video frames are transmitted to an edge cloud for Object Detection and Feature Extraction. Subsequently, the original image of objects can be recognized from extracted features and according to auxiliary information from the uploaded frames (i.e. point of view), the Object Matching function will provide validation and selection of proper AROs in required gesture [117][109]. Different MAR system architectures enable different modules to execute these functions with other supporting ones in different layers (3.1) [81][83][117]. However, the general working process of the decomposed functions in a MAR service can be, in general, summarized into an overall decomposed flow of functions as presented in Figure (3.1).

It is worth pointing out that it is not efficient to deploy all the above mentioned functions at the mobile device due to their limited computational resources [6][88][86]. The above MAR functions can be further decomposed into computational and storage intensive ones,

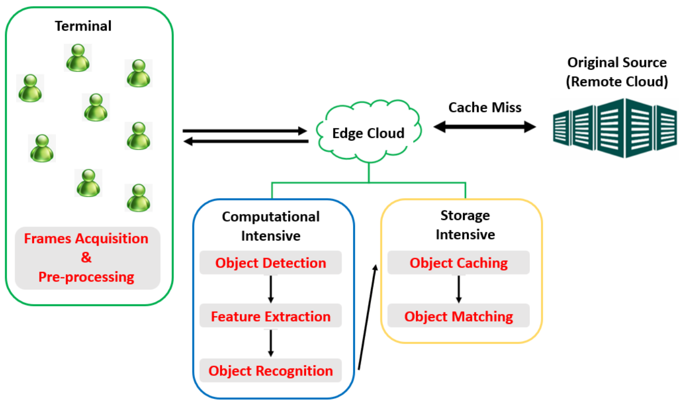


Fig. 3.1 An example of EC-supported working flow of MAR applications including 2 types of MAR functions.

in essence related to those that consume significant CPU resources with the database for ARO matching and those that consume significant cache memory resources. Such decomposition of the underlying MAR functions leads to a more flexible arrangement when some servers have sufficient computing resources with low availability on cache resources or vice versa. Optimization of these functions was the focus by some previous research to efficiently transmit AR tasks to the edge cloud but the mobility effect has not been explicitly taken into account as well as the above classification of the different AR functions [67][109] [66].

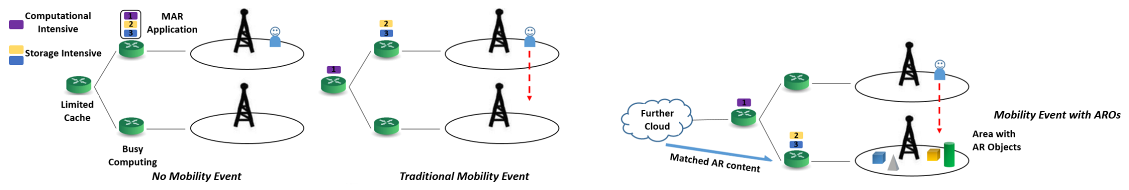


Fig. 3.2 Illustrative toy examples on the effect of mobility on pro active resource allocation of MAR functions. Differences between applying a complete MAR application and its decomposed functions in a no mobility event, a nominal mobility event and a mobility event with AROs

As an illustration, several examples of an AR application applying service decomposition supported by ECs is presented by Figure (3.2). It reveals the mobility effect on the communication between the terminal and the target EC. According to this figure, there could be a detrimental effect in terms of the overall latency when changing the point of attachment during the lifetime of a MAR session. The reason is that the new point of attachment is not an optimized choice in terms of the EC support for such a latency sensitive application. Compared to other applications, which might be tolerant to delay, the mobility effect in MAR services should be carefully considered. When facing both AROs and the user mobility, decomposition and proactive resource allocation can be combined to achieve an optimized decision making.

Motivated by the above aspects, the key novelty is that in this chapter we explicitly consider the user mobility and formulate a multi-objective optimization problem aiming to balance the overall delay, video frame quality and caching of augmented reality objects in the network faced by the MAR application that requires offloading task to an EC. To the best our our knowledge no previous research work considered explicitly the user mobility and service decomposition in order to optimize the service delivery and allocation of network resources. We extend hereafter our previous work [42] to construct the proposed optimal framework (via mathematical programming) including the development of a simulated annealing based scheme and a long short term memory (LSTM) neural network based scheme to provide competitive solutions, whilst being amenable to real-time decision making. Through a wide set of simulations, the performance of the above proposed schemes are compared with several baseline schemes stemming from closely related research. Advantages of the proposed schemes are discussed in analysis and evaluation section.

3.2 Related work

The proposed scheme in this chapter provides, in essence, an EC-supported MAR system whereas MAR service decomposition and user mobility is explicitly taken into account whilst balancing augmented reality objects caching, video frame length and accuracy. In that frontier, there has been various recent research efforts to improve overall MAR systems. The work in [34] proposes a trade-off model between workload and service latency. It selects from candidate locations to place ECs and then focus on the users' allocation issue. While in this chapter, our proposed scheme could find optimal solutions for both proactively caching and allocation decisions under pre-given network conditions. The research in [66] proposes a dynamic adaptive protocol over the edge for an MAR system. It also only focuses on the resource allocation and enables the AR reconfiguration by users to adapt to variations in wireless channels and computation workload. However, our proposed scheme reacts to changes through finding another suitable EC instead of changing configuration and causes potential interference to other requests. In a similar setting, the work in [50] considers a cloud-based architecture to improve the performance of MAR applications in development, deployment and maintenance. The above mentioned work considered the MAR system as a whole, however some other relevant works considered specific aspect or a particular functionality of a MAR application. The work in [64] improves object detection, while the work in [52] improves object recognition. On the other hand, Wu et al.

focus on finding a better local cache management strategy [109]. Furthermore, Li et al. reduce the amount of initially loaded model data to minimize computational pressure [62]. Among existing works, mobility of AR applications is mentioned but its effect on service latency is not considered explicitly as is the case in this work.

In a closely related work, Liu et al. focus on the relationship between latency and frame resolution in MAR systems and edge clouds [67]. They provide an edge network orchestrator for EC-based MAR systems that considers the trade-off between service latency and analytic accuracy [67]. This is similar to the proposed optimization trade-off balancing model for EC-based MAR systems, however there are some significant differences. The most important one is the inherent user mobility. In [67], user's mobility is not taken into account which means that the model considers only the path between user's initial location to assigned edge cloud. However, in this work, mobility is explicitly taken into account which relates to the path between candidate user locations and different edge clouds. Furthermore, another key difference is that in the so-called FACT scheme proposed in [67], the MAR service is regarded as a whole while in this work it is decomposed into two types of functions, i.e., computational and memory intensive. This allows us to have a more flexible assignment, compared to other previous monolithic solutions, according to cache size and CPU frequency. In addition, the work in [67] considers a convex type of function between frame length and computational complexity which is only affected by the frame length. However, this work takes into account differences between EC CPU frequencies, i.e., considers heterogeneous edge clouds. Finally, the authors in [67] considered three main constraints: minimum accuracy requirement, once service per request and range of the decision variable. However, in this chapter a more realistic scenario is considered where additional constraints such the available capacity of the edge clouds, the cache size and the cache miss penalty are explicitly taken into account. In [91], authors also consider the cache hit/miss at ECs similarly to this work. However, the emphasis is into the caching and power consumption of the mobile AR devices and the associated mobile cache management. Hence, a different trade-off model has been considered with focus on balancing between energy consumption of the MAR device and latency in terms of caching size. In addition, service decomposition for the MAR application is applied in [87]. Their proposed framework also achieves efficient computing and offloading through dynamically reconfiguration of video and server. They also take processing and transmission delay into account similarly to this work. However, when considering the quality, they consider the user expected latency and user perceived video quality. Thus, we consider the FACT algorithm in [67] as a typical example from these related works to compare our proposed framework. A more detailed comparison in results between the previously proposed FACT algorithm and the proposed schemes in this chapter is presented in numerical investigation section.

3.3 System Model

3.3.1 Preliminaries

An undirected graph $\mathcal{G} = \{\mathbb{V}, \mathbb{L}\}$ is used to represent the mobile network, where $\mathbb{V}$ expresses the set of vertices (i.e., network nodes) and $\mathbb{L}$ denotes the set of links. The available ECs in the network are expressed with a location set which is defined as $\mathbb{M} = \{1,2,...,M\} \subseteq \mathbb{V}$. Requests $r \in \mathbb{R}$ are assumed to be created by a set of MAR devices and their initial connected access router is $f(r)$. Due to mobility events, they can connect to different adjacent access routers k belonging to set $\mathbb{K} = \{1,2,...,K\} \subseteq \mathbb{V}$. Based on available historical data of a mobile network operator, it can be estimated with high accuracy a probability matrix P_{ij} which denotes the total moving probability of a mobility event between access routers $i, j \in \mathbb{K}$ [124]. In this chapter, the model diagram is actually a tree topology as shown by Fig. (3.3).

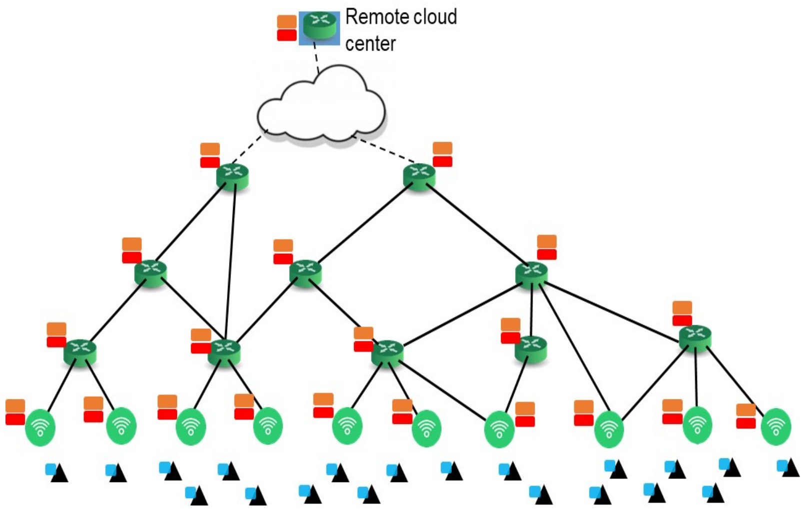


Fig. 3.3 Typical tree-like designed network topology

As already eluded above, two main AR sets of functionalities are required to run (i.e., offloading) on the ECs. These are categorized into ARO caching related functions that require storage space and intense image processing functions requiring significant CPU processing power [117][109]. Hence, these two functionalities are expressed with η and ρ for CPU processing and caching respectively. We denote the set of available AROs as **N**={1,2,...,N}. A subset of those AROs, denoted as L_r, are randomly allocated to each request $r \in \mathbb{R}$. Thus, $l \in \mathbf{L_r} = \{1,2,...,L_r\} \subseteq \mathbf{N}$ denotes each ARO l that is required in the request r and its size is denoted as O_l^r. $F_{\eta r}$ and $F_{\rho r}$ are used to describe the file size for each function of the request r. $F_{\eta r}$ represents the size of frames sent for object detection while $F_{\rho r}$ represents the size of recognized objects used for matching. Denoting the frame length in its set as $s \in \mathbb{S}$, $F_{\eta r}$ is measured by σs^2; we note that the product of frame length

s^2 and σ express the number of bits to represent a single pixel [67]. A nominal frame rate is assumed of 15 frames/second and the service could happen at every other frame ($\tilde{1}$33.2ms interval) [24][75][72]. Thus, the service delay of the aforementioned work flow within the above time interval could be regarded as acceptable.

Hereafter, we look into the scenario where a user requests a certain set of AROs and functions η and ρ run in a pre-defined order at the same or different ECs. Due to the consideration of mobility events, user's potential points of attachment can influence these functions' hosted ECs. If a cache hit happens for all required AROs, the matched ones will then be sent back to the MAR device for displaying [117][109]. Otherwise, an extra latency D for ARO cache miss will be triggered since the AROs will have to be fetched from a remote core network server that host the augmented reality objects [109].

3.3.2 Mathematical Programming Formulation

According to the overall MAR system model and aforementioned preliminaries, the required decision variables are introduced paving the way for the proposed mathematical programming formulation. To this end, the decision variables are defined as follows,

$$x_{rj} = \begin{cases} 1, \text{if computational function for request } r \\ \quad \text{located at node } j, \\ 0, \text{otherwise.} \end{cases} \tag{3.1}$$

$$y_{rj} = \begin{cases} 1, \text{if caching function for request } r \text{ located at} \\ \quad \text{node } j, \\ 0, \text{otherwise.} \end{cases} \tag{3.2}$$

$$h_{rlj} = \begin{cases} 1, \text{if ARO } l \text{ of request } r \text{ cached at} \\ \quad \text{node } j, \\ 0, \text{otherwise.} \end{cases} \tag{3.3}$$

$$a_{rs} = \begin{cases} 1, \text{if request } r \text{ use frame length } s, \\ 0, \text{otherwise.} \end{cases} \tag{3.4}$$

The following decision variables are also defined to capture the case of cache miss or hit,

$$z_{rj} = \begin{cases} 1, \text{if there is a cache hit for AROs of request } r \\ \quad \text{at node } j, \\ 0, \text{otherwise.} \end{cases} \tag{3.5}$$

We describe the relationship between the decision variables h_{rlj} and z_{rj} as follows,

$$z_{rj} = \begin{cases} 1, \text{ if} \sum_{l \in N} h_{rlj} \geqslant L_r, \\ 0, \text{ otherwise.} \end{cases} \tag{3.6}$$

As mentioned above, it is necessary to ensure that if $\sum_{l=1}^{N} h_{rlj} \geqslant L_r$ then $z_{rj} = 1$ in the mathematical programming formulation. It means at the target server, if cached AROs fulfill or even overfill (i.e., caching extra AROs for other requests to be served here) the user's requested AROs in the set L_r, then it is defined as a cache hit for this user. An equivalent either-or constraint can be applied to replace this conditional constraint. More specifically, it can be written as follows,

$$\left(\sum_{l \in N} h_{rlj} < L_r \right) \text{ or } (z_{rj} = 1) \tag{3.7}$$

A small tolerance value ε can be specified to change the first expression to inequality. Thus, the either-or logical constraint can be written as,

$$\left(\sum_{l \in N} h_{rlj} + \varepsilon \leq L_r \right) \text{ or } (z_{rj} = 1) \tag{3.8}$$

A new decision variable q_j is now introduced with a large arbitrary number U to rewrite the above either-or constraint into the following expressions that need to be added in the mathematical programming formulation,

$$\begin{aligned} \sum_{l \in N} h_{rlj} + \varepsilon \leq L_r + U(1 - q_j) \\ z_{rj} = 1 - q_j \end{aligned} \tag{3.9}$$

The overall cost measured in terms of delay can be split into wireless delay, wired network delay and computational delay [67]. The actual transmission time is mainly driven by the uploaded frame size and can be written as follows,

$$\frac{\sigma s^2 a_{rs}}{\widetilde{R}_r} \tag{3.10}$$

An average achievable data rate is assumed to be allocated to a user across the whole session lifetime and we are not posing any constraints and/or requirements on the instantaneous data rate. Hence, without loss of generality, an average rate of $\widetilde{R}_r$ at different connected cells is allocated to each request r based on a defined Service Level Agreement (SLA) [76].

The delay between ECs is noted by C_{ij}, which encapsulate the least communication delay (shortest path routing cost) between locations i and j [124]. Propagation delay is also captured together with other types of delay during this period such as for example nominal queuing delay.

Two types of computing delay at ECs are captured by parameters V_{rj} and W_{rj} and can be estimated as follows [98],

$$V_{rj} = \sum_{s \in S} \frac{\omega F_{\eta rs}}{f_V^j} a_{sr}, \; W_{rj} = \frac{\omega F_{\rho r}}{f_V^j} \tag{3.11}$$

where

$$\sum_{s \in S} a_{sr} = 1 \tag{3.12}$$

In the formula above, the number of CPU cycles to process 1 byte input is represented by ω [98][100][26][65]. The VM computational speed f_V^j is assigned for request r at location j and is assumed, without loss of generality, to be allocated equally [65]. Given the condition that each VM takes up a fixed portion of a CPU core, the product of core frequency and allocated percentage can express its computational speed [68]. Thus, in above formula, the computing delay can be achieved through dividing total consumed CPU cycles (the production of file size and cpu cycle-byte rate) by assigned portion of CPU speed.

An EC at location j is assumed to have an allowable cache capacity Θ_j and an allowable computing capacity Δ_j measured by the number of server's remaining available VMs [65].

Finally, notations including key variables and brief explanations are summarized in table (3.1).

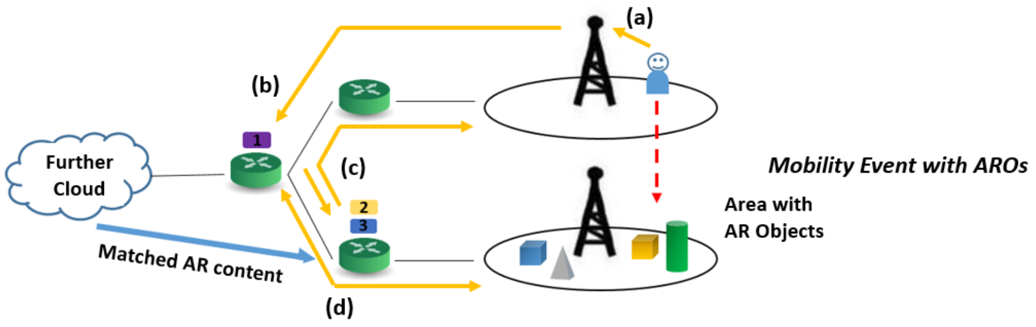


Fig. 3.4 A toy example for Equation (3.13a)

As shown in the formulation below, the overall service latency is captured including transmission, computation and delay caused by cache retrieval and processing. The

Equation (3.13a) consists of 4 main parts in order, which are shown in Fig. (3.4). The first term captures the wireless transmission delay from the user's initial location to the access point as shown by (a) in the figure. While the second term shown in the Figure by (b) captures the link delay to the first target edge cloud server and the processing delay of computational intensive functions at that location. The third term shown by (c) captures the link delay between two edge clouds, the processing delay of the matching function and the incurred delay of returning to the initial access point. Finally, the fourth term shown by (d) captures the link delay from potential destinations. The repeated process like wireless transmission or executing functions are not counted again for destinations. Note that both the so-called before and after mobility events are included in the objective function and it captures the overall service latency. To this end, the mathematical optimization problem can be formulated as follows,

$$\begin{aligned} min \; & \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\frac{F_{\eta rs}a_{rs}}{\widetilde{R}_r}+\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}(C_{f(r)i}+V_{ri})x_{ri}+ \\ & \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}\left(C_{ij}x_{ri}+W_{rj}+C_{jf(r)}+q_jD\right)y_{rj}+ \\ & \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{k\in\mathbf{K}}P_{f(r)k}C_{ik}(x_{ri}+y_{ri}) \end{aligned} \tag{3.13a}$$

s.t. (3.12)

$$\sum_{r\in\mathbf{R}}\sum_{l\in\mathbf{L_r}}h_{rlj}O_l \le \Theta_j, \forall j\in\mathbf{M} \tag{3.13b}$$

$$\sum_{r\in\mathbf{R}}x_{rj}+y_{rj}\le\Delta_j, \forall j\in\mathbf{M} \tag{3.13c}$$

$$\sum_{l\in\mathbf{N}}h_{rlj}+\varepsilon\le L_r+U(1-q_j)\forall j\in\mathbf{M}, r\in\mathbf{R} \tag{3.13d}$$

$$z_{rj}=1-q_j, \forall j\in\mathbf{M}, r\in\mathbf{R} \tag{3.13e}$$

$$\sum_{j\in\mathbf{M}}x_{rj}=1, \forall r\in\mathbf{R} \tag{3.13f}$$

$$\sum_{j\in\mathbf{M}}y_{rj}=1, \forall r\in\mathbf{R} \tag{3.13g}$$

$$\begin{aligned} & a_{rs}, x_{rj}, y_{rj}, h_{rlj}, z_{rj}, q_j \in \{0,1\}, \\ & \forall r\in\mathbf{R}, j\in\mathbf{M}, l\in\mathbf{L_r}, s\in\mathbf{S} \end{aligned} \tag{3.13h}$$

The different costs added to delay are captured by the objective function (3.13a), which embed the routing to the ECs and computational delays for the two functions of AR video

Table 3.1 Notation

Parameter	Description
$\mathbf{N}$	Set of all AROs
$\mathbf{M}$	Set of Edge Clouds
$\mathbf{R}$	Set of User Requests
Θ_j	Cache capacity at EC j
$\mathbf{S}$	Set of possible frame lengths
$f(s)$	get accuracy from frame length
$g(r)$	Initial access router for request r
Δ_j	Number of available VMs at EC j
P_{ij}	Mobility probability from EC i to j
D	Fixed delay cost caused by a cache miss
η, ρ	Two types of AR functions (CPU, Cache)
C_{ij}	Communication delay between nodes i and j
$O_l, \mathbf{L_r}$	Size of object l, and set of objects in request r
σ	the number of bits required to represent 1 pixel
a_{rs}	0/1 var.: if the frame legnth s is selected for r
x_{rj}, y_{rj}	0/1 var.: if function η or ρ for r is set at EC j
ω, f_V^j	Computational load and CPU availability at EC j
V_{rj}, W_{rj}	Processing delay for request r for function η, ρ at EC j
$F_{\eta r}, F_{\rho r}$	Input video frame size for functions η, ρ in request r
h_{rlj}	0/1 var.: if object l appearing in r is cached at EC j
z_{rj}	0/1 var.: if there is a cache hit of all ARO for r at EC j

processing and ARO caching. Cache capacity limit is captured by constraint (3.13b), while the limit on the number of utilized VMs is captured by constraint (3.13c). As explained earlier, we initially have the case that if AROs required by a user are all pre-cached, then it is a cache hit. This is equivalent to an either-or Constraint (3.7). Through adding a tolerance value to be Constraint (3.8) and then bringing in the decision variable q_j, the both cases could be merged into an inequality expression, Constraint (3.13d), which describes the cache hit/miss requirement. Constraint (3.13e) help bind decision variables z_{rj} and q_j. Finally, constraints (3.13f) and (3.13g) capture the initialization requirement for MAR functions which means they should run only once at a specific EC location for each request.

Note that the product of decision variables make the previous defined mathematical program (3.13a) non-linear. However, it is possible to linearize the problem by introducing the following auxiliary decision variables,

$$\sum_{r \in \mathbf{R}} \sum_{i \in \mathbf{M}} \sum_{j \in \mathbf{M}} \xi_{rij} = \sum_{r \in \mathbf{R}} \sum_{i \in \mathbf{M}} \sum_{j \in \mathbf{M}} x_{ri} y_{rj} \tag{3.14}$$

$$\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}\psi_{rj}=\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}q_jy_{rj} \tag{3.15}$$

$$\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\sum_{j\in\mathbf{M}}\phi_{rsj}=\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\sum_{j\in\mathbf{M}}a_{rs}x_{rj} \tag{3.16}$$

In addition, the binding constraints added for these new decision variables are as follows,

$$\xi_{rij}\le x_{ri},\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{3.17}$$

$$\xi_{rij}\le y_{rj},\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{3.18}$$

$$\xi_{rij}\ge x_{ri}+y_{rj}-1,\ \forall r\in\mathbf{R},i,j\in\mathbf{M} \tag{3.19}$$

$$\psi_{rj}\le q_j,\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{3.20}$$

$$\psi_{rj}\le y_{rj},\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{3.21}$$

$$\psi_{rj}\ge q_j+y_{rj}-1,\ \forall r\in\mathbf{R},j\in\mathbf{M} \tag{3.22}$$

$$\phi_{rsj}\le a_{rs},\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{3.23}$$

$$\phi_{rsj}\le x_{rj},\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{3.24}$$

$$\phi_{rsi}\ge a_{rs}+x_{rj}-1,\ \forall r\in\mathbf{R},s\in\mathbf{S},j\in\mathbf{M} \tag{3.25}$$

Finally, the previous defined expression of delay can be rewritten into its linearized version as follows,

$$\begin{aligned} min\ &\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S}}\frac{F\eta_{rs}a_{rs}}{\widetilde{R}_r}+\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}(C_{f(r)i}x_{ri}+\frac{F\eta_{rs}\phi_{rsi}}{f_V^i})+\\ &\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M}}\big(C_{ij}\xi_{rij}+(W_{rj}+C_{jf(r)})y_{rj}+\psi_{rj}D\big)+\\ &\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{k\in\mathbf{K}}P_{f(r)k}C_{ik}(x_{ri}+y_{ri}) \end{aligned} \tag{3.26a}$$

s.t. (3.12)

$$\sum_{r\in\mathbf{R}}\sum_{l\in\mathbf{L_r}}h_{rlj}O_l\le\Theta_j,\forall j\in\mathbf{M} \tag{3.26b}$$

$$\sum_{r\in\mathbf{R}}(x_{rj}+y_{rj})\le\Delta_j,\forall j\in\mathbf{M} \tag{3.26c}$$

$$\sum_{l\in\mathbf{N}}h_{rlj}+\varepsilon\le L_r+U(1-q_j)\,\forall j\in\mathbf{M},r\in\mathbf{R} \tag{3.26d}$$

$$z_{rj}=1-q_j,\ \forall j\in\mathbf{M},r\in\mathbf{R} \tag{3.26e}$$

$$\sum_{j\in\mathbf{M}}x_{rj}=1,\forall r\in\mathbf{R} \tag{3.26f}$$

$$\sum_{j\in\mathbf{M}} y_{rj} = 1, \forall r \in \mathbf{R} \tag{3.26g}$$

$$(21) - (29) \tag{3.26h}$$

$$a_{rs}, x_{rj}, y_{rj}, h_{rlj}, z_{rj}, q_j \in \{0,1\},$$
$$\forall r \in \mathbf{R}, j \in \mathbf{M}, l \in \mathbf{L_r}, s \in \mathbf{S} \tag{3.26i}$$

$$\psi_{rj}, \xi_{rij}, \phi_{rsj} \in \{0,1\}, \forall r \in \mathbf{R}, i, j \in \mathbf{M}, s \in \mathbf{S} \tag{3.26j}$$

The relation between uploaded frame size and its accuracy can be given below by an approximation function in [67],

$$f(s) = 1 - 1.578e^{-6.5\times10^{-3}s} \tag{3.27}$$

Note that the RMSE (root mean square error) of function $f(s)$ is less than 0.03 [67]. Thus, the product $f(s)a_{rs}$ can transfer frame lengths into discrete values with high accuracy.

Finally, the frame selection, EC allocation and route decision problem in the MAR system can be transferred into an optimization problem,

$$\begin{aligned} min\ F &= \mu\frac{T}{T_{max}} - (1-\mu)\sum_{r\in R}\sum_{s\in S}\frac{f(s)a_{rs}}{f(s)_{max}} \\ &= \mu\frac{T}{T_{max}} - \frac{(1-\mu)}{|R|}\sum_{r\in R}\sum_{s\in S} f(s)a_{rs} \end{aligned} \tag{3.28a}$$

$$\text{s.t. } (3.12), (3.26b) - (3.26j) \tag{3.28b}$$

Where $\mu \in [0,1]$ is the weight parameter, T is the objective function shown in (3.26a) and $f(s)_{max}$ is assumed to be 1. The above described integer linear optimization problem due to the combinatorial nature can not be solved efficiently for large network instances. State of art branch and bound techniques can help finding the optimal solution for low to medium size network instances. Hence, metaheuristic algorithms are needed to provide competitive solutions whilst being amenable for real-time implementation.

3.3.3 A Simulated Annealing Based Heuristic

Although the proposed scheme can provide an optimal decision making, suffers from the curse of dimensionality due to the combinatorial nature of the mixed integer linear program (MILP). To this end, a Simulated Annealing framework is developed that evolves from a basic greedy solution and avoids sinking into local optima (please refer to [108][30] and references therein regarding the specifics of the simulated annealing framework). In this chapter, a Simulated Annealing based algorithm firstly starts with a greedy scheme that selects an EC which is close in distance (Euclidean) to the user and then continuously switching to neighbor solutions to identify ones that increase the quality of the objective function. The typical cooling process of Simulated Annealing allows a quick decision making stage and accepts worse case sometimes in a smaller and smaller range during iterations [108][30]. The pseudo-code of the proposed SA-based mobility aware AR algorithm (SAMAR) is shown in (1). Temperature T can be calculated from $T = 0.93^k \frac{g}{200}$, where g is the frame length and k is the current number of iterations ($k \in [1, 500]$). Normalizing the gap between neighbor solutions over the temperature through $\exp^{-\frac{f(k)-f(k-1)}{T}}$ into (0,1), we get the probability on accepting the solution $f(k)$ even if it's worse ($f(k) \geq f(k-1)$). Noticing that the temperature goes down with the number of iterations, the probability becomes smaller as well. Thus, the SAMAR schemes also accepts the worse solution in a decreasing probability. From (1), the first loop generates a basic greedy solution based on the distance and CPU frequency. Then multiple iterations are applied to explore its neighbor solutions to find a solution of increased quality. The neighbor solution means that compared with the original solution, only one request's assignment or frame length choice is changed randomly to another feasible one. SAMAR accepts a better solution and an inferior one within a probability controlled by the temperature parameter. Its time complexity is $O(n^2)$ since the most complex nested loop shown by the pseudo code is in 2 layers, which takes n^2 times to finish at the worst case (e.g. going through each request and then each available frame length to achieve results from formula 3.28a).

3.3.4 A Long Short Term Memory (LSTM) Neural Network Based Scheme

To increase the efficiency and reliability in responding to the changes in the network conditions, a LSTM neural network based scheme with its time complexity as $O(n^2)$ is developed in this section. It is a well known and used machine learning technique that avoids gradient vanishing and gradient exploding through keeping the important information and key features in the long term while forgetting unnecessary inferences [40][125]. Hence, in this chapter, it is utilized to learn from optimal solutions and provide high quality decision making during the inference phase in an efficient manner. Regarding the service requests the possible destinations in terms of selecting edge cloud support are limited to the adjacent servers compared to the current location of a user and is denoted as $d_r \in \mathbb{K}$ for each request. According to previous notations, routing paths per request are pre-defined and denoted as $RT_r = [f(r), x_{ri}, y_{rj}, d_r]$. The set of this matrix $\mathbb{RT}$ is provided by the optimal scheme described in the previous section (denoted as Optim in the sequel). During training, initial locations and destinations of requests can be fed as input and decisions of where to anchor MAR functionalities can be provided as the output. Thus, the route matrix can be separated into $X_r = [s_r, d_r] \in \mathbb{X}$ and $Y_r = [x_{ri}, y_{rj}] \in \mathbb{Y}$. Most

Algorithm 1 SAMAR

Input: requests **R**, EC list **ECL**, weight μ, distances between servers **C**, frame length set **G**, destinations set **K**

Output: server assignment **A**, chosen frame length CFL

```
1: for $r \in \mathbf{R}$ do
2:     ECL $\leftarrow$ sort servers by its distance to starting
              point $s(r)$ and all possible destinations $\mathbf{K}$;
3:     FL $\leftarrow$ select $n$ closest servers in ECL and sort
              again according to vcpu frequency;
4:     A $\leftarrow$ select $\{i, j\}$ from top servers in FL;
5:     if free VM at $i == 0$ then
6:         ECL $\leftarrow$ remove $i$-th server;
7:     end if
8:     if free VM at $j == 0$ and $i \neq j$ then
9:         ECL $\leftarrow$ remove $j$-th server;
10:    end if
11:    for $g \in \mathbf{G}$ do
12:        $f(g) \leftarrow$ value of formula (3.28a) with
                  assignment A and frame length $G$;
13:    end for
14:    $f(0), CFL \leftarrow$ minimum $f(g)$ and corresponding $g$;
15: end for
16: for $k = 1 \rightarrow BOUND$ do
17:    $\{\mathbf{A}', CFL'\} \leftarrow$ swift to neighbor solution
18:    $f(k) \leftarrow$ value of formula (3.28a) with assignment
              $\mathbf{A}'$ and $CFL'$;
19:    $p \leftarrow$ a random number from 0 to 1;
20:    if $f(k) < f(k-1)$ then
21:        $\{\mathbf{A}, CFL\} \leftarrow \{\mathbf{A}', CFL'\}$
22:    else
23:        if $p < \exp^{-\frac{f(k)-f(k-1)}{T}}$ then
24:            $\{\mathbf{A}, CFL\} \leftarrow \{\mathbf{A}', CFL'\}$
25:        end if
26:    end if
27: end for
28: return $\{\mathbf{A}, CFL\}$
```

routes inside consist a training set while the rest of them are applied for testing. Thus, $\mathbb{XTrain}, \mathbb{XTest} \subset \mathbb{X}$ and $\mathbb{YTrain}, \mathbb{YTest} \subset \mathbb{Y}$ are denoted to differentiate training and testing sets. The predictions made by this scheme are denoted as $\mathbb{Pred}$. Firstly, the Optim scheme is executed to provide the optimal decision making for anchoring MAR functionalities to different ECs. The process from obtaining optimal solutions to providing predictions is presented in Fig. (3.5). $\mathbb{YTrain}$ is adapted to the categorical type to enable the scheme to complete the classification work. Selecting any two different access routers from the set $\mathbb{K}$ give K^2 different types of results. Hence, each assignment is allocated a unique index $Ind_r \subset \{1, ..., M^2\}$ to differentiate its type. During training, the scheme will aim to classify and fathom out the relation between input sets. Then, the predictions are provided and transferred back for Feasibility Check. This stage will check whether the predictions can still fit the original network settings and satisfy all constraints. If an overloaded EC is selected, the request will be sent to a core cloud deeper in the network and as a result this allocation will trigger an extra penalty. The predictions are finally compared with the optimal solutions to evaluate the quality of the LSTM-based scheme.

Fig. (3.6) shows the architecture of the LSTM-based scheme. The LSTM layer follows the nominal design without modifying its state changing and gate controlling formula. Therefore, the potential of nominal LSTM network for service decomposition is explored. A dropout layer follows the LSTM layer to avoid over-fitting by setting a random number of elements to 0 ($rand(size(X)) < P$, X is a layer input and P is a self-defined probability) and scaling other elements by $\frac{1}{1-P}$ [97][60]. Two sets of such layers are applied to improve the overall performance. Then, a fully connected layer multiplies a weight matrix and adds a bias vector to the result in order to combine features and learn the underlying pattern [32][37]. Since the logistic sigmoid function is a smooth curve in (0,1) and good at classify sets without very large differences,it is applied as the output activation function in the softmax layer [12]. Finally, the classification layer calculates the entropy loss at the current time step t, denoted as y_t. This process is repeated until the training phase is concluded and then the resulting LSTM neural network can be used for real-time prediction at the inference stage.

3.4 Numerical investigations

3.4.1 Parameterization

In this section, the proposed schemes are investigated and compared vis-á-vis with baseline schemes to assess their performances through a wide-set of numerical investigations.

Throughout the simulations, the wireless transmission rate from MAR device to the access point under 5G environment is set in the range of 100 to 120 Mbps. Each EC is

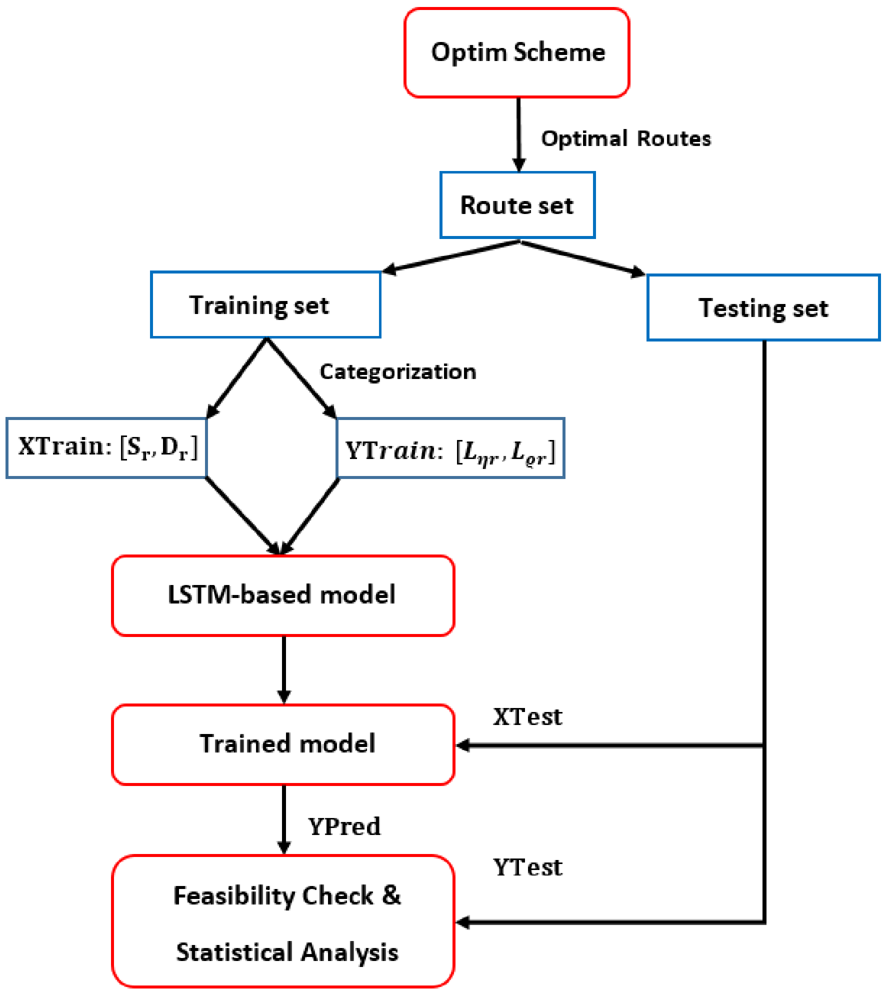


Fig. 3.5 Working process

assumed to have a physical CPU with 8 cores and 16GB memory [124][117]. The CPU frequency is generated randomly for each EC ranging from 2 to 4 GHz [117][68]. Each VM in an EC is assumed to take up the CPU resources equally and hence virtual CPU frequencies between these VMs can also be equally split [68]. In each EC, up to 14 VMs are assumed to be available. In addition, the computation load ω at each EC is assumed as 10 cycles/bit [113]. The communication latency between the terminal and target ECs can be achieved through adding link delays in the shortest path. The average latency for each hop is assumed to be 2ms, including propagation delay and queuing delay [89][9]. When a cache miss takes place, the corresponding penalty is triggered and set as 25ms. A tree-like topology is created as presented in Figure (3.3) and is supported by dense ECs (20 in total). Because MAR applications are not necessary to run on every EC in the network, only a random subset of ECs are assumed to be active in simulations.

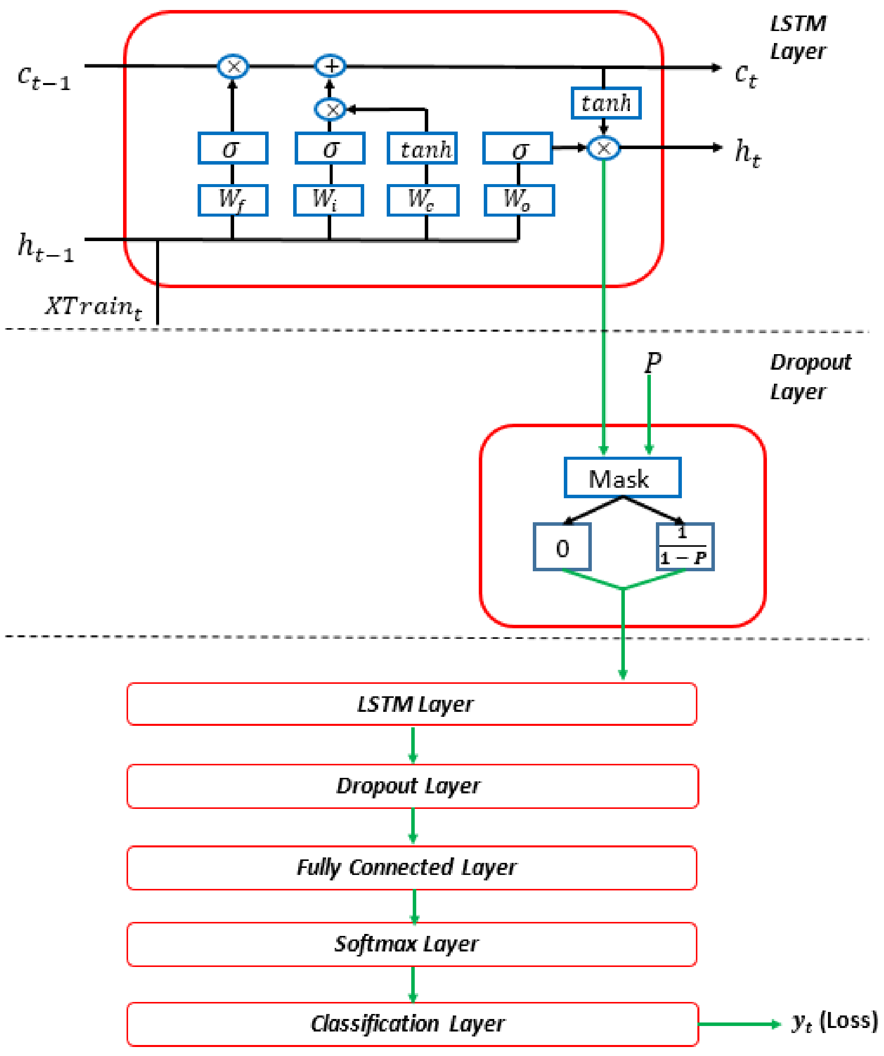


Fig. 3.6 Layered structure of LSTM-based model (at time step t)

The majority of AR applications tend to apply square shaped frames or vision markers due to its advantage in providing multiple co-planar corresponding points [119]. To balance the usability, efficiency, accuracy and reliability, frame lengths like VGA (640×480 pixels) and Stefan (352×288 pixels) are accepted as suitable ones, such as for example in ARCore, ARToolKit and MixedRealityToolKit [96][119][93]. Although some AR applications could handle high resolution images, a small-sized region of markers (from $200 \times 200to514 \times 414$ in VGA sized frames) are uploaded for processing when focusing on its marker detection system such as for example in ARToolKit (ATK) or in Siemens Corporate Research (SCR) [119]. Hence, in this chapter, the average length of uploaded video frames (or region of markers) to the EC is chosen from the following set: $\{200 \times 200, 400 \times 400, 600 \times 600, 800 \times 800\}$ (pixels). These video frame lengths can be translated to approximately 0.12MB to 2MB in terms of data size [67]. After extracting features from frames, each compact feature representation (using fisher vectors for example) are around 25KB ($[100, 300]$KB in total) [78]. Each ARO is $(0, 50]$MB [117] and we assume that follows a normal distribution. Since we simulate in a small scale network, a nominal LSTM network, including a single LSTM layer is utilized. The acceptable validation accuracy is set above 85% and if the trained network only fits the training set without satisfying this requirement for other testing sets (over-fitting), the network has to be retrained with fewer neurons and a larger dropout rate. In the simulation, we start from 100 neurons and reduce by 5 until finding a suitable LSTM network with 80 neurons. 90% of 300 route vectors with corresponding optimal decisions are used for training, while the rest 10% are used for validation. The initial learning rate is set to 0.005, the maximum number of epochs is 160 and the dropout probability is 5% to avoid over-fitting. These key parameters used in the numerical investigations are summarized in Table (3.2).

Table 3.2 Simulation parameters

Parameter	**Value**
Number of available ECs	6
Number of requests	$[20, 40]$
AR object size	$(0, 50]$ Mbyte
Total moving probability	$[0, 1]$
Frame length	$\{200^2, 400^2, 600^2, 800^2\}$ pixels
Compact feature representations	$[100, 300]$ KByte
CPU Memory Capacity	16 GByte
CPU frequency	$2-4$ GHz
CPU cores	8
CPU portion per VM	$0.125-0.25$
Computation load	10 cycles/bit
Wireless transmission rate	$[100, 120]$ Mbps
Average hop's latency	2 ms
Cache miss penalty	25 ms
Number of user route vectors / optimal decisions	300
Initial LSTM learning rate	0.005
Maximum number of epochs	160
LSTM Dropout probability	5%

The weight parameter (μ) as defined in (3.28a) belongs to $[0, 1]$ while the assumed β value for the FACT scheme is a positive number (the assumed range is $[0, 100]$ in [67]). When implementing the FACT scheme for comparison, detailed settings of simulation in [67] are adjusted according to the system settings in this work. Besides, cache limit and cache miss penalty are not considered in FACT [67]. During implementation, the penalty is triggered whenever FACT's solution excesses the limit. In order to compare the two schemes fairly, after deciding a value for μ, we calculate the corresponding accuracy in the proposed scheme and then experiment with different β values for the FACT scheme to find a suitable one so that both schemes share the same level of accuracy. The relation between the weighting parameters μ and β when sharing a common level of accuracy is presented by Fig. (3.7). It shows a very different tendency between the FACT scheme and the proposed optimal framework based on mathematical programming. For example, when our proposed scheme sets the weight μ as 0.5 which regards accuracy and latency as equal, the weight β of the FACT scheme is only 20 to achieve the same level of frame accuracy. As pointed out by authors in [67], such a small value indicates a decreased importance on the frame accuracy. Hence, by considering also mobility, service decomposition and an increased set of constraints, the proposed scheme is able to provide better decision making and hence increase the performance of the application. The comparison of service delay with the same level of frame accuracy are discussed in detail in the next section.

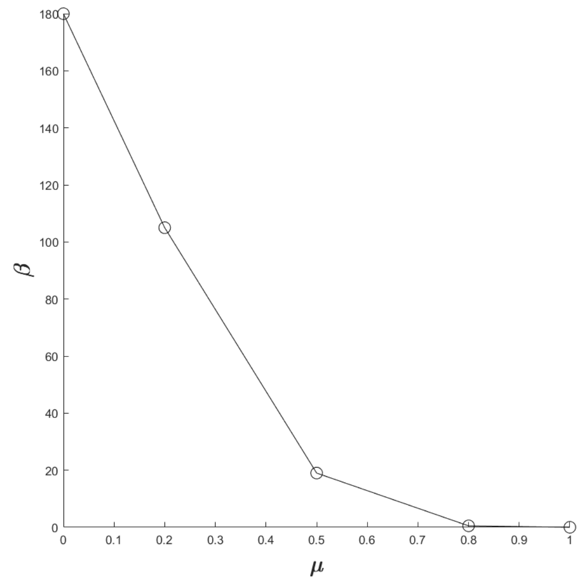


Fig. 3.7 Relation between weight parameters μ and β

Three other schemes are also designed for comparison reasons. These are the random selection scheme (RandS), the closest-first scheme (CFS) [101] and the utilization based scheme (UTIL) [95]. The RandS scheme randomly selects an available EC and picks a free VM from the selected EC. The CFS scheme focuses on the nearest EC to the user and accepts an available neighbor EC as a backup choice [101]. Similarly, the UTIL scheme selects the closest EC at first but will turn to neighbor least loaded ECs if the closest server is highly utilized (for example over 80% of the total VMs are occupied) [95]. Note that the penalty still works for the above schemes when their first and backup choices are all not feasible in the network. Overall delay and after mobility event delay are defined here for a more in-depth analysis of different schemes and a detailed comparison. The overall delay includes the aggregated latency consumed in communication and is in essence the result of formula (3.28a). However, the so called "after mobility event delay" captures the communication process and acquired delay between the assigned EC and the potential destinations. The results reported below are average values stemming from 50 Monte Carlo simulations.

3.4.2 Simulation Results

As eluded in the related work section, mobility awareness in edge cloud support is one of the main difference between the proposed schemes and other schemes. The LSTM scheme, which is trained by the Optim scheme, is able to capture efficiently the effect of mobility. Table (3.3) evaluates the impact of the total moving probability on the performance of schemes' delay (the results presented are for 30 Requests, the value of the weight μ is equal to 0.5 and we assume 14 units for the EC capacity). Observe from rows 1-6 in table (3.3) that the delay reduction achieved by the Optim scheme is increasingly more significant compared to the other schemes when the total moving probability increases. Note that, as expected, all schemes share a similar level of overall delay when there is enough capacity and no mobility. As compared with the FACT scheme, the Optim scheme reduces delay up to 8% and maintain the similar frame accuracy ($\beta = 20$) when the total moving probability increases. From row 7-12 in table (3.3), CFS, UTIL and FACT schemes faced increased delays which is significant evident in the after mobility event because they always try to deploy the service at an adjacent EC to the end user instead of considering potential destinations due to mobility effects. Therefore, the Optim scheme owns an obvious advantage compared to the other schemes in a high moving probability scenario, especially in the so-called after mobility event.

Table 3.3 Combined delay table for different moving probabilities(msec)

P	Optim	FACT	CFS	RandS	UTIL	SAMAR	LSTM
0	47.2	47.2	47.6	56.1	50.4	47.4	48.5
0.2	49.4	50.3	50.8	59.1	53.1	49.8	50.9
0.4	51.5	53.5	54.0	62.3	55.7	52.1	52.7
0.6	53.6	56.7	57.3	65.4	58.5	54.5	55.2
0.8	55.8	59.9	60.5	68.6	61.3	57.0	57.2
1	58.2	63.2	63.8	71.8	64.1	59.7	59.9
Overall delay for schemes under different moving probabilities							
0	0	0	0	0	0	0	0
0.2	10.6	12.0	12.2	11.2	12.4	10.8	10.8
0.4	21.3	24.1	24.6	22.7	24.7	21.6	21.7
0.6	32.0	36.1	36.9	33.9	37.0	32.4	32.6
0.8	42.5	48.1	49.2	45.4	49.4	43.1	43.4
1	53.2	60.2	61.5	56.7	61.8	53.9	54.2
After mobility event delay for schemes under different moving probabilities							

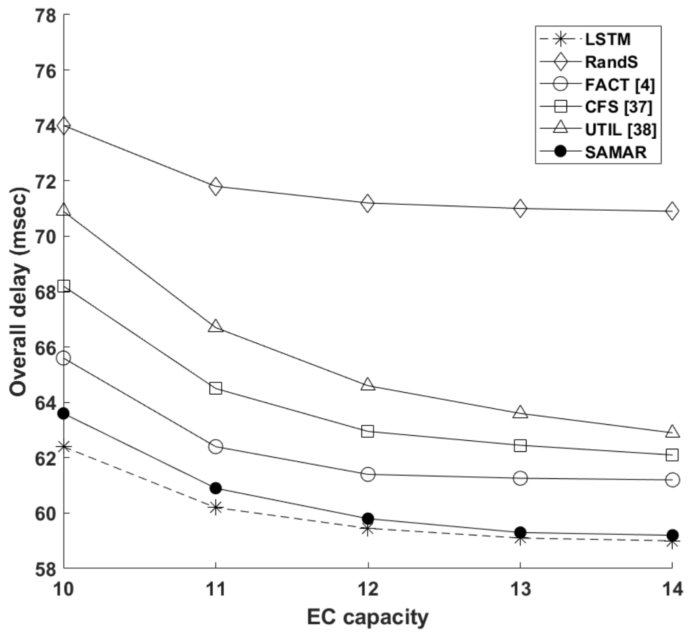


Fig. 3.8 Overall delay for schemes under different EC capacities (P=1, $\mu = 0.5$ and 30 requests)

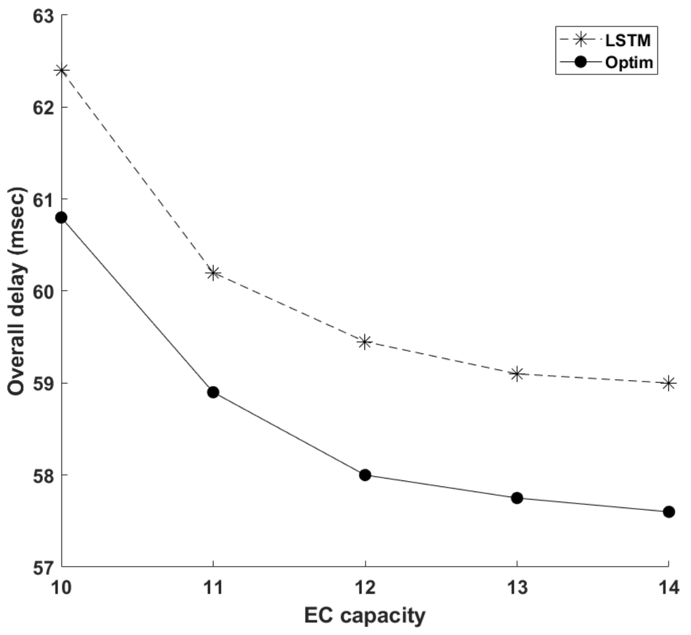


Fig. 3.9 Difference in performance between LSTM and Optim

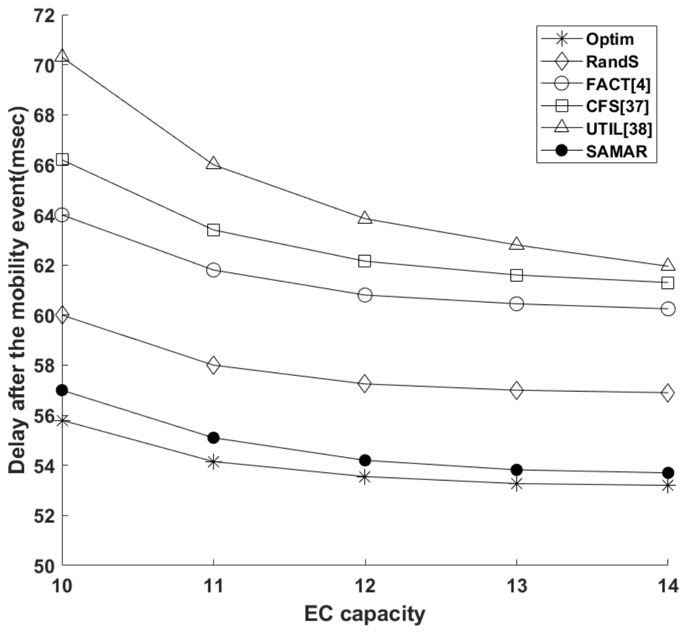


Fig. 3.10 After mobility event delay for schemes under different EC capacities (P=1, $\mu = 0.5$ and 30 requests)

Another aspect is the available EC capacity, i.e., the number of unused VMs in an activated EC. Although the FACT scheme does not explicitly utilize the notion of VMs as is the case of the proposed schemes and the Optim framework in this chapter, it still has computing capacity measured by TFLOPS [67]. Thus, by translating this computing capacity to EC capacity, results of the FACT scheme can be compared with others'. The range of EC capacity is 10 to 14 and the corresponding computing capacity in the FACT scheme is 0.63 to 0.88 TFLOPS. As shown in Figure (3.8) and (3.10), when EC capacity increases, the average delay for all schemes decreases because an increased available capacity usually leads to a better potential solution and a lower probability of overload and cache miss. Notice that the UTIL scheme is more sensitive to capacity than others and hence has the fastest dropping rate. The RandS scheme is worse in overall delay than CFS, FACT and UTIL schemes, but is better in the after mobility event. The reason is that the RandS scheme treats all active servers equally while other three schemes tend to select the closest server to the user's initial location. From this figure and Fig. (3.9), the LSTM-based scheme is slightly better than the SAMAR scheme. It is interesting to show that the gap between the Optim scheme and the proposed LSTM-scheme is also quite narrow. Different values of parameters will not interfere its learning process and affect its learning effect. Therefore, the gap between the LSTM-based scheme and the Optim scheme will maintain narrow at different network conditions, which reveals the reliability and stability of this proposed scheme. Clearly, the advantages of the proposed three schemes become more evident in a congested network. Most greedy schemes tend to allocate MAR services to a few ECs. They neglect the flexibility of decomposition and hence will suffer from a serious penalty because of the limited memory space and computing resources in the case of network congestion episodes.

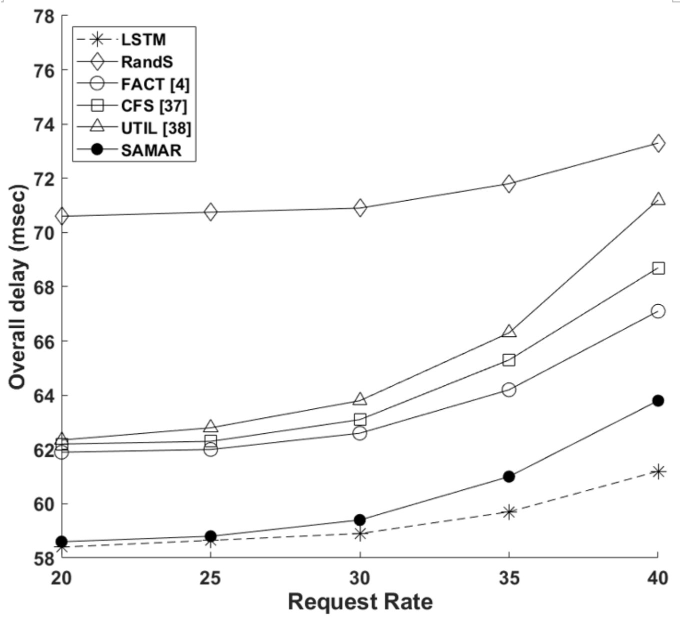


Fig. 3.11 Overall delay for schemes under different request rates (P=1, $\mu = 0.5$ and 14 units of EC capacity)

We are also looking into the effects of increased number of requests stemming from mobile augmented reality users which increase network utilization and congestion in the

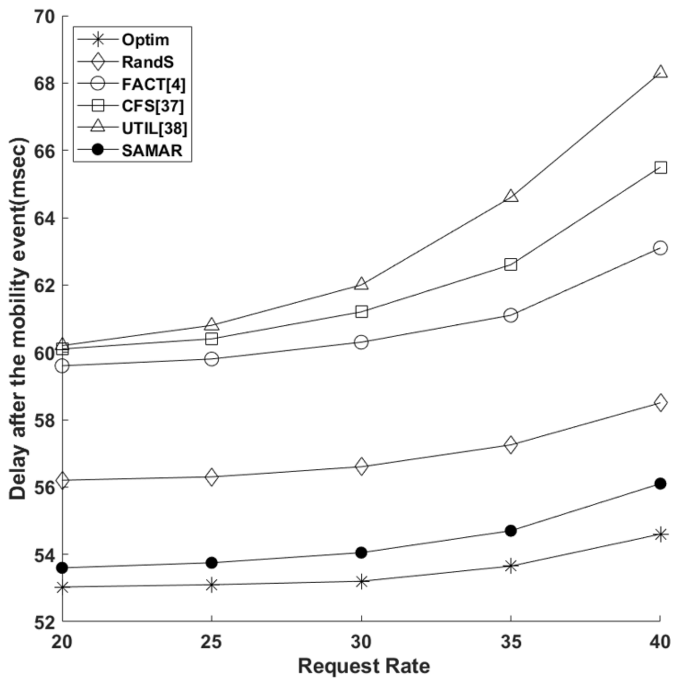


Fig. 3.12 After mobility event delay for schemes under different request rates (P=1, $\mu = 0.5$ and 14 units of EC capacity)

network. The request rate is defined as the number of requests served in the given short time slot. As expected, both types of delay has an increasing trend for all schemes as the request rate increases but the Optim scheme seems to better maintain lower overall latency with a slower ascending trend. These findings are illustrated in Figure (3.11) and Figure (3.12). Observe that the CFS and FACT schemes witness an increased delay and this is because their EC selection methods create several "hot spots" areas and these "hot spots" lead inevitably to higher delays as the number of requests increases. Also, the UTIL scheme by redirecting requests to adjacent ECs when an EC is operating at more than 80% utilization make this scheme also this scheme more vulnerable to network congestion episodes due to increased number of requests. However, it performs better under a low utilization level in a mobility event due to its balanced workload. Finally, the proposed SAMAR scheme approaches the Optim scheme and is less sensitive to increased congestion compared to the other baseline schemes.

Table (3.4) shows the variation of delay under different values of weight μ (30 Requests, P=1 and 14 units of EC capacity). The Optim scheme is around 10.2% less than the FACT scheme in overall delay and around 14.0% less in after mobility event delay. According to Table (3.4), the SAMAR scheme achieves a similar level of delay to the Optim scheme. Compared with the SAMAR scheme, the LSTM-based scheme is more stable and only a bit inferior to the SAMAR scheme with a large weight parameter.

Table 3.4 Combined delay table for different weights(msec)

μ	**Optim**	**FACT**	**CFS**	**RandS**	**UTIL**	**SAMAR**	**LSTM**
0	163.7	177.8	179.3	184.2	180.8	167.1	165.4
0.2	104.9	114.2	115.3	122.6	116.1	108.0	106.5
0.5	58.2	63.1	64.3	71.7	64.1	59.6	59.6
0.8	29.5	33.9	34.9	41.7	35.5	30.7	30.8
1	29.4	33.9	34.8	41.6	35.3	30.5	30.7
Overall delay for schemes under different weights							
0	158.9	174.7	176.7	169.4	177.1	161.5	160.8
0.2	100.3	111.0	112.7	107.8	113.3	102.6	102.9
0.5	53.2	60.2	61.3	56.7	61.9	53.8	54.7
0.8	24.7	30.9	32.3	26.8	33.2	25.3	25.9
1	24.7	30.8	32.2	26.8	33.1	25.2	25.9
After mobility event delay for schemes under different weights							

In simulations, when comparing predictions (LSTM-based) and optimal solutions (Optim), we get $rmse < 1$ (root mean square error), $\delta < 11\%$ (relative error) and $r^2 > 0.88$ (determination coefficient) in most of the cases. This is a clear indication that the predictions can be deemed as reliable and highly competitive to the optimal solutions. According to Formula (3.29) and (3.30), small values of *rmse* and δ indicate that the LSTM-based scheme generates a decision policy that closely resembles the optimal solution. The comparison of *rmse* values of all schemes is shown by Table (3.5). The indexes have a location information, i.e., access routers which are topologically close have also adjacent indices. The user mobility is limited to adjacent servers within each period for active MAR sessions. Thus, the high similarity to the optimal solutions indicates the high-quality solutions of the LSTM-based scheme. In addition, the value of r^2 also reveals that the LSTM-based scheme has a strong fitting ability in the MAR service decomposition problem. It is rarely the case in experimentation when the request is served by a remote cloud server, hence triggering a penalty. The variation of training and validation accuracy is shown in Fig. (3.13). The LSTM-based scheme gradually increases its decision making quality and closely follows the training set. The training and validation accuracy increase with iterations and eventually converge at 93.6% and 88% respectively.

$$rmse = \sqrt{\frac{\sum_{i=1}^{n} (YTest_i - YPred_i)^2}{n}} \tag{3.29}$$

$$\delta = \frac{\sum_{i=1}^{n} \left| \frac{YTest_i}{YPred_i} - 1 \right|}{n} \times 100\% \tag{3.30}$$

Table 3.5 RMSE Values of All Schemes

Algorithm	LSTM	RandS	CFS	FACT	UTIL	SAMAR
RMSE	0.9	6.8	1.9	1.5	3.4	1.3

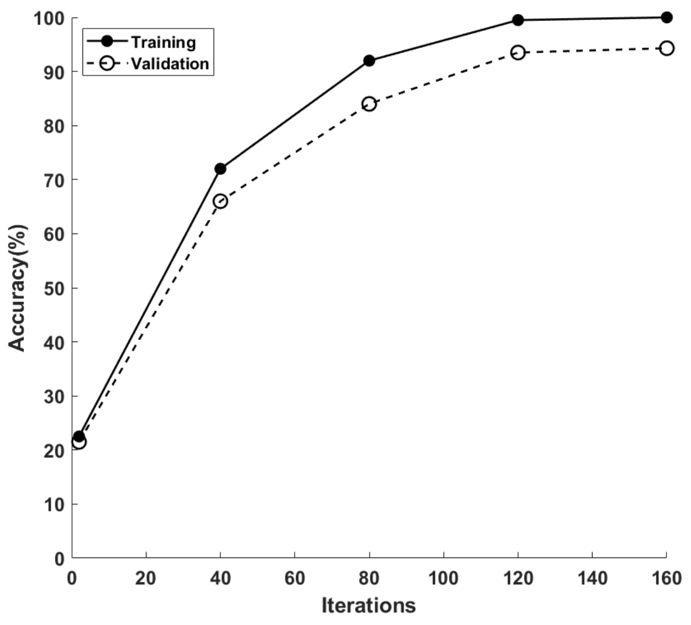


Fig. 3.13 Training and Validation Accuracy with increasing iterations

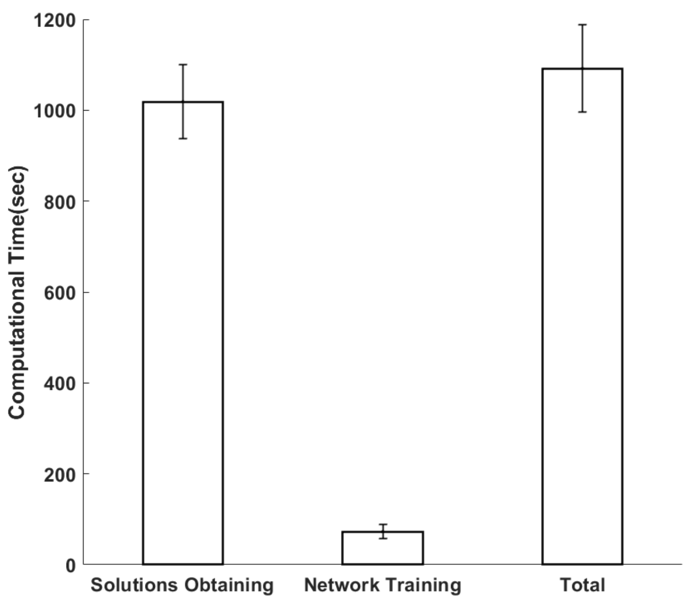


Fig. 3.14 Average execution time of the training phase (30 Requests and 14 units of Capacity)

Compared with the Optim scheme, the SAMAR scheme and the inference stage of the LSTM-based scheme are efficient in terms of complexity. The computational complexity of the SAMAR scheme is $O(n^2)$. According to (3.6) (executed once), the SAMAR scheme has a slight increased complexity compared to RandS, CFS and UTIL. Although relaxing an ILP into an LP problem in general greatly reduces the computational time for FACT[67], constructing and solving linear programming optimizing problems still consume more time than other schemes. The Optim scheme solves a complex MILP problem and hence is the most time consuming one. As shown by Fig. (3.14) and (3.6), the LSTM-based scheme suffers from a time consuming process for obtaining the solutions for network training, but its inference stage itself is efficient compared with other algorithms. Note that its solution obtaining and network training stage only needs to execute once offline, its efficient inference stage is more dominant in the whole process. However, it is also possible to train the LSTM neural network with advanced metaheuristic algorithms (like SAMAR for example) and provide real-time decision making during the inference phase.

Table 3.6 Processing time of algorithms

Algorithm	**Average Processing time(sec)**	**STD**
RandS	1.076	0.151
CFS	1.083	0.156
UTIL	1.091	0.155
SAMAR	1.497	0.163
FACT	2.132	0.447
LSTM	1.727	0.205
Optim	201.506	21.965

*tested in a PC with intel i7, 6500U, 2 cores

3.5 Conclusions

Mobile augmented reality applications are still in a rather embryonic state but they currently attracting significant attention from both academia and industry stemming from the increased capabilities offered from the network side as well as the terminals. The ability of augmented reality services to overlay digital content over the physical space and surroundings results in an immersive metaverse type of environment which open up the possibility for a plethora of potential use cases and applications. As with respect to mobile wireless networks, mobility has been proved to be an important factor in service latency of MAR systems. Previous research mainly focused on various techniques of offloading processing without considering the decomposition of MAR functions and the effect of mobility in an explicit manner. To this end, this chapter proposes an optimization framework that considers a rich set of augmented reality specific constraints, provides service decomposition and allows a trade off between service latency and frame accuracy to improve the communication efficiency. Based on this scheme, a simulated annealing based mobility aware AR (SAMAR) algorithm and a long short term memory based scheme (LSTM) neural network for predicting edge clouds for anchoring MAR functions are designed for achieving high-quality solutions whilst being amenable for real-time implementation. Both proposed schemes deliver competitive performance and focusing on the inference stage, the long short term memory based scheme (LSTM) shows enhanced performance and shorter execution time.

As illustrated by a wide set of numerical investigations, the proposed set of solutions can efficiently decompose the augmented reality service to available edge cloud resources by considering mobility patterns of users and hence increase the quality of decision making compared to other previously proposed and baseline schemes.

This chapter focuses on basic mobility scenario that transfers user physical mobility into moving probability and the actual moving traces are not tracked. The utilization of a random tracing model might greatly makes the work more practical but this requires more efforts on the moving process and a much deeper exploration in changes of view points during the mobility event. Current solutions hanlde requests together in the service time slot and directly sacrifice frame accuracy for better latency. This also indicates the limitation that, some users sending out requests late in the time slot might suffer from both low quality and late response. In following chapters, we further think about user view points and other potential trade-offs that might partly overcome the limitation.

Chapter 4

Field of View Aware Proactive Caching for Mobile Augmented Reality Applications

Based on the previous chapter, we further extend the research into multiple 2D field of views (FoVs) of 3D AROs. When confronted with limited resources, we propose a novel trade-off between service quality and latency through replacing some AROs by their most popular 2D FoVs.

4.1 Introduction

The work in [42] shown by Chapter 3 outlines how a MAR application can be decomposed into a series of granular micro-services together with an optimization framework to provide optimal pro-active mobility-aware decision making in terms of EC assignment. As illustrated in Fig. (4.1), the captured video of a MAR application is preprocessed at the terminal and then frames with AR objects are transmitted for detection, extraction and recognition. The local cache is searched to find if there is a match. Finally, the matched results are transmitted back for presentation. According to their features, these MAR functions are categorized into two types: computational intensive ones that require significant CPU resources and storage intensive ones that require significant cache resources. Clearly, they have a predefined order when working as a service chain. When there is no mobility, it could be possible to deploy the MAR application at a single server. However, it is clear that, without decomposition, the complete MAR application uses significant levels of CPU and cache memory resources. On the other hand, decomposition allows a more flexible allocation but the proactive caching is still not necessary here because there is no target objects distributed in the area. When bringing in both AR objects and the user mobility, an isolated module could be designed to provide predictions for proactive resource allocation according to target AROs and the user's possible future destination [2]. Thus, the given prediction not only contains where to set functions and send requests, but also indicates what AR content should be allocated to the target server.

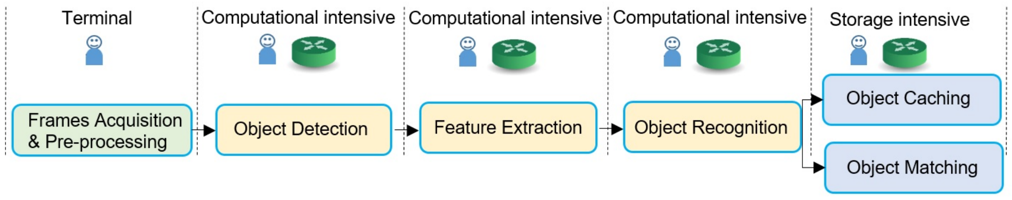


Fig. 4.1 The flow of the different nominal mobile AR functions, their characteristics and the potential location where those functions can run (terminal and/or at the EC).

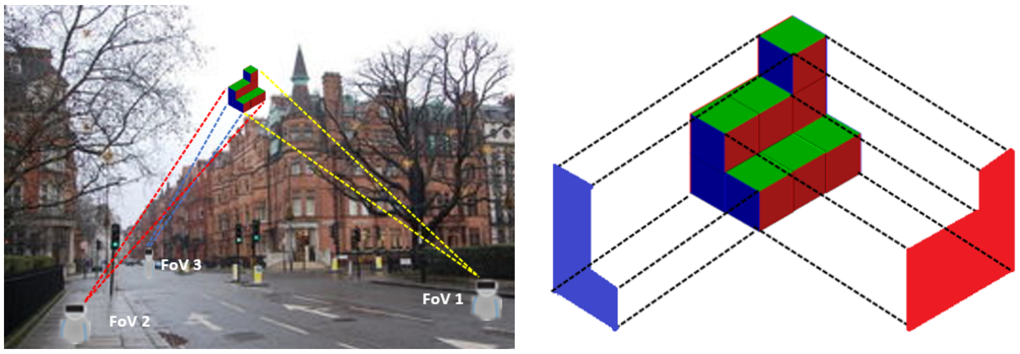


Fig. 4.2 Possible FoVs from different viewpoints on street (a) and Example FoVs of the target ARO (b)

As illustrated in Fig. (4.2), the different field of views (FoVs) can be achieved when viewing the target ARO from different positions and two such typical 2D projections of the target 3D ARO are presented as examples. Comparing with caching 3D AROs, which are usually in the order of tens of Mbytes in terms of size[117], caching their 2D counterparts from given FoVs can be another choice. These 2D FoVs are only several hundred KBytes in size and can also be easily reconstructed or conversed back (if needed) into 3D objects through algorithms like correlation-based, tracking-based and deep extraction [77][102][41]. In addition only few FoV can be deemed as of interest. For example the work in [8], collected and analyzed 295500 viewpoint samples from 153 volunteers. Subsequently, they splitted a 360° range into several groups; for example, in 90% of time, the X, Y and Z angles are within $-15.31^\circ \sim 15.06^\circ$, $-34.73^\circ \sim 35.14^\circ$, and $-8.61^\circ \sim 8.62^\circ$ [8]. This means the user prefer to observe the target in this angle which makes such FoV a popular one. In [8][55] and [122], the probabilistic model are developed and learn from the historical data of FoVs. According to results in [8], when considering applying 2D figures as an alternative, it is reported that up to 80% bandwidth can be saved through caching only figures of popular FoVs. In addition, 3D AROs are more complex and the matching function requires a pose calculation stage for them. Thus, 3D AROs suffer from a higher computation complexity and require more computing resources. Although proactive caching 2D FoVs with high probability cannot always ensure the proper pose of the target ARO, it owns an obvious advantage in massively reducing caching and computing resources requirements. Clearly, more stored FoVs per ARO leads to a higher probability of a cache hit during matching but consumes more cache space and more processing delay. The differences between nominal caching and pro-active caching for MAR applications in this chapter is further revealed by Fig. (4.3). Therefore, the joint optimization problem in this chapter subjects to strict edge resources constraints and seeks the balance between service latency and allocation of FoVs in the mobility event.

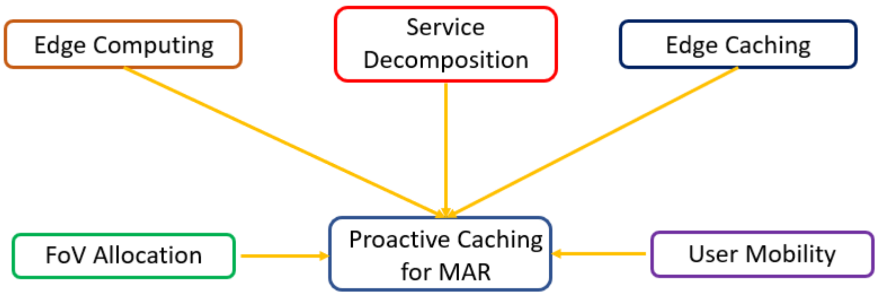


Fig. 4.3 Pro-active Caching for MAR

Constructing and solving a complex mixed integer linear problem is quite time-consuming and cannot respond to network changes in time. Therefore, the previous scheme should be improved to take into account both high-quality solutions and computational efficiency. Leveraging machine learning (ML) techniques on the network, such as neural networks or reinforcement learning, are considered to have the potential to solve resource allocation problems in different types of networks [10][73][53]. Thus, a well-known ML technique, long short term memory (LSTM), is applied in this chapter to learn from optimal solutions and provide reliable predictions. LSTM is widely accepted as an enhanced recurrent neural network due to its ability in memorizing significant information and forgetting unnecessary inferences to overcome potential gradient vanishing and gradient exploding problems [40][125]. As pointed out by our previous work [42], the execution sequence of two main types of functions is pre-defined during the decomposition of MAR services. Therefore, the correct order of the path could be figured out whenever the assignment is provided. An extra feasibility check stage is added to ensure the predictions can still satisfy constraints and fit the original network environment.

As discussed earlier, the obvious advantage in saving edge resources when caching 2D FoVs motivates our research in its optimization problem. The mathematical programming formulation consists of the service latency based on [42], the size and allocation of pre-cached 2D FoVs; in that case, an edge cloud resource allocation joint optimization framework is proposed to proactively allocate resources and to satisfy related requirements of MAR applications. The user mobility is considered explicitly and the overall delay is minimized as a multi-objective optimization problem. Although the optimal solution is desirable however such a framework cannot be used to provide real-time decision making since solving a mixed integer mathematical program suffers from the curse of dimensionality which means that requires significant amount of time to provide a solution. Therefore, in this chapter, an LSTM-based approach is further explored where the optimal solutions are used to train the deep neural network in an offline manner.

4.2 System Model

Similar to Chapter 3, a set $\mathbb{M} = \{1, 2, ..., M\}$ is defined to represent the available ECs in the network and multiple users are assumed to create MAR service requests $r \in R$. With η and ρ we also define computational intensive and storage intensive MAR functionalities respectively [42]. The allowed destinations for a user are limited to adjacent access routers. Thus, The probability of a user moving from the initial location to an adjacent server can be defined as $p_{rij} \in [0, 1]$, where adjacent servers $\{i, j\} \subset \mathbb{M}$. The location of executing one functionality for a request can be denoted as $L_{\eta r} \in \mathbb{M}$ or $L_{\rho r} \in \mathbb{M}$. Also, in a typical MAR service, each request requires the execution of two functionalities in a pre-defined order [42]. Thus, the assignment of one request can be arranged according to the route of its flow. The route matrix can be denoted as $RT_r = [s_r, L_{\eta r}, L_{\rho r}, d_r]$. This matrix can be provided by the optimal scheme and consist a route assignment set $\mathbb{RT}$ [42]. In LSTM training, requests' initial locations and their destinations can be fed as inputs and decisions of where to anchor MAR functionalities can be treated as the output. Thus, we further split the route matrix into $X_r = [s_r, d_r] \in \mathbb{X}$ and $Y_r = [L_{\eta r}, L_{\rho r}] \in \mathbb{Y}$. The major part of the route set $\mathbb{RT}$ is used as input and fed into the LSTM-based model for training. The rest of this set remains as a testing set, Therefore, we denote with $\mathbb{XTrain}, \mathbb{XTest} \subset \mathbb{X}$ and $\mathbb{YTrain}, \mathbb{YTest} \subset \mathbb{Y}$ to differentiate training and testing sets. The output is the predictions made by the trained model and hence denoted as $\mathbb{YPred}$.

At first, a joint optimization scheme considering the service delay, the total size and allocation of pre-cached FoVs is designed to track the MAR requests in the network. Denote $p \in \mathbb{P}$ as a 2D FoV to observe an 3D ARO. Then the decision variable h_{rlj} in [42] can be rewritten as h^p_{rlj} that decides whether to pre-cache the FoV p of the ARO l required by the request r at the location j.

$$h_{rlj}^{p} = \begin{cases} 1, \text{ if the FoV } p \text{ of an ARO } l \text{ required} \\ \quad \text{by the request } r \text{ is cached at node } j, \\ 0, \text{ otherwise.} \end{cases} \tag{4.1}$$

With extra constraints,

$$\sum_{j \in M} \sum_{l \in N} \sum_{p \in P} h_{rlj}^{p} \geqslant 1, \, \forall r \in R \tag{4.2}$$

$$\sum_{r \in R} \sum_{l \in N} \sum_{p \in P} h_{rlj}^{p} \leqslant 1, \, \forall j \in M \tag{4.3}$$

(4.2) means at least one FoV of an ARO is required by each request. (4.3) means even the same FoV is required by different requests at the same location, it only needs to be pre-cached once. Then the decision variable for cache hit/miss can be written as,

$$z_{rj} = \begin{cases} 1, \text{ if} \sum_{l \in N} \sum_{p \in P} h_{rlj}^{p} \geqslant L_r, \\ 0, \text{ otherwise.} \end{cases} \tag{4.4}$$

The constraints (19b) and (19d) in [42] that reveal the cache limitation and the cache hit/miss relation can be updated as,

$$\sum_{r \in \mathbf{R}} \sum_{l \in \mathbf{L_r}} \sum_{p \in \mathbf{P}} h_{rlj}^{p} l_{rj}^{p} \leq \Theta_j, \forall j \in \mathbf{M} \tag{4.5}$$

$$\sum_{l \in \mathbf{N}} \sum_{p \in \mathbf{P}} h_{rlj}^{p} + \varepsilon \leq L_r + U(1 - q_j) \, \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{4.6}$$

where ε is a small positive real number [42]. More pre-cached FoVs lead to an extra burden for processing the matching function. Hence, denote the size of a FoV p of an ARO l at location j as l_{rj}^{p}. The processing delay of the matching function can be written as,

$$W_{rj} = \frac{\omega(F_{\rho r} + \sum_{l \in N} \sum_{p \in P} h_{rlj}^{p} l_{rj}^{p})}{f_V^j} \tag{4.7}$$

Where ω is the computation load and $F_{\rho r}$ is the extracted features defined in [42]. Noticing that when processing the matching function at the location j ($W_{rj} y_{rj}$), the production of decision variables $h_{rlj}^{p} y_{rj}$ appears inside and can be handled through bringing in a new decision variable α_{rlj}^{p} with following constraints,

$$\alpha_{rlj}^{p} = h_{rlj}^{p} y_{rj} \tag{4.8a}$$

$$s.t. \alpha_{rlj}^{p} \leqslant h_{rlj}^{p} \tag{4.8b}$$

$$\alpha_{rlj}^{p} \leqslant y_{rj} \tag{4.8c}$$

$$\alpha_{rlj}^{p} \geqslant h_{rlj}^{p} + y_{rj} - 1 \tag{4.8d}$$

The service latency L maintains the same as (19a) in [42] and L_{max} here means the maximum possible latency. Thus, we have,

$$\frac{L}{L_{max}} \in [0,1] \tag{4.9}$$

The total size of pre-cached 2D figures can be written as,

$$S = \sum_{r \in R} \sum_{j \in M} \sum_{l \in N} \sum_{p \in P} h_{rlj}^{p} l_{rj}^{p} \tag{4.10}$$

S_{max} here means the maximum allowable size of pre-cached FoVs. The allocation of FoVs of an ARO at a location is $\sum_{r \in R} \sum_{p \in P} h_{rlj}^{p}$. Denote $t_{lj}^{p} \in \mathbb{T}_{\mathbb{lj}}$ as the probability of viewing a certain FoV at the location. Due to the cache size constraint, sometimes not all FoVs but only several preferred ones of a target ARO could be cached. For one ARO, maximizing probabilities means caching its FoVs in an descending order of probability. Then, we sum probabilities of all cached FoVs of all target AROs to allow the algorithm to cache the most preferred FoVs. Thus the summary reveals the quality of FoV allocation and can be written as,

$$A = \sum_{j \in M} \sum_{l \in N} \sum_{r \in R} \sum_{p \in P} h_{rlj}^{p} t_{lj}^{p} \tag{4.11}$$

A_{max} here means the sum of probabilities of all available FoVs of target AROs. As mentioned earlier, if the cache space is limited, pre-caching several FoVs with higher probabilities from multiple AROs is better in achieving a better quality than storing the same number of FoVs but from only several AROs. In addition, $\mathbb{T}_{\mathbb{lj}}$ is defined as a descending probability vector and t_{lj}^{p} inside can be achieved through,

$$t_{lj}^{p} = \begin{cases} 1, \text{ if } p = 1, \\ \text{rand } (t_{lj}^{p-1}), \text{ if } p > 1. \end{cases} \tag{4.12}$$

Where the function rand(x) means generating a random number within $[0,x]$. Then to limit the sum of probabilities of an ARO's different FoVs as 1, the normalization should be done and each element t_{lj}^{p} in the vector can be updated by $\frac{t_{lj}^{p}}{\sum_{p \in P} t_{lj}^{p}}$.

Therefore, the formula capturing both size and allocation can be written as,

$$\frac{1}{2}\left(\frac{S}{S_{max}}+\frac{A}{A_{max}}\right) \in [0,1] \tag{4.13}$$

Denote the weight parameter as $\mu \in [0,1]$, the joint optimization problem can eventually be written as,

$$min\ \mu\frac{L}{L_{max}}-\frac{1-\mu}{2}\left(\frac{S}{S_{max}}+\frac{A}{A_{max}}\right) \tag{4.14a}$$

$$\text{s.t. } z_{rj}=1-q_j,\ \forall j\in\mathbf{M}, r\in\mathbf{R} \tag{4.14b}$$

$$\sum_{r\in\mathbf{R}}(x_{rj}+y_{rj})\leq\Delta_j, \forall j\in\mathbf{M} \tag{4.14c}$$

$$\sum_{j\in\mathbf{M}}x_{rj}=1,\forall r\in\mathbf{R} \tag{4.14d}$$

$$\sum_{j\in\mathbf{M}}y_{rj}=1,\forall r\in\mathbf{R} \tag{4.14e}$$

$$\xi_{rij}\leq x_{ri},\ \forall r\in\mathbf{R}, i,j\in\mathbf{M} \tag{4.14f}$$

$$\xi_{rij}\leq y_{rj},\ \forall r\in\mathbf{R}, i,j\in\mathbf{M} \tag{4.14g}$$

$$\xi_{rij}\geq x_{ri}+y_{rj}-1,\ \forall r\in\mathbf{R}, i,j\in\mathbf{M} \tag{4.14h}$$

$$\psi_{rj}\leq z_{rj},\ \forall r\in\mathbf{R}, j\in\mathbf{M} \tag{4.14i}$$

$$\psi_{rj}\leq y_{rj},\ \forall r\in\mathbf{R}, j\in\mathbf{M} \tag{4.14j}$$

$$\psi_{rj}\geq z_{rj}+y_{rj}-1,\ \forall r\in\mathbf{R}, j\in\mathbf{M} \tag{4.14k}$$

$$x_{rj}, y_{rj}, h^p_{rlj}, z_{rj}, q_j, \alpha^p_{rlj}, \psi_{rj}, \xi_{rij}\in\{0,1\},$$
$$\forall r\in\mathbf{R}, j\in\mathbf{M}, l\in\mathbf{L}, p\in\mathbf{P} \tag{4.14l}$$

(4.2), (4.3), (4.5), (4.6)

(4.8b), (4.8c), (4.8d)

The constraint (4.14b) together with updated constraints (4.5) and (4.6) reveal the relation between pre-caching decisions and the cache miss/hit for each request[42]. The constraint (4.14c) is the cache limitation while (4.14d) and(4.14e) forces the once execution of each function of a request at a single server as explained in [42]. The constraints (4.8b), (4.8c), (4.8d) and (4.14f) to (4.14k) are brought in to solve products of decision variables for linearization.

The optimal assignments generated by the aforementioned ILP model are grouped and reshaped into a route matrix and then separated for training and testing. Since execution locations for functionalities can only be selected from the set $\mathbb{M}$, all possible different results can be calculated as selecting any two from M elements with order, which is M^2. For simplicity, each assignment can be given an index to show which type it belongs to. The LSTM-based model will then classify and create a mapping between the two input sets. After training, the model will provide its predictions according to the input training set. The predictions are checked for feasibility in terms if constraints such as cache and EC capacity limit are satisfied. If for an EC capacity is reached, remaining predicted requests will be allocated to an available neighbor EC as a backup choice. However, if both ECs are occupied, then the request is sent to a cloud deeper in the network and, as a result, an extra penalty is triggered. In order to shed light into the quality of the decision making, the assignments are compared with the optimal assignments to evaluate the quality of the LSTM-based scheme. The architecture of the LSTM-based model follows the nominal design and its state changing and gate controlling follows the original formula. The aim is to explore the potential of using a nominal LSTM network for service decomposition.

4.3 Numerical Investigations

Hereafter, the effectiveness of the proposed Optim and LSTM-based schemes are investigated via a wide-set of simulations and compared with previously proposed schemes.

We assume a typical tree-like network topology [42], with 20 ECs in total with 4 to 8 ECs being activated and up to 40 requests are sent by MAR devices. The remaining available resources allocated for MAR support of an EC is assumed to be CPUs with 4 to 8 cores and 16GB memory[42]. Each request requires a single free unit for each service function (for example a VM) and each 2D FoV of the target ARO is set from 0.1MB to 2MB to cover different use cases [67][77][102][41]. Each ARO is assumed to have up to 4 different FoVs. There are 14 available VMs in each EC, with equal splitting of CPU resources [42]. Furthermore, available remaining storage for each EC is assumed to range between 100 to 500 Mbytes for supporting MAR services. As eluded previously, users move between adjacent access routers from the initial location. Optimal solutions are calculated based on the proposed Optim scheme and a set of solutions are used to feed the LSTM neural network. From that set, 90% is fed into the model as a training set while the rest 10% is used for testing. In the LSTM layer, the initial learning rate is set to 0.005 and the maximum number of epochs is 160. The probability in the dropout layer is set to 5% to avoid over-fitting.

The proposed optimization framework (Optim) provides the optimal solutions by solving the underlying MILP problems [42]. The Mobility Oblivious Allocation scheme (MOA) finds optimal solutions but without taking the user's mobility effect into account[42]. Same as Chapter 3, baseline schemes also include RandS, CFS and UTIL. They are FoV-oblivious which means storing all FoVs without considering their probability. As shown by Fig. (4.4), the weighted fraction of stored FoVs' probability per ARO drops as expected when the weight μ increases. Table (4.1) shows the weighted fraction of stored FoVs' size per ARO under different weights. The value of such factor drops from 0.81 (over 3 FoVs per ARO) to 0.28 (slightly over 1 FoV per ARO) and hence the Optim scheme can save much storage resources as expected.

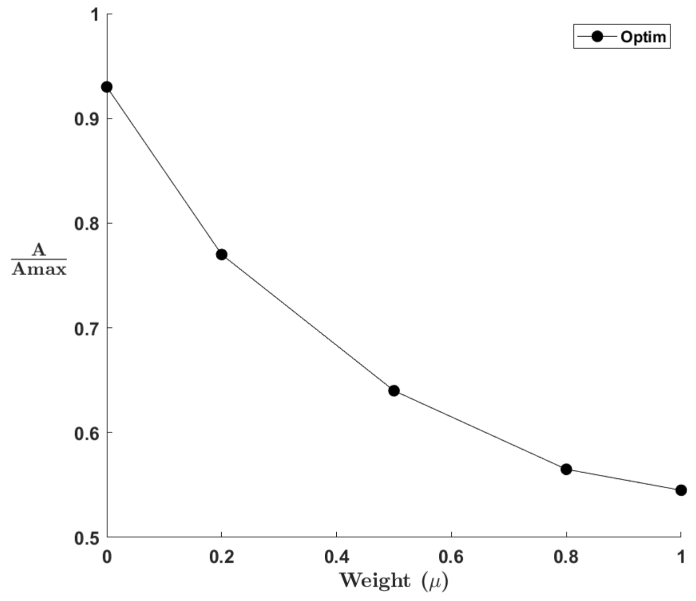


Fig. 4.4 Weighted Fraction of Stored FoVs' Probability per ARO under Different Weights (μ)

Table 4.1 Weighted Fraction of Stored FoVs' Size per ARO under Different Weights (μ)

Weight	**0**	**0.2**	**0.5**	**0.8**	**1**
$\frac{S}{S_{max}}$	0.81	0.62	0.46	0.34	0.28

According to Fig. (4.5), the proposed Optim scheme has the least delay whilst depicting a desired insensitivity as the weight increases. The gap between the Optim scheme ($\mu = 0$) and other baseline schemes increase with the number of requests and hence the Optim scheme owns an obvious advantage in a more congested network. The greedy schemes share a common tendency to execute decomposed MAR services on a few ECs. The flexibility of decomposition is reduced and hence these greedy schemes can suffer a penalty cost because of the narrow space and limited resources. However, the gains in delay between Optim schemes with different weights are becoming narrow. Compared to the CFS scheme, when $\mu = 0$, the Optim scheme has a similar weighted fraction of stored FoVs' probability per ARO but can save around 24.4% in terms of delay. When $\mu = 0.2$, the Optim scheme reduce A/A_{max} by 16% to achieve around 47% less delay than the CFS scheme. When $\mu = 0.5$, A/A_{max} reduced by 29% but for 62% less delay. In an extreme condition that $\mu = 1$, the Optim scheme saves 67.5% delay but has a 39% loss in the weighted fraction of stored FoVs' probability per ARO. Hence, a balance between delay and amount of FoV cache should be carefully considered. This is also supported by the narrowing gap between weighted fraction of stored FoVs' size per ARO according to Table (4.1). Corresponding average cache hit rate for these Optim schemes in the above case are shown by Fig. (4.6). When there is enough resources, pre-caching and matching target AROs at ECs are always better than ignoring them and suffering a much larger penalty to fetch further cloud data. Thus, the variation of average cache hit rate matches the variation of latency and clearly reveals that perception quality focus and growing requests are negative to cache hit. Fig. (4.7) further dives into the case when $\mu = 0.5$ that treats service latency and user perception quality equally. Obviously, more available VMs at an EC brings about more potential resources and result in better performance in overall delay and hence causes a dropping trend as expected. However, when zooming the gap between LSTM predictions and optimal solutions, we could see that the gap itself is not stable and might become larger under a large capacity (e.g. capacity is 13). Our Feasibility Checking Stage ensures each predicted server to be one of the activated server and hence sometimes have to accept a neighbor available one. Thus, LSTM based scheme might not keep approaching optimal solutions stably under any condition and in general fit congested network conditions better.

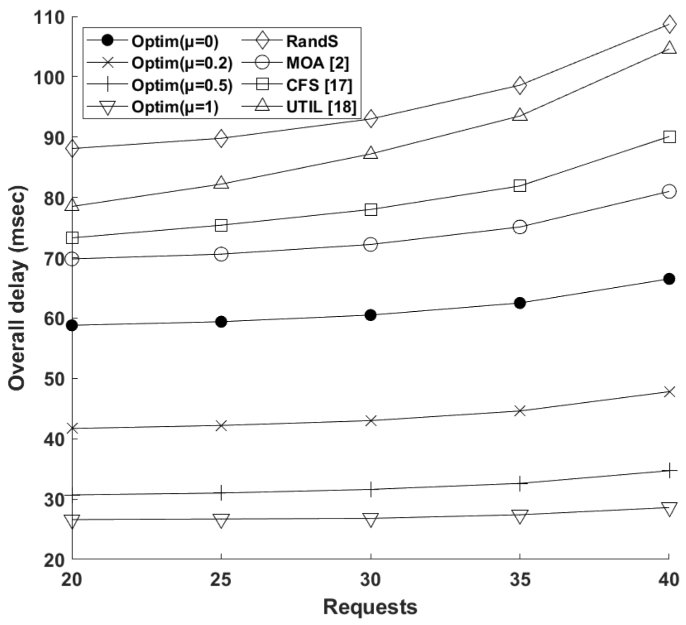


Fig. 4.5 Average service delay of different schemes under different numbers of requests (6 ECs and Capacity is 14)

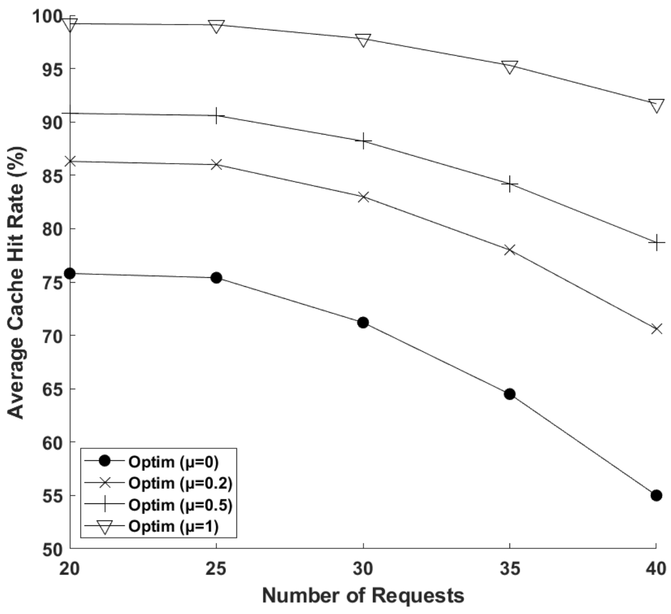


Fig. 4.6 Average cache hit rate of Optim schemes in different weights under different number of requests (6 ECs and Capacity is 14)

According to Table (4.2), the LSTM-based scheme can follow the Optim scheme closely and performs better when focusing on delay. According to Table (4.3), when there is no mobility and the network is not in a congested state, different schemes perform similarly. The Optim scheme shares the same performance as the MOA scheme and is also close to the CFS scheme. The LSTM-based scheme is only 1.8ms worse than the Optim scheme. Compared with other baseline schemes, its performance is still acceptable in this case.

Table 4.2 Delay (ms) for LSTM and Optim under different weights

Weight	**0**	**0.2**	**0.5**	**0.8**	**1**
LSTM	64.5	45.4	34.1	29.3	27.1
Optim	60.3	42.8	32.5	28.1	26.2

Table 4.3 rmse values and Delay in no mobility event ($\mu = 1$, 6 ECs, 30 requests and Capacity is 14)

Scheme	**Optim**	**LSTM**	**MOA**	**CFS**	**RandS**	**UTIL**
Delay (ms)	23.5	25.3	23.5	24.2	28.1	25.6
RMSE	-	1.5	7.0	2.6	2.1	4.4

When comparing predictions and optimal solutions, we also have $rmse < 1.6$ (root mean square error), $\delta < 14\%$ (relative error) and $r^2 > 0.83$ (determination coefficient) in most cases, which indicate the predictions can be deemed as reliable and highly competitive

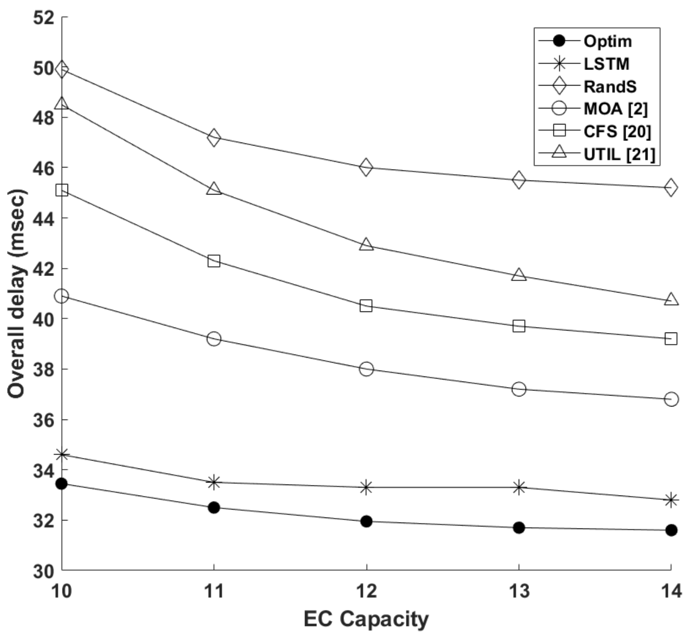


Fig. 4.7 Average service delay of different schemes with different EC capacities ($\mu = 0.5$, 6 ECs and 30 requests)

Table 4.4 Variation of Training and Validation Accuracy for the LSTM based scheme

Iterations	2	40	80	120	160
Training	22.5	70.9	91.0	98.3	98.4
Validation	21.3	66.2	83.8	93.7	94.3

to the optimal solutions. Small values of *rmse* and δ indicates that the proposed scheme generates a very similar version to the Optim's solution. *rmse* values compared to the Optim scheme are also shown by Table (4.3). We note that the indexes have a location information, i.e., access routers which are topologically close have also adjacent indices. In addition, the user mobility is limited to adjacent EC within each period which is a valid assumption for active MAR sessions. Thus, high similarity to the optimal solutions usually indicates that the corresponding scheme has the ability to provide high-quality solutions as well. The value of r^2 is close to 1 and hence the proposed model shows a high quality fitting ability in this problem. Clearly, the quality will deteriorate when the output solution lead to an infeasible allocation. In that case, requests will have to be served by a cloud server which will entail some penalty. Experimentation showed that this was rarely the case during the training set but tends to appear in the testing set. As shown by Table. (4.4), the validation accuracy increases with iterations, gradually appraoches to the training accuracy and finally maintains stable at around 94.3%. This indicates that the trained LSTM network could improve its predictions through more iterations and could approach optimal solutions under given network conditions.

Although obtaining optimal solutions and training the LSTM model based on the Optim scheme can be time-consuming, the inference stage itself is highly efficient. Also, the time required to train the network dominated by the time to provide the optimal solutions as shown in Fig. (4.8) (averaged by 50 executions). The average processing time of the inference stage is compared with other algorithms in Table (4.5) (executed once). Considering the fact that optimal solutions obtaining and network training stage only execute offline, the inference stage has a more dominant effect. Other greedy schemes need to check and handle different constraints (e.g. capacity and cache size) repeatedly in iterations and lead to a longer processing time. In the Optim scheme, a complex MILP problem is constructed and solved to find optimal solutions and hence is the most time-consuming one. Thus, the inference stage of the proposed scheme is the most efficient one among all schemes. This is of a significant importance for network operation since it achieves better performance and takes much less time to provide the decision making than other algorithms.

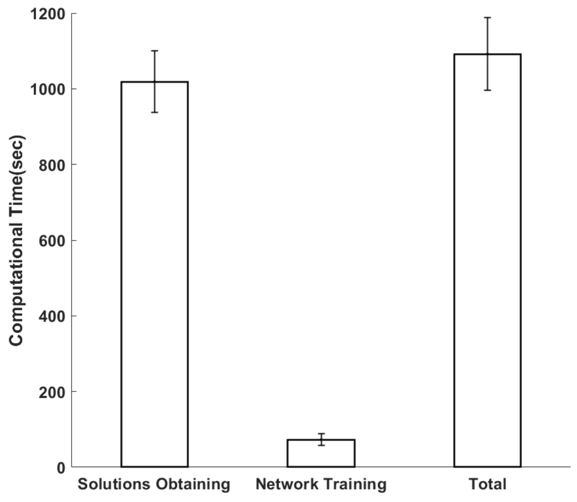


Fig. 4.8 Average execution time of the training phase (6 ECs, 30 Requests and Capacity is 14)

Mobile Augmented Reality (MAR) applications are sensitive to the user mobility and service delay. Service decomposition together with edge clouds allow for a more flexible resource allocation to better tackle the mobility event with AR objects for MAR applications. Enabling content-aware caching by storing high probability 2D FoVs instead of 3D AROs could significantly ease the requirements for storage and computing resources with limited effects on overall quality. To this end, this work proposes a joint optimization framework considering the balance between service delay, field-of-view aware proactive caching and FoV allocation. Furthermore, an LSTM-based deep neural network is explored to provide real time decision making. The LSTM network is trained using optimal solutions offline, and during inference can efficiently provide high-quality decision making in real-

Table 4.5 Average processing time of algorithms

Algorithm	**Average Processing time (sec)**	**STD**
RandS	1.076	0.151
CFS	1.083	0.156
UTIL	1.091	0.155
LSTM	0.427	0.065
Optim [42]	201.506	21.965

*tested by PC, intel i7, 6500U, 2 cores

time. This is validated by a series of simulations showing that the proposed scheme outperforms previously proposed solutions. Future avenues of research will articulate on the use of advanced meta-heuristic algorithms to provide near-optimal decision making for training the deep neural network in large network instances where optimal decision making is not efficient.

It is necessary to point out that the 2D FoV substitution should only happen when having really limited resources because applying static 2D figures sacrifice much user perception quality, especially during the mobility event. In previous cases, the request from each user is handled independently which means adjacent users might cause duplicate caching and computing. To overcome this limitation, we take ARO sharing into consideration and try to exploit the potential of video streams in the following chapter.

Chapter 5

Optimal Proactive Caching for Multi-View Streaming Mobile Augmented Reality

The chapter is based on [45] and [46]. Noticing that 2D FoVs are static and hinder users' view during moving process, we explore a novel solution that embeds 3D AROs into a continuous video stream. To realize flexible allocation and cut down resource consumption, we enable adjacent users sharing a same video stream without losing much user perception quality. A more realistic wireless channel model, Raleigh model, is applied and the user's physical location at initial and destination cells are handled in formulation.

5.1 Introduction

Although the utilization of 2D FoVs could be an alternative, its static nature might cause an obvious deterioration in user experience, especially when the user keeps moving. Inserting more view points could partly solve the problem but when user moves around, 2D FoVs still suffer from its insufficiency in providing 360-degree ARO experience. Hence, we further bring into the video stream providing continuous view during user's moving process.

Multi-view streaming applications have recently attracted increased popularity especially in virtual reality (VR) and augmented reality (AR) fields. In these types of applications, each viewpoint is a sequence of frames captured by a given camera and is usually treated as a single view stream. Through transmitting different high-quality view streams to the VR/AR equipment, the user could enjoy a more realistic perspective of 3D scenes [120, 5]. However, multi-view applications are still confronted with limited resources in the network (e.g., the terminal, the wireless channel and cloud resources), and hence, it is not recommended to store and transmit all view streams at a very high quality [123]. Thus, it is more efficient in terms of network resource utilization to endure a subset of view streams or a lower quality level during the service optimization process [120, 13]. In this chapter, a fixed given quality level is applied for the different view streams during modeling, and only popular ones are proactively cached and transmitted for multi-view streaming AR applications. To illustrate the use case, a toy example of the proposed work flow of multi-view streaming AR applications is detailed by Figure 5.1. When receiving a set of frames with recognized AROs, the target server searches the matched view stream embedded with the requested ARO in its local cache. Clearly, a series of view streams might contain the same requested ARO but only a popular subset (or in some instances, depending on available resources, the most popular one) is proactively cached and returned to the user according to the given network constraints.

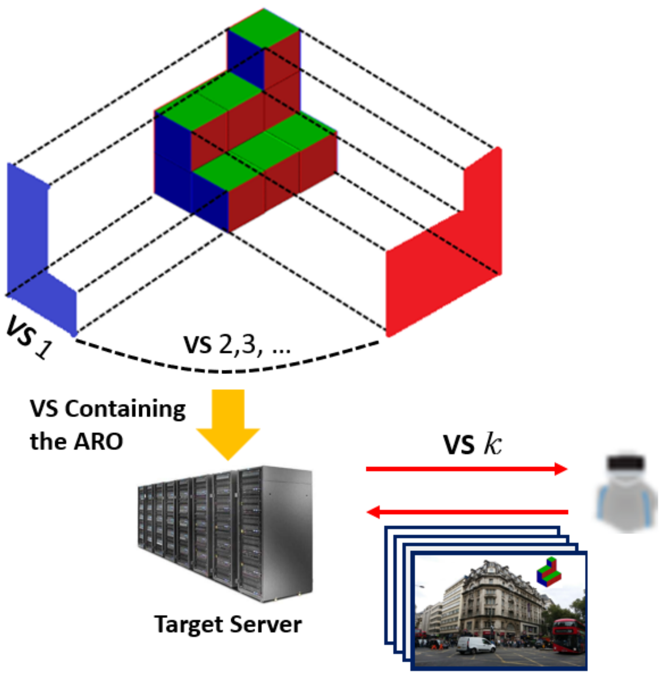
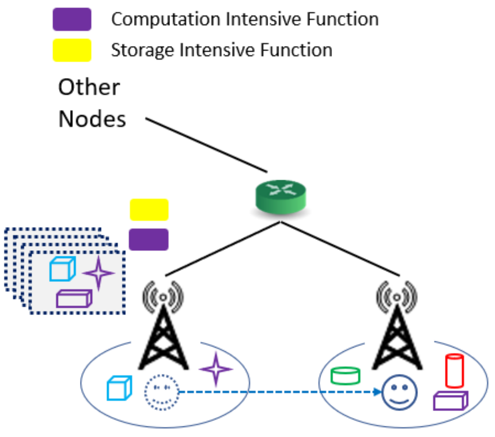


Fig. 5.1 An example of multi-view streaming AR application.

Essentially, multi-view streaming MAR applications can be decomposed into two main types of functions that can operate in tandem (i.e., as a chain), known as storage intensive functions and computing intensive functions [42]. Service decomposition is also suitable for such applications because they are susceptible to storage and computing resources, especially when utilized in a high mobility scenario [42]. Figure 5.2 presents toy examples of whether considering service decomposition and multi-view streaming for MAR. In case (a), mobility is not considered, leading to a sub-optimal EC allocation. However, case (b) depicts the proposed mobility-aware allocation that considers also popular view streams and AROs. As shown in Figure 5.2, caching all AROs that are embedded in a multi-view streaming AR application at a single EC close to the user's initial location might be suitable only when there is no mobility event and there are sufficient EC resources. However, when taking mobility and congestion episodes, the different view streams and the number of AROs into account, it is more flexible and efficient to apply service decomposition and cache popular view streams and AROs into different ECs. Clearly, service decomposition allows a flexible assignment that fits better in a congested network with limited cache and computing resources. As shown, for example, by view stream 4 in case (b), if the target EC has limited resources, then for the selected view stream, it is recommended to cache only the most popular AROs. As a result, users request only popular view streams, such as for example view stream 4, as shown in the figure. It can be accessed with very low latency, since it can be cached in an EC at the user proximity. Hence, by exploring the popularity of the view streams and the AROs, a more efficient allocation can be implemented without affecting the overall user experience, since caching is taking place by considering the most popular service composition for the end users. Hereafter, the view streams under consideration are assumed to be non real time, so that they could be cached at suitable EC locations during the service period [79]. It is also worth pointing out that in multi-view streaming AR applications, it is possible to identify popular view streams and associated AROs through historical data [42, 115]. In that case, view streams and associated AROs per stream could be optimally anchored and decomposed in different EC by exploring, at the same time, the mobility pattern of the end users. In [116], a federated learning-based scheme is proposed to predict view ports based on the view history, and then, a stochastic game is formulated to optimize virtual reality video streaming through caching high or low-quality tiles of video according to given conditions. Similarly, considering latency and user preference, we aim to optimize the allocation of decomposed functions with popular view streams and embedded AROs on different target cloud servers.



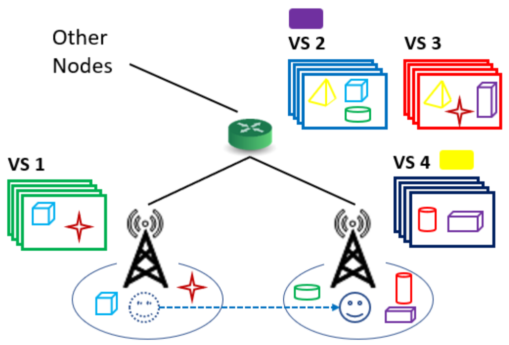


Fig. 5.2 Illustrative toy examples where in case (**a**) mobility is not considered and (**b**) where mobility of the end user is taken into account for service decomposition and the different multi view streams on proactive resource allocation of MAR functions.

A Mixed Integer Linear Problem (MILP) is formulated and solved to obtain optimal solutions in this chapter. However, due to the curse of dimensionality, it is not ideal for a large network with dynamic changes. Machine learning (ML)-based techniques can be deemed as an effective method for AR applications as they are able to learn from given data and provide reliable predictions [10, 71]. A well-known ML technique, the long short-term memory (LSTM) neural network, is utilized in this chapter to learn from optimal solutions. In this chapter, we consider a typical Rayleigh fading channel to model the wireless link of a 5G access point, and the potential of multi-view streaming and service decomposition is explored through a nominal LSTM network.

In this chapter, multiple view streams with embedded AROs are considered together with user mobility and MAR service decomposition to construct a joint optimization problem. The proposed framework seeks a balance between service latency and user preference of view streams and embedded AROs. Due to the combinatorial nature of the problem, the proposed optimization problem cannot be deemed as suitable for real-time decision making. Thus, we further propose an LSTM-based scheme to learn from optimal decision making in an offline manner to provide high-quality predictions during inference in an efficient manner suitable for real-time operation.

5.2 System Model

5.2.1 Multi-view streaming AR

As shown in previous chapters, the set $\mathbb{M} = \{1, 2, \ldots, M\}$ is still the available ECs in the wireless network with requests $r \in \mathbb{R}$ (each user makes a single request). As already mentioned in previous sections, a set of AROs is assumed to be embedded across the different non real-time view streams. To this end, we first define a set $\mathbb{N} = \{1, 2, \ldots, N\}$ to represent the set of available AROs. The video content available to each user has multiple streams, and we define them as a set $\mathbb{S}_r = \{1, 2, \ldots, S\}$. Then, the decision variable for proactively caching a view stream s at the EC $j \in \mathbb{M}$ is denoted as p_{sj}. The subset $\mathbb{L}_{rs}$ represents the target AROs required by the user r in view stream $s \in \mathbb{S}_r$ and the size of each target ARO $l \in \mathbb{L}_{rs}$ is denoted as O_l. Finally, the decision variable for proactively caching an ARO required by a request r is denoted as h^s_{rl}. More specifically, p_{sj} and h^s_{rl} can be written as follows,

$$p_{sj} = \begin{cases} 1, \text{if view stream } s \text{ is cached at node } j, \\ 0, \text{otherwise.} \end{cases} \tag{5.1}$$

$$h^s_{rl} = \begin{cases} 1, \text{if ARO } l \text{ required by request } r \text{ embedded} \\ \quad \text{in view stream } s \text{ is cached,} \\ 0, \text{otherwise.} \end{cases} \tag{5.2}$$

in addition to the above, the following constraints should also be satisfied,

$$\sum_{r \in \mathbf{R}} h^s_{rl} \leqslant 1, \; \forall j \in \mathbf{M}, \; \forall s \in \mathbf{S_r}, \; \forall l \in \mathbf{L_{rs}} \tag{5.3}$$

$$\sum_{s \in \mathbf{S_r}} \sum_{l \in \mathbf{L_{rs}}} h^s_{rl} \geqslant 1, \; \forall r \in \mathbf{R} \tag{5.4}$$

$$\sum_{j \in \mathbf{M}} p_{sj} \geqslant h^s_{rl}, \; \forall r \in \mathbf{R}, \; \forall s \in \mathbf{S_r}, \; \forall l \in \mathbf{L_{rs}} \tag{5.5}$$

$$h^s_{rl} \leqslant h^s_{rl} \sum_{j \in \mathbf{M}} p_{sj}, \; \forall r \in \mathbf{R}, \; \forall s \in \mathbf{S_r}, \; \forall l \in \mathbf{L_{rs}} \tag{5.6}$$

The constraints in (5.3) ensure that each ARO can be cached at most once in a view stream. The constraints in (5.4) ensure that at least one view stream and an ARO is required to compose a valid request. The constraints in (5.5) guarantee that the ARO must be allocated to a view stream first before deciding to proactively cache it, and the constraints in (5.6) certify that any ARO planned to be stored in this view stream should not be cached when deciding not to proactively cache a view stream at all. Constraints (5.5) and (5.6) together ensure that either the view stream or ARO cannot be handled alone during the formulation.

As alluded earlier, due to the resource constraints in the network, in case of congestion episodes, an optimized subset of view streams and embedded AROs need to be proactively cached at suitable ECs. Thus, we bring in the user preference for both view streams and AROs. The preference of view streams and embedded AROs are measured by their requesting probability, which enables the proposed scheme to select the more popular ones. The probability of selecting a view stream s is denoted as t_s and the probability of requiring an ARO l is denoted as t_l. Such probabilities are pre-defined and could be achieved by exploring historical data; hereafter we assume that we have access to such information. Thus, we further define $\mathbb{T_l}$ and $\mathbb{T_s}$, respectively to represent two descending probability vectors of AROs and view streams. The probability t_l and t_s inside can be generated through,

$$t_l = \begin{cases} 1, \text{ if } l = 1, \\ \text{rand}\left(t_{l-1}\right), \text{ if } l > 1. \end{cases} \tag{5.7}$$

$$t_s = \begin{cases} 1, \text{ if } s = 1, \\ \text{rand}\left(t_{s-1}\right), \text{ if } s > 1. \end{cases} \tag{5.8}$$

where the function rand(x) generates a random number within $[0,x]$. Then, each element should be normalized through,

$$\frac{t_l}{\sum_{l \in \mathbf{L_{rs}}} t_l}, \frac{t_s}{\sum_{s \in \mathbf{S_r}} t_s} \tag{5.9}$$

In a similar manner with [42], computational-intensive and storage-intensive MAR functionalities are defined as η and ρ, respectively. In addition, the corresponding execution location for a functionality is denoted as x_{ri} and y_{ri}, respectively [42]. The starting location of the request r is the access router where this user is initially connected to; this initial location is defined as $f(r)$. A user moves to a destination $k \in \mathbb{K}$ in the case of a mobility event (i.e., changing the point of attachment). In this chapter, and without loss of generality, only adjacent access routers can be regarded as allowable destinations in the mobility event. Hence, $u_{f(r)k} \in [0,1]$ is defined to represent the probability of a user moving from the initial location to an allowable destination, where adjacent servers $\{f(r),k\} \subset \mathbb{M}$. Such probabilities can also be learned from historical data, which are readily available from mobile operators. The size of uploaded frames with target AROs used in the Object Detection function is denoted as $F_{\eta r}$ and the size of pointers used for identifying target AROs is denoted as $F_{\rho r}$ [42, 115]. During the matching process, the target AROs are possibly not pre-cached in the local cache and such case is known as a “cache miss” (otherwise there is a “cache hit”). Whenever confronted with a cache miss, the request is redirected to a core cloud deeper in the network and suffers from an extra latency D as penalty.

At first, a joint optimization scheme considering the balance between the service delay and the preference of view streams and embedded AROs is designed to track the MAR requests in the EC-supported network. The cache hit/miss is captured by the decision variable z_{rj} and can be written as follows,

$$z_{rj} = \begin{cases} 1, \text{if} \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} p_{sj} h_{rl}^{s} \geqslant L_{rs}, \\ 0, \text{otherwise.} \end{cases} \tag{5.10}$$

In [42], constraints (10b) and (10d) reveal the cache limitation and the cache hit/miss relation. In this chapter, they should be rewritten as follows,

$$\sum_{r \in \mathbf{R}} \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} p_{sj} h_{rl}^{s} O_l \leq \Theta_j, \forall j \in \mathbf{M} \tag{5.11}$$

$$\sum_{l \in \mathbf{N}} \sum_{s \in \mathbf{S_r}} h_{rl}^{s} + \varepsilon \leq L_{rs} + U(1 - q_{rj}) \, \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{5.12}$$

where Θ_j is the cache capacity at node j. In (5.12), to rewrite the either-or constraint that $\sum_{l \in \mathbf{N}} \sum_{s \in \mathbf{S_r}} h_{rl}^{s} < L_{rs}$ or $z_{rj} = 1$, we define ε as a small tolerance value, U as a large arbitrary number and q_{rj} as a new decision variable satisfying $1 - q_{rj} = z_{rj}$ [42]. Requiring more pre-cached view streams and embedded AROs in a request naturally lead to an extra burden for the matching function. More specifically, the processing delay of the matching function can be written as,

$$W_{rj} = \frac{\omega(F_{\rho r} + \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} p_{sj} h_{rl}^{s} O_l)}{f_V^j} \tag{5.13}$$

where ω is the computation load, f_V^j is the virtual CPU frequency and $F_{\rho r}$ are the size of the uploaded pointers [42, 115]. When finding the target ARO during matching, other view streams containing it should also be transferred to the user. Hence, the delay of sending results back to the user can be written as follows,

$$\sum_{s \in \mathbf{S_r}} \sum_{i \in \mathbf{M}} \sum_{j \in \mathbf{M}} C_{ij} y_{ri} p_{sj} + \sum_{s \in \mathbf{S_r}} \sum_{r \in \mathbf{R}} \sum_{j \in \mathbf{M}} \sum_{k \in \mathbf{K}} C_{jk} u_{f(r)k} p_{sj} \tag{5.14}$$

where $u_{f(r)k}$ is the moving probability from the initial location $f(r)$ to a potential destination k. In previous expressions, the product of decision variables $p_{sj} h_{rl}^{s}$ and $p_{sj} y_{rj}$ exists and creates a non-linearity. In addition, note that when executing the matching function at the location j ($W_{rj} y_{rj}$), the product of decision variables $p_{sj} h_{rl}^{s} y_{rj}$ also appears. To linearize the expressions above, so that to utilize linear integer programming solution methodologies, new auxiliary decision variables are brought in. A new decision variable

α_{rsj} is introduced as $\alpha_{rsj} = p_{sj}y_{rj}$ and the constraints should be added as follows,

$$\begin{aligned} &\alpha_{rsj} \leqslant p_{sj}, \\ &\alpha_{rsj} \leqslant y_{rj}, \\ &\alpha_{rsj} \geqslant p_{sj} + y_{rj} - 1 \end{aligned} \tag{5.15}$$

similarly, a new decision variable β_{rslj} is introduced as $\beta_{rslj} = p_{sj}h^{s}_{rl}$ and the constraints should be added as follows,

$$\begin{aligned} &\beta_{rslj} \leqslant p_{sj}, \\ &\beta_{rslj} \leqslant h^{s}_{rl}, \\ &\beta_{rslj} \geqslant p_{sj} + h^{s}_{rl} - 1 \end{aligned} \tag{5.16}$$

the aforementioned constraint (5.6) is affected and should be rewritten as follows,

$$h^{s}_{rl} \leqslant \sum_{j \in \mathbf{M}} \beta_{rslj}, \ \forall r \in \mathbf{R}, \ \forall s \in \mathbf{S_r}, \ \forall l \in \mathbf{L_{rs}} \tag{5.17}$$

In addition, note that p_{sj} is a binary decision variable and causes $p_{sj} = p^{2}_{sj}$. Thus, we have $p_{sj}h^{s}_{rl}y_{rj} = \alpha_{rsj}\beta_{rslj}$. A new decision variable λ_{rslj} is introduced as $\lambda_{rslj} = \alpha_{rsj}\beta_{rslj}$ and the following constraints should be added as follows,

$$\begin{aligned} &\lambda_{rslj} \leqslant \alpha_{rsj}, \\ &\lambda_{rslj} \leqslant \beta_{rslj}, \\ &\lambda_{rslj} \geqslant \alpha_{rsj} + \beta_{rslj} - 1 \end{aligned} \tag{5.18}$$

5.2.2 Wireless Channel Modeling and Achievable Data Rate

In previous chapters, we considered a nominal average wireless transmission rate for each user. However, this neglects the influence of the wireless channel such as for example link gain variation due to the user distance (Euclidean) to the access router. To capture a more detailed use case scenario, we further consider user location and apply a well used wireless channel model (a fast-fading channel following Rayleigh distribution) to achieve a more representative achievable data rate for each user.

According to Shannon formula, the achievable wireless data rate for the user in a 5G access point is $B_j \log_2(1 + \gamma_{rj})$, where B_j is the bandwidth allocated to the user's resource block and γ_{rj} is the Signal to Interference plus Noise Ratio (SINR) of the user r at the node j. We denote P_j as the current resource block power, H_{rj} as the channel gain, N_j as the noise, a as the path loss exponent and d_{rj} as the Euclidean Distance between the user and the access router in the cell. Furthermore, we utilize a nominal Rayleigh fading channel to model the channel between a 5G access point and the users [31]. In this case, the channel gain H_{rj} can be written as follows [20],

$$H_{rj} = \sqrt{\frac{1}{2}}(t + tJ) \tag{5.19}$$

where $J^2 = -1$, t is a random number following the standard normal distribution. Then, the SINR γ_{rj} in a 5G wireless network can be written as follows [20, 105],

$$\gamma_{rj} = \frac{P_j H_{rj}^2 d_{rj}^{-a}}{N_j + \sum_{i \in \mathbf{M}, i \neq j} P_i H_{ri}^2 d_{ri}^{-a}} \tag{5.20}$$

Thus, the wireless transmission delay in a mobility event can be written as follows,

$$\begin{aligned}&\sum_{r \in \mathbf{R}} \frac{F\eta r + \sum_{s \in \mathbf{S_r}} \sum_{l \in \mathbf{L_{rs}}} h_{rl}^s O_l}{B_{f(r)} \log_2(1 + \gamma_{rf(r)})} + \\ &\sum_{r \in \mathbf{R}} \sum_{k \in \mathbf{K}} u_{f(r)k} \frac{F\eta r + \sum_{s \in \mathbf{S_r}} \sum_{l \in \mathbf{L_{rs}}} h_{rl}^s O_l}{B_k \log_2(1 + \gamma_{rk})}\end{aligned} \tag{5.21}$$

5.2.3 Latency and Preference

Based on the above derivations and in line with [42], the service latency can be written as follows,

$$\begin{aligned}L = &(5.21) + \sum_{r \in \mathbf{R}} \sum_{i \in \mathbf{M}} (C_{f(r)i} + V_{ri}) x_{ri} + \\ &\sum_{r \in \mathbf{R}} \sum_{i \in \mathbf{M}} \sum_{j \in \mathbf{M}} \Big(\frac{\omega(F_{\rho r} y_{rj} + \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} \lambda_{rslj} O_l)}{f_V^j} + \\ &C_{ij} \xi_{rij} + C_{jf(r)} y_{rj} + \psi_{rj} D \Big) + \\ &\sum_{s \in \mathbf{S_r}} \sum_{i \in \mathbf{M}} \sum_{j \in \mathbf{M}} C_{ij} \alpha_{rsj} + \sum_{s \in \mathbf{S_r}} \sum_{r \in \mathbf{R}} \sum_{j \in \mathbf{M}} \sum_{k \in \mathbf{K}} C_{jk} u_{f(r)k} p_{sj}\end{aligned} \tag{5.22}$$

L_{max} here denotes the maximum allowed service latency. Thus, we have,

$$\frac{L}{L_{max}} \in [0, 1] \tag{5.23}$$

The overall preference of view streams and embedded AROs is measured by the summation of probabilities of the chosen view streams and AROs. It is denoted as Q and can be written as follows,

$$\begin{aligned}Q &= \sum_{s \in \mathbf{S_r}} \sum_{j \in \mathbf{M}} p_{sj} t_s + \sum_{r \in \mathbf{R}} \sum_{s \in \mathbf{S_r}} \sum_{j \in \mathbf{M}} \sum_{l \in \mathbf{L_{rs}}} p_{sj} h_{rl}^s t_l \\ &= \sum_{s \in \mathbf{S_r}} \sum_{j \in \mathbf{M}} p_{sj} t_s + \sum_{r \in \mathbf{R}} \sum_{s \in \mathbf{S_r}} \sum_{j \in \mathbf{M}} \sum_{l \in \mathbf{L_{rs}}} \beta_{rslj} t_l\end{aligned} \tag{5.24}$$

similarly, Q_{max} here denotes the maximum allowable preference of view streams and embedded AROs. Hence, for the preference of view streams and embedded AROs, we also have,

$$\frac{Q}{Q_{max}} \in [0,1] \tag{5.25}$$

The weight parameter is denoted as $\mu \in [0,1]$, and the joint optimization problem can eventually be written as follows,

$$min\ \mu \frac{L}{L_{max}} - (1-\mu)\frac{Q}{Q_{max}} \tag{5.26a}$$

$$\text{s.t. } z_{rj} = 1 - q_{rj},\ \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{5.26b}$$

$$\sum_{r\in\mathbf{R}} (x_{rj} + y_{rj}) \leq \Delta_j, \forall j \in \mathbf{M} \tag{5.26c}$$

$$\sum_{j\in\mathbf{M}} x_{rj} = 1, \forall r \in \mathbf{R} \tag{5.26d}$$

$$\sum_{j\in\mathbf{M}} y_{rj} = 1, \forall r \in \mathbf{R} \tag{5.26e}$$

$$\xi_{rij} \leq x_{ri},\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \tag{5.26f}$$

$$\xi_{rij} \leq y_{rj},\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \tag{5.26g}$$

$$\xi_{rij} \geq x_{ri} + y_{rj} - 1,\ \forall r \in \mathbf{R}, i, j \in \mathbf{M} \tag{5.26h}$$

$$\psi_{rj} \leq z_{rj},\ \forall r \in \mathbf{R}, j \in \mathbf{M} \tag{5.26i}$$

$$\psi_{rj} \leq y_{rj},\ \forall r \in \mathbf{R}, j \in \mathbf{M} \tag{5.26j}$$

$$\psi_{rj} \geq z_{rj} + y_{rj} - 1,\ \forall r \in \mathbf{R}, j \in \mathbf{M} \tag{5.26k}$$

$$\begin{gathered} x_{rj}, y_{rj}, p_{sj}, h^s_{rl}, z_{rj}, q_j \in \{0,1\}, \\ \alpha_{rsj}, \beta_{rslj}, \lambda_{rslj}, \psi_{rj}, \xi_{rij} \in \{0,1\}, \\ \forall r \in \mathbf{R}, j \in \mathbf{M}, l \in \mathbf{L_{rs}}, s \in \mathbf{S_r} \end{gathered} \tag{5.26l}$$

(5.3), (5.4), (5.5), (5.11), (5.12), (5.15), (5.16), (5.17), (5.18)

As mentioned earlier, the constraint (5.26b) together with constraints (5.3) to (5.5), (5.11), (5.12) and (5.17) reveal the relation between pre-caching decisions and the cache miss/hit for each request [42]. The constraint (5.26c) is the virtual machine limitation while (5.26d) and(5.26e) guarantee the once execution of each function of a request at a single server as explained in [42]. The constraints (5.15), (5.16), (5.18) and (5.26f) to (5.26k) are auxiliary and required to solve the product of decision variables for linearization. Key variables defined in the formulation are shown with their description in the following Table 5.1.

Table 5.1 Notation

Parameter	Description
$\mathbf{N}$	Set of all AROs
$\mathbf{M}$	Set of Edge Clouds
$\mathbf{R}$	Set of User Requests
$\mathbf{S_r}$	Set of View Streams
$\mathbf{K}$	Set of User Destinations
$\mathbf{L_{rs}}$	Set of target AROs for request r in corresponding view stream s
$\mathbf{T_l}, \mathbf{T_s}$	Accessing probability vectors of AROs and view streams
η, ρ	Two types of AR functions (CPU, Cache)
O_l	Size of ARO l
h^s_{rl}	0/1 var.: if ARO l in view stream s for the request r is cached
p_{sj}	0/1 var.: if view stream s is cached at node j
x_{rj}, y_{rj}	0/1 var.: if function η or ρ for r is set at EC j
z_{rj}	0/1 var.: if a cache hit is spotted for request r
C_{ij}	Communication delay between nodes i and j
$F_{\eta r}, F_{\rho r}$	Input size for functions η, ρ in request r
H_{rj}	Channel gain for request r at node j
γ_{rj}	SINR for request r at node j
B_j, P_j, N_j	Resource block bandwidth, power and noise at node j
α	path loss exponent
d_{rj}	Euclidean distance between user r to node j
$f(r)$	Initial access point for request r
D	Cache miss penalty
$u_{f(r)k}$	Moving probability from initial location $f(r)$ to destination k
ω, f^j_V	Computation load and CPU availability at EC j
V_{rj}, W_{rj}	Processing delay for request r for function η, ρ at EC j
L, Q	Overall Latency and User Preference
Θ_j	Cache capacity at EC j

5.2.4 Heuristic Algorithm using LSTM

The aforementioned optimization problem could provide a series of optimal solutions for learning. These optimal solutions are grouped and reshaped into a route matrix and then separated for training and testing purposes. A nominal LSTM network developed in previous Section 3.3 is then applied here to explore the potential of multi-view streaming and improve the computing efficiency. Its state changing and gate controlling also follow the nominal LSTM architecture. As shown in Figure 5.3, we feed user initial locations and destinations into the LSTM network as input and optimized locations of servers anchoring MAR functionalities as output. All possible different results can be calculated as selecting any two execution locations from the set $\mathbb{M}$ with order, which is M^2. Then, the LSTM-based model classifies and creates a mapping between the two sets. A Feasibility Check Stage is required afterwards to guarantee that predictions from the LSTM network could still satisfy the given constraints in the original network. If violating a constraint, such as exceeding an EC's available cache size for example, the remaining predicted requests served by this EC will be redirected to an available neighbor EC as a backup choice. However, if both ECs are fully occupied, such remaining requests have to be transmitted to a further core network cloud and hence suffer from an extra penalty in latency. Figure 5.4 shows a toy example of applying the LSTM network in our case. The LSTM network treats vectors with low appearance frequency such as [1,3]/[2,4] and [3,1]/[2,2] as unimportant ones and forgets them during iterations. When several types of optimal solutions (e.g., [2,5] and [4,5]) for the same user route vector (e.g., [1,5]) share a similar level of appearance frequency, it is possible that the LSTM network generates a prediction between given solutions (e.g., [3,5]). The validation accuracy might still be acceptable, but in the Feasibility Check Stage, an activated neighbor EC with enough resources will substitute the current one (e.g., [2,5] replaces [3,5])

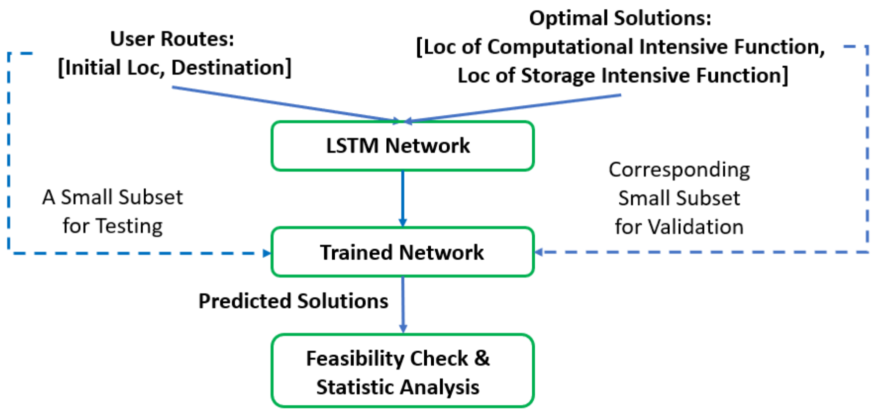


Fig. 5.3 Working Process of the LSTM-based Model.

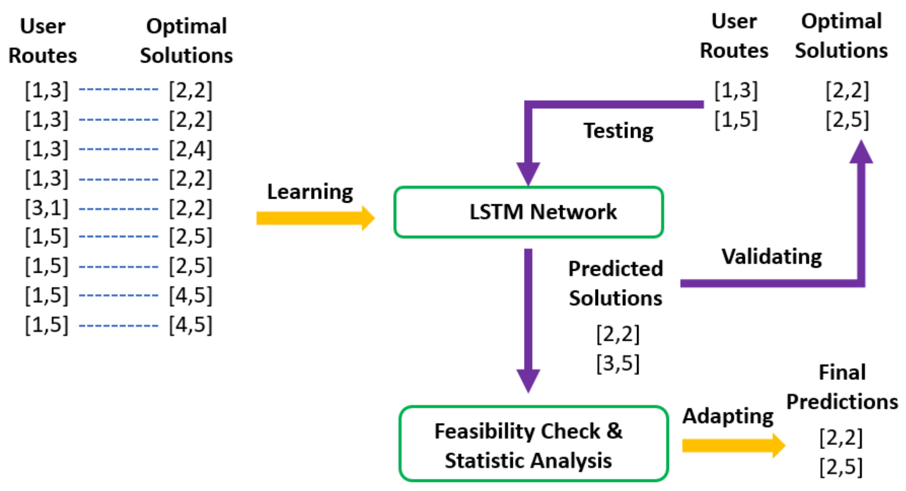


Fig. 5.4 A Toy Example of the LSTM-based Model.

5.3 Numerical Investigations

Hereafter, the effectiveness of the proposed optimization scheme (noted as Optim) and the heuristic algorithm using the LSTM network (noted as LSTM) are investigated via a wide set of simulations and compared with a number of nominal/baseline schemes.

Parameterization

The same as in [42], a typical tree-like network topology as shown by Figure 3.3 is applied with 20 ECs in total with four to eight ECs being activated for the current MAR service, and 20 to 40 requests are sent by MAR devices. The remaining available resources allocated for MAR support of an EC are assumed to be CPUs with four to eight cores and 100 to 400 MBytes of cache memory [42]. Each request requires a single free unit for each service function (for example, a VM) [67]. There are 10 to 14 available VMs in each EC, with equal splitting of the available CPU resources [42]. Up to four different view streams of the video content per user can be cached and are similar to each other. All target AROs must be embedded within the corresponding view stream before being streamed to an end user based on a matched result. The user could trigger the MAR service by a certain behavior (e.g., clicking an AR label), and pointers such as a name or index are transferred to the EC to identify the ARO [115]. The size of such pointers used for matching are only a few bytes in size, and hence, their transmission and processing latency is neglected in the simulations. The bandwidth of a nominal 5G access point is set to 20 MHz, its transmission power is assumed to be 20 dBm with the maximum of 100 resource blocks, and we assume, without loss of generality, each user can utilize only one resource block [59, 63, 19]. The noise power is 10^{-11} W and the path loss exponent is 4 [59]. The location of the users is randomly generated as well as the potential destinations. Furthermore, we assume that each cell has a radius as 250 m in the 5G wireless network [59]. For the LSTM network, 90% of 300 route vectors with corresponding optimal decisions are utilized for training, while the remaining 10% are used for testing and validation. The initial learning rate is set to 0.005, the maximum number of epochs is 160 and the dropout probability is 5% to avoid over-fitting. Key simulation parameters are summarized in the following Table 5.2.

Table 5.2 Simulation parameters

Parameter	**Value**
Number of available ECs	$[4,8]$
Number of available VMs per EC (EC Capacity)	$[10,14]$
Number of requests	$[20,40]$
Number of view streams per user	4
AR object size	$(0,20]$ MByte
Total moving probability	$[0,1]$
Cell Radius	250m
Remained Cache Capacity per EC	$[100,400]$ MByte
CPU frequency	$[2,4]$ GHz
CPU cores	$[4,8]$
CPU portion per VM	$0.125 - 0.25$
Computation load	10 cycles/bit
Bandwidth of Access Router	2 GHz
Power of Access Router	20dBm
Path Loss Exponent	4
Noise Power	10^{-11}W
Number of Resource Blocks	100
Average hop's latency	2 ms
Cache miss penalty	25 ms
Number of user route vectors / optimal decisions	300
Initial LSTM learning rate	0.005
Maximum number of epochs	160
LSTM Dropout probability	5%

To appreciate the quality of the proposed set of solutions, baseline schemes for caching MAR content are also included in the investigations, such as RandS, CFS and UTIL from previous chapters. Note that all such baseline schemes assign view streams and embedded AROs according to the above mentioned policies and hence neglect the user mobility (i.e., they are mobility oblivious). Furthermore, they will also suffer from a penalty cost if their target ECs (including backup EC choices) are all fully occupied.

5.3.1 Simulation Results

According to Figure 5.5, an increasing weight (μ) leads to a dropping overall delay as expected. A larger weight means that the proposed Optim scheme targets a solution with less latency and is willing to sacrifice some preference for this in terms of importance. As a result, the Optim scheme tends to proactively cache fewer view streams with fewer embedded AROs, and such a decision naturally results in a reduced overall delay in transmission and computation. As shown by Table 5.3, it is interesting to point out that when the emphasis is placed solely in delay reduction ($\mu = 1$) and without considering user mobility, the proposed Optim scheme closely relates to the other greedy baseline schemes. This means those schemes commonly tend to anchor view streams with embedded AROs as close as possible to the user. However, the Optim scheme is still superior to others and is around 15.5% better than the CFS scheme in this case because its service decomposition and separation of view streams enables flexible allocation and hence avoids overloading. In addition, it is worth pointing out that solutions offered from other greedy schemes sometimes provide non-feasible solutions due to constraints violations. Thus, the Optim scheme significantly outperforms other schemes especially in a high mobility scenario and/or in the cases where changes of the point of attachment take place during the service time. The observed gains between the proposed Optim scheme and other baseline schemes also increase with overall delay when continuously seeking a larger preference and caching a higher number of view streams and AROs. Because the UTIL scheme prevents further transmitted requests when the activated ECs reach the 80% occupation limit, it becomes the most sensitive scheme to the weight μ and hence its performance is heavily affected in a congested network. Figure 5.6 shows the relation between preference and weight (μ) and further reveals the advantage of the proposed Optim scheme in preference. Noticing that even in the extreme case when $\mu = 1$, the proposed Optim scheme still achieves approximately 59.6% preference due to its optimized selection of the most preferred view streams with the most popular AROs set inside. This means the proposed Optim scheme minimizes the cost of weight by exploring preference when seeking to reduce service latency

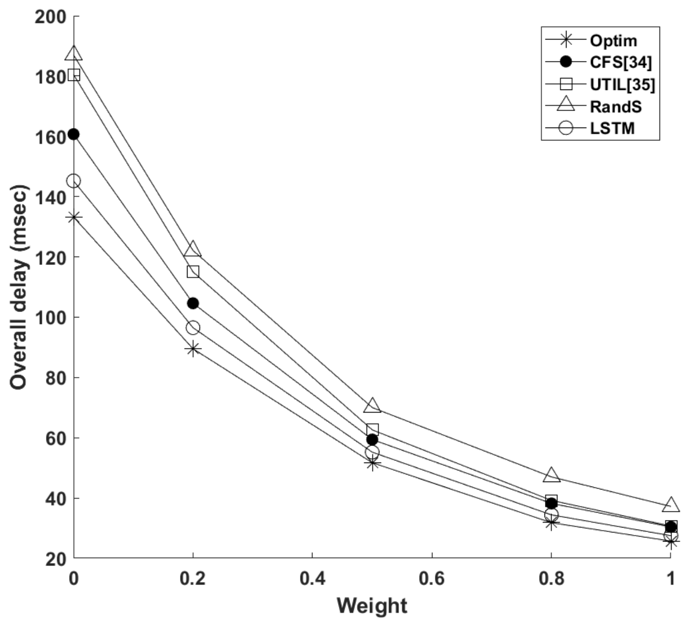


Fig. 5.5 Service delay with weight μ (6 EC, 30 requests, EC capacity is 14 and total mobility probability is 1).

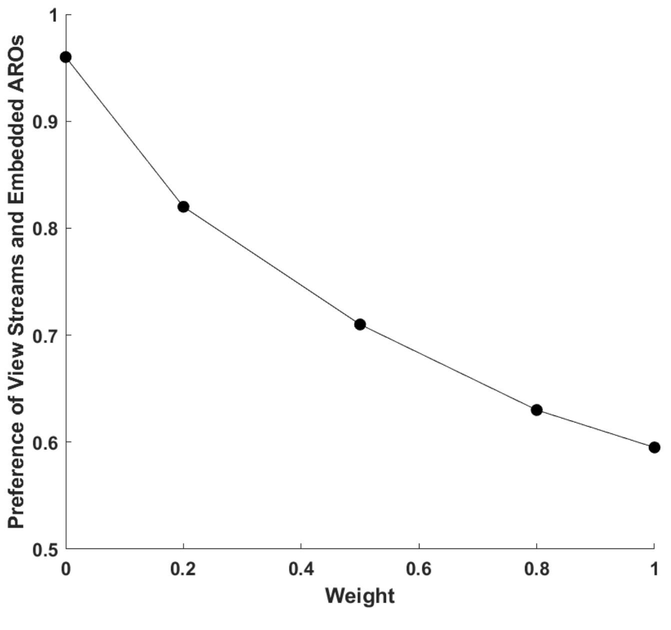


Fig. 5.6 Preference of View Streams and Embedded AROs with Weight μ (6 EC, 30 Requests, EC Capacity is 14 and total mobility probability is 1)

According to Figure 5.7, the service delay increases when the network congestion levels increase due to a higher number of requests from the users. The LSTM scheme shows a 5.6% to 9.8% optimality gap compared to the Optim scheme and approaches the Optim scheme better in a non-congested condition. The above aspect can also be seen

Table 5.3 RMSE Values & Service Delay in no mobility event
($\mu = 1$, 6 ECs, 30 requests and EC Capacity is 14)

Scheme	**Optim**	**LSTM**	**CFS**	**UTIL**	**RandS**
Delay (ms)	25.7	27.9	30.2	31.1	37.4
RMSE	-	2.4	8.5	3.6	5.8

from Figures 5.8 and 5.9. All schemes suffer from an increased delay when fewer ECs are activated or there is less available capacity per EC. However, in all use cases under investigation, the proposed schemes show much better performance in latency than other baseline schemes.

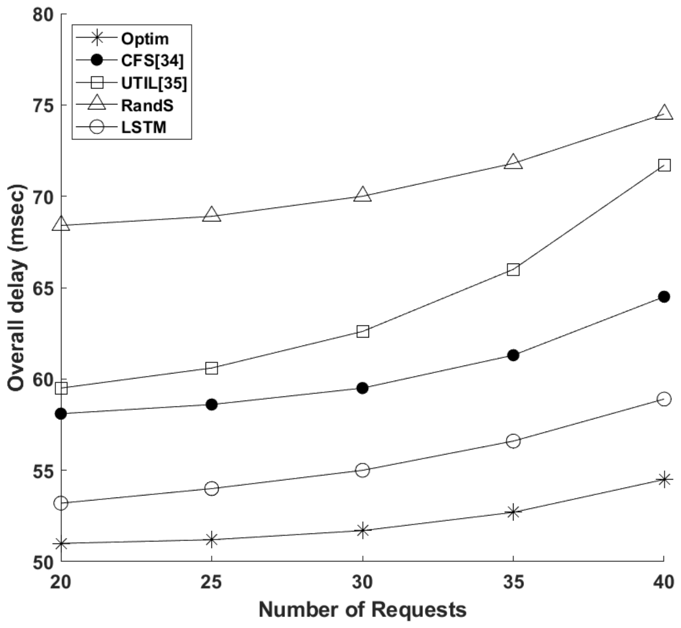


Fig. 5.7 Service Delay with Different Number of Requests (6 EC, $\mu = 0.5$, EC Capacity is 14 and Total Moving Probability is 1).

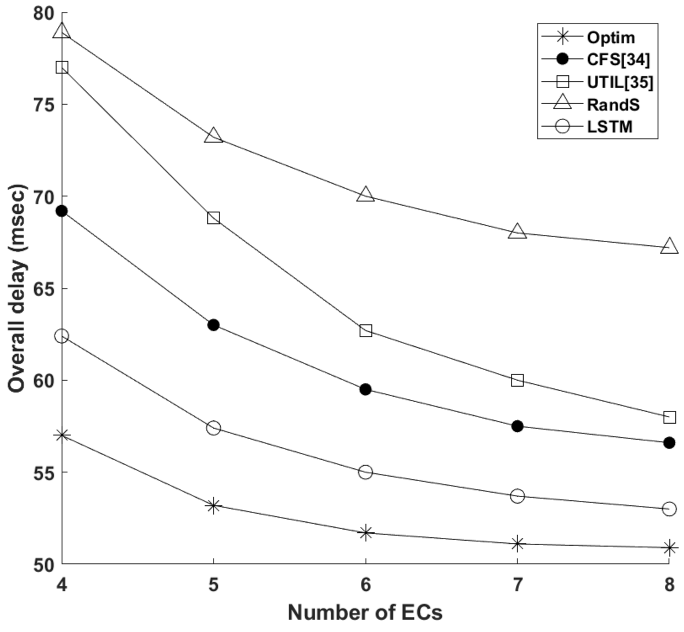


Fig. 5.8 Service Delay with Different Number of ECs (30 Requests, $\mu = 0.5$, EC Capacity is 14 and Total Moving Probability is 1).

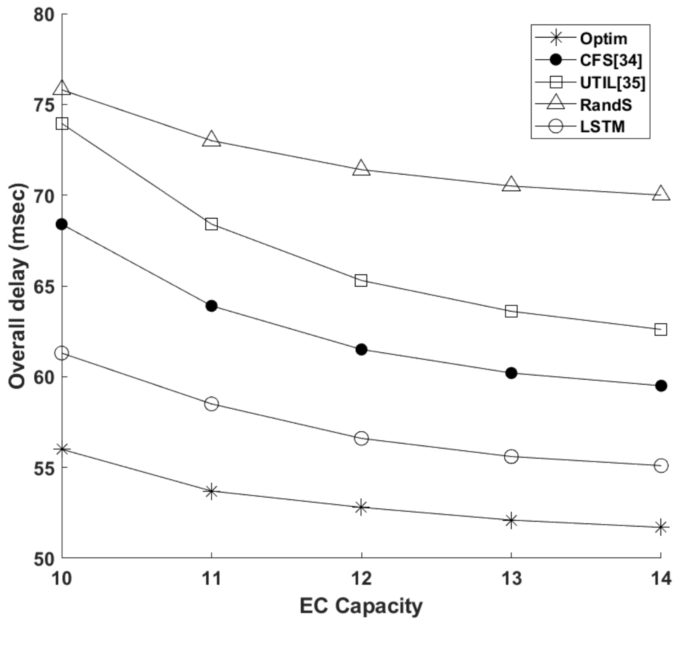


Fig. 5.9 Service Delay with Different EC Capacities (6 EC, 30 Requests, $\mu = 0.5$ and Total Moving Probability is 1).

To shed light on the performance, we calculate the *rmse* (root mean square error), δ (relative error) and r^2 (determination coefficient) for the LSTM scheme. The predictions from the LSTM network are compared with the optimal solutions, and we have $rmse < 2.5$, $\delta < 15\%$ (relative error) and $r^2 > 0.83$ (determination coefficient) in most cases. Hence, the predictions of the LSTM scheme can be regarded as reliable and closely follow the optimal solutions. Table 5.3 shows *rmse* values including the LSTM scheme and greedy baseline schemes for comparison. In simulations, topologically close servers are allocated adjacent indices, and the user mobility is limited to the adjacent server in a period of an MAR session. Hence, indices decided for each server also reveal their location information. Thus, the similarity between the predictions of the LSTM scheme and the optimal solutions reveals the LSTM scheme's competitive edge over other greedy baseline schemes. Noticing that an infeasible prediction might appear and cause a deteriorating quality of service after adjusted in the Feasibility Check Stage. However, according to Figure 5.10, the LSTM network's validation accuracy is maintained at around 86.3% and is less than 6% worse than its training accuracy, which indicates that the infeasible solutions are rare and the general learning capabilities are effective.

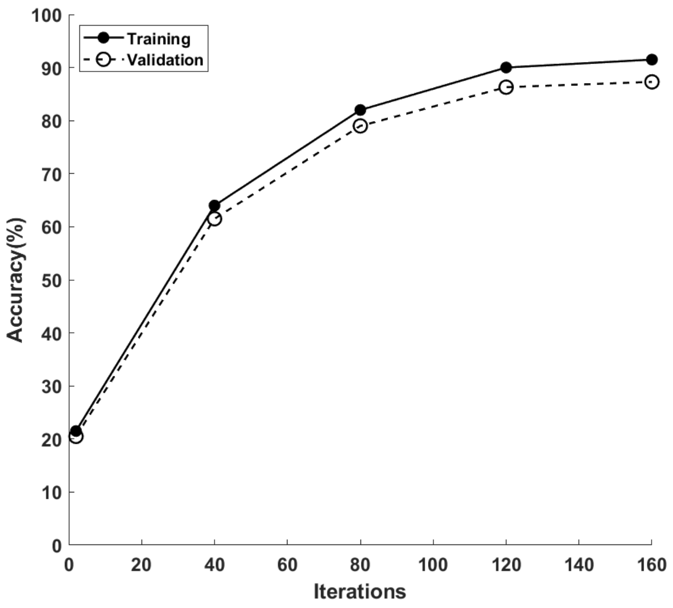


Fig. 5.10 Average Training and Validation Accuracy of the LSTM Scheme through Iterations.

Before training the LSTM scheme, it requires a long computational time to create a large number of optimal solutions (∼810 s in Fig. (5.11)) from the Optim scheme. However, this stage only needs to be executed a few times and is a purely offline task. Similarly, the network training (∼70 s in Fig. (5.11)) is also offline and does not need to repeat itself if the validation accuracy of the trained LSTM network meets the expectation. As a result, these offline stages do not have a serious detrimental effect on the general computing efficiency. As shown in Table 5.4, based on a trained LSTM network, the online decision-making stage of the LSTM scheme can generate predictions in an efficient manner for MAR sessions. Because the Optim scheme constructs and solves a complex mixed integer linear problem, it suffers from the longest processing time and is not ideal for a highly dynamic network environment. The greedy baseline schemes own the obvious advantage in average processing time, but as discussed earlier, they cannot approach optimal solutions as well as the decision-making quality offered by the LSTM scheme.

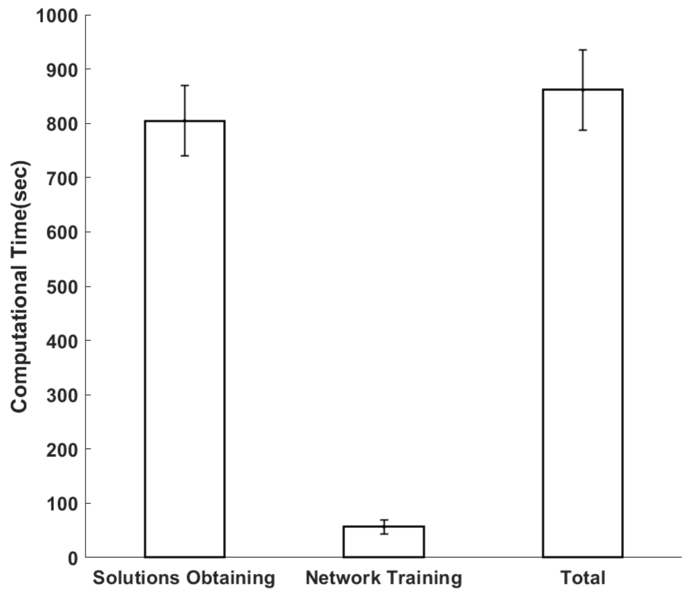


Fig. 5.11 Average Processing Time for different stages in the LSTM Scheme

Table 5.4 Average processing time of algorithms

Algorithm	Average Processing time (sec)	STD
RandS	0.696	0.284
CFS	0.731	0.295
UTIL	0.753	0.308
LSTM	1.386	0.223
Optim	168.277	15.392

*tested by PC, intel i7, 1.3GHz, 4 processors

Multi-view streaming applications with embedded augmented reality objects (AROs) are highly delay sensitive and confronted with a series of constraints in terms of storage and computing resources. In this chapter, an optimization framework is proposed to realize optimal service decomposition in an edge-cloud supported 5G and beyond wireless network aiming to balance service latency and content popularity by taking into account the inherent user mobility. In addition, a nominal long short-term memory (LSTM) neural network is proposed that is trained using optimal solutions to explore real-time decision making for supporting edge proactive caching for multi-view streaming with popular augmented reality objects. It is important to note that the proposed framework is generic enough to be applied in a plethora of mobile-augmented reality applications that are composed by multiple view streams and popular augmented objects per stream. Numerical investigations reveal that the proposed schemes outperform a number of previously defined baseline schemes. More specifically, gains up to 38% are reported in terms of reducing the delay compared to previously proposed techniques that do not explicitly take into account user mobility and the inherent decomposition of the MAR services. in the proposed framework, during the online inference stage, the LSTM-based scheme is able to efficiently provide competitive predictions approaching the optimal solutions. On the other hand, the learning and training stage of the LSTM network require longer processing times, but those phases can be executed in an offline manner where high-quality solutions from the defined optimization framework could be explored. In that way, the proposed framework is able to provide high-quality decision making in an online manner by exploring the capabilities of the LSTM neural network and the powerful integer programming formulation that is able to provide optimal decision making that can be used for training the neural network.

Although the video stream does provide a continuous experience, nowadays it is usually divided into segments for better transmission. The user might have special interest on certain segments instead of treating all segments equally like in this chapter. Hence, considering popularity of internal segments and their relation with embedded AROs could be an interesting extension.

Chapter 6

Optimal Mobility Aware Wireless Edge Cloud Support for the Metaverse

The chapter is based on [44] and [47]. Based on previous research, MAR is extended into metaverse in this chapter. The similarity could be identified during our modelling in Rayleigh channel modelling, user mobility and service decomposition. We also accept the general sharing idea of applying view streams in chapter 5 and hence enable the metaverse region sharing and multi-rendering in this chapter. However, metaverse environment owns unique features in consuming much energy during rendering and providing mixed experiences through foreground interactions and background interactions. Thus, in this chapter, we apply decomposed MAR functions following metaverse features and construct a trade-off between energy consumption and latency under the constraint of quality. Additional to all previous chapters, we further consider terminals' capacities and make a detailed comparison between whether using terminals.

6.1 Introduction

Recently, the metaverse, which could be described as an endless virtual world where users interact with their avatars, has become popular in both academic and commercial areas [7]. Augmented reality manages to combine digital information with the real world for real-time presentation, and such experiences can be regarded as a continuum ranging from assisted reality to mixed reality, according to the level of local presence [84, 85]. Mobile augmented reality (MAR) could be extended and enhanced in the wireless edge-supported metaverse with today's available technologies, such as digital twin and head-mounted display rendering [112, 28]. In addition, compared to existing MAR applications, users could seamlessly mix their experience of the metaverse and physical world through various metaverse MAR applications, such as massively multiplayer online video games and virtual concerts [112]. Users equipped with MAR devices can upload and analyze their environment through AR customization to achieve appropriate AR objects (AROs) and access the metaverse in mobile edge networks [111]. AR marketing is regarded as having potential in metaverse, and can be seen as a typical example of a more forward-looking metaverse AR application because it replaces physical products with AR holograms and enables direct foreground interactions between the customer and the digital-marketing application interface in the background environment [84, 23]. Rendering three-dimensional (3D) AROs with the background virtual environment and updating in the metaverse consume significant amounts of energy and could be highly demanding in terms of required caching and computing resources [111, 62]. Hence, such applications are delay- and energy-sensitive and face challenges in ensuring the quality of user experience and providing reliable and timely interactions with the metaverse [112, 111]. Similar to previous chapters, a nominal frame rate is assumed as 15 frames/second and the metaverse rendering takes place at every other frame ($\tilde{1}$33.2ms interval) [24][75][72].

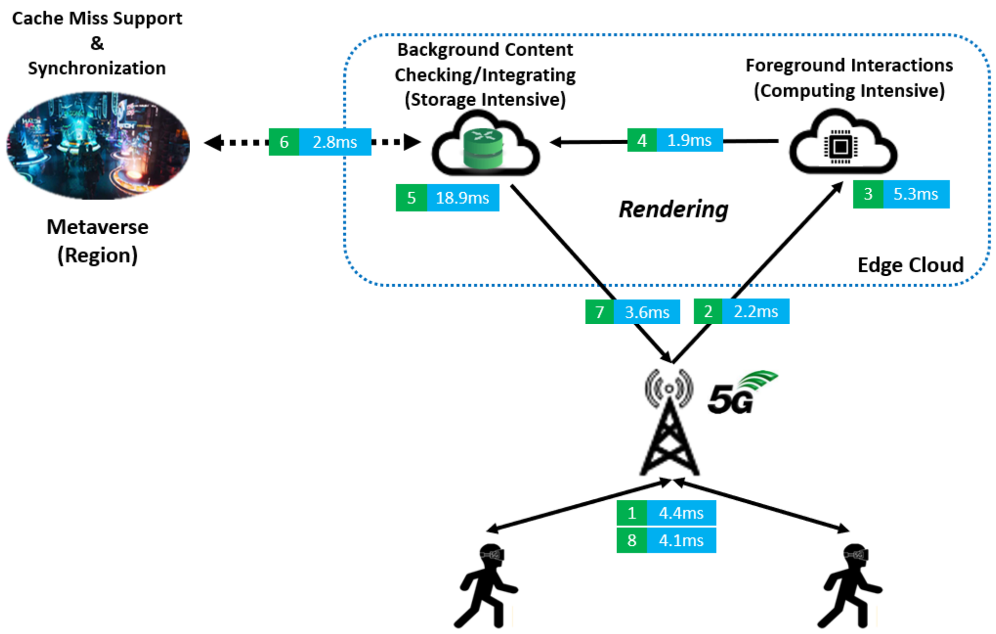
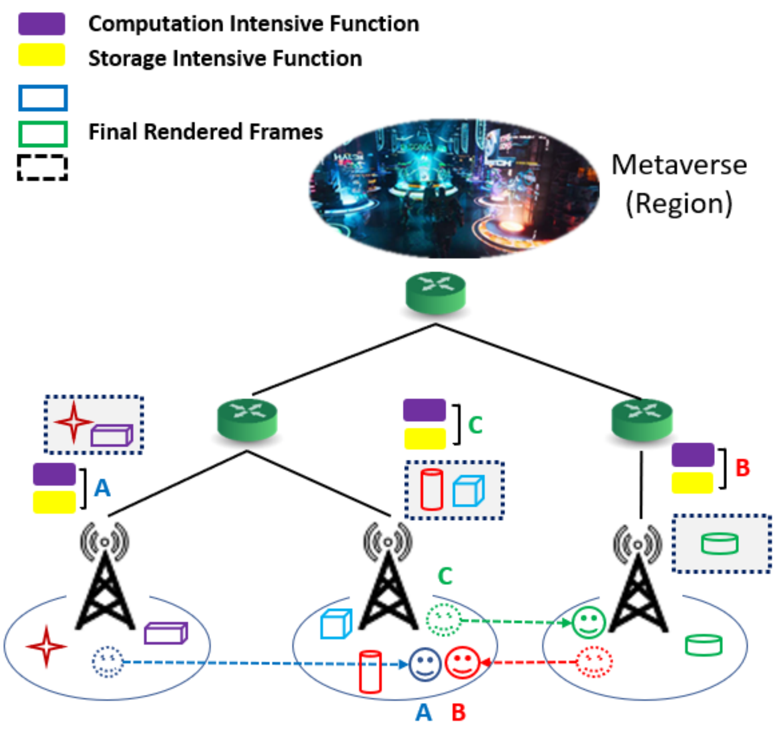


Fig. 6.1 The general work flow of metaverse AR applications with delay of each stage (6 EC, 30 requests, weight μ is 1, EC Capacity is 14 and total mobility probability is 1)

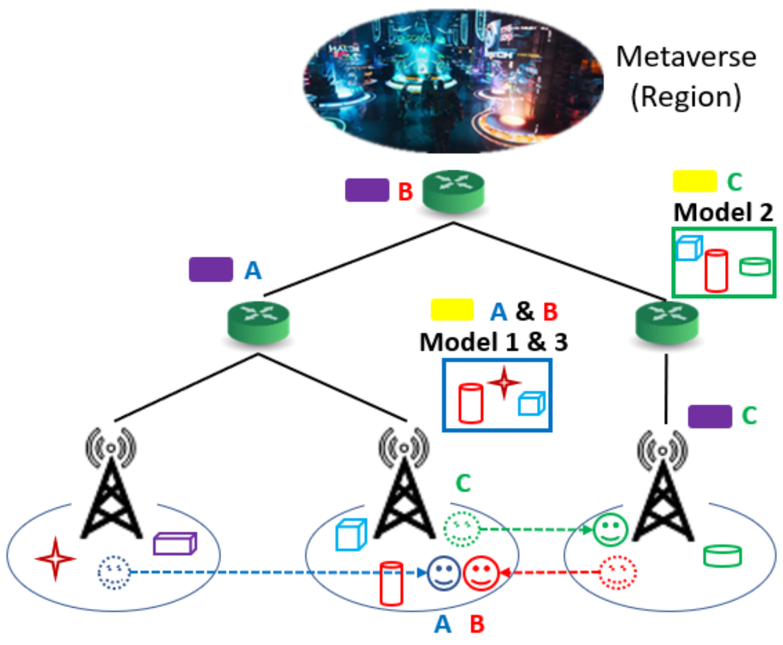
Generally speaking, a metaverse scene will, in essence, consist of a background view as well as many objects in foreground interactions. The background view at a defined amalgamated virtual and physical location can be deemed as static or slowly changing [111, 33]. A typical background scene can be the 3D model of the metaverse, a presentation of a related background virtual environment based on a certain user viewport [33, 57]. Its size can reach tens of MB and the corresponding complexity of rendering related functionalities measured by computation load is also large (e.g., 10 CPU cycles/bit) [33, 113]. On the other hand, objects (such as, for example, avatars) in foreground interactions which are embedded in the metaverse scene change much more frequently; however, they are significantly less complex than the background scene (e.g., 4 CPU cycles/bit) [62, 33]. Howeve, even though those objects are less complex than the background scene, due to their frequent changes they also require rendering in a timely manner to avoid a considerable degradation in the quality of user experience. Thus, in this chapter, rendering for both foreground and background are deployed at the edge clouds (ECs) rather than only at the terminals, to make full use of caching and computing resources. Notice that uploaded information is focused in foreground interactions, while background content checking consumes not only computing resources but also a significant amount of local cache to match and integrate AROs and related models of the metaverse. Hence, similar to our previous work in [42], the metaverse MAR application could also be decomposed into computational- and storage-intensive functions, which serve as a chain for improved assignment and resource allocation.

The general work flow of a metaverse MAR application supported by ECs is shown in Figure 6.1. A metaverse region is a fraction of the complete metaverse and is assumed to be located on a server geographically close to its corresponding service region in the mobile network (not necessarily running within an EC). The metaverse MAR service could be triggered by certain behavior including foreground interactions [111, 33, 39]. Then, content related to background content, such as, for example, pre-cached 3D models and AROs, are first searched in the EC cache to check if they are what the user requires. If the target AROs or model information cannot be found in the cache, then this case is labelled as a "cache miss" and the request is redirected to the original metaverse region stored in a cloud deeper in the network. Finally, according to the user's physical mobility and virtual orientation extracted from foreground interactions, the matched AROs and model are integrated into a final frame and transmitted back to the user [111, 33]. At the same time, updated information is also sent to the metaverse region for synchronization. Thus, the user could be aware of changes caused by other participants if they share the same metaverse region during the service. Based on the above discussion, it is becoming apparent that the overall quality of metaverse MAR applications depends on communication delays and the capabilities of the above-mentioned network entities, which participate in service creation.

Figure 6.2 further reveals the difference between cases with consideration, or not, of user mobility with service decomposition to render requirements by metaverse MAR applications. Clearly, when neglecting user mobility and service decomposition, as shown in case (a), models, target AROs and metaverse MAR applications are all cached as close as possible to the user's initial location. This might leave a heavy burden for the adjacent server when there are multiple users at the same cell and cause a "hot-spot" area [42].



(a) Traditional caching without user mobility and service decomposition



(b) Proactive caching with user mobility and service decomposition

Fig. 6.2 Illustrative toy examples of caching by a metaverse MAR application

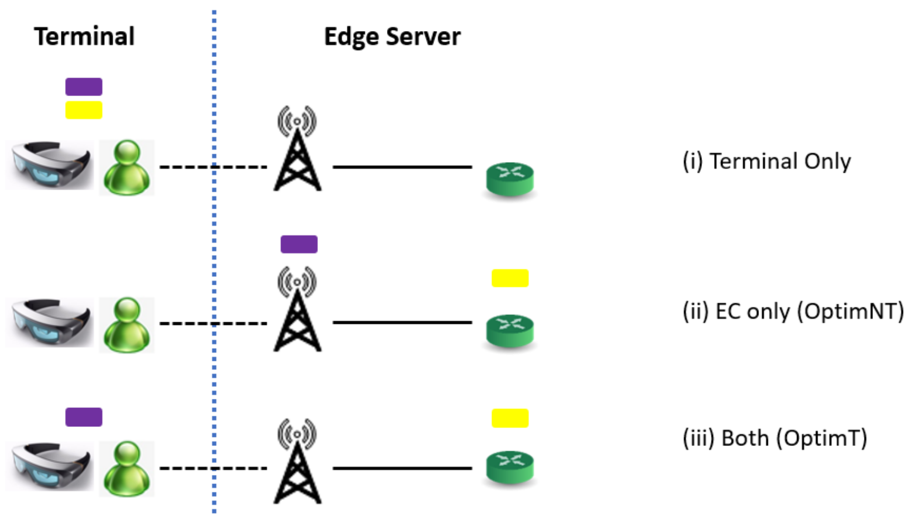


Fig. 6.3 Illustrative toy examples where in case (i) only terminals are activated, in case (ii) only ECs are activated and in case (iii) both are activated

However, when user mobility and service decomposition are enabled, as shown in case (b), decomposed functions can spread over ECs between user's initial location and potential destination. Hence, the service delivery becomes more flexible and efficient in terms of assigning requests and allocating network resources. In case (a), although user A only needs wireless communication in the initial location, it takes two hops after moving to the middle cell. However, when taking mobility and local resources into consideration, the same user, A, in case (b) could experience a shorter delay in the after-mobility event by allowing two more hops before the mobility event. Hence, in a high-mobility scenario, it might not always be ideal to allocate requests and services as close as possible to the user's initial location. As shown in the figure for users A and B in case (b), the AR contents in the model might be similar in terms of the viewport of different users. Hence, participating users should be aware of each other's updates and could share rendering functions to reduce the consumed resources. In this chapter, we apply structural similarity (SSIM), proposed by [107], for user perception experience. It is a widely accepted method which measures the user perception quality of an image by comparing to its original version [107]. Caching more models and AROs also causes more processing and transmission delays, with energy consumption [112, 111]. Hence, the joint optimization has to accept some potential loss due to constraints of computing and storage resources.

Figure 6.3 further reveals the difference among cases which allocate MAR service on either terminals or ECs. Note that when neglecting the EC support and service decomposition, the whole MAR application should run on the terminal and could be a heavy burden; this is shown in case (i). According to [111, 15], MAR applications on terminals take up around 47% of the available power consumption and could also affect the performance of other (potentially critical) functionalities on the terminal when this service becomes more complex, such as in the metaverse. In case (ii), the processing time is significantly reduced through enabling EC support with the cost of an increased transmission delay [44]. Although the terminals are still limited in computing resources and more sensitive to energy consumption, neglecting their capabilities is not an optimal configuration, especially when recent technical improvements showcase their computing and caching potential. Noticing that the foreground interactions are much less complex than background scenes and are more suitable to terminals, we further consider terminals in the scenario, as shown by case (iii), and execute only computationally intensive functions. Hereafter, the optimization of the integration of terminals and ECs becomes the scheme OptimT and is compared to the previous case, (ii), which neglects the terminals (OptimNT). To maintain a fair comparison and focus on energy consumption and service latency, the overall energy consumption is measured explicitly (in Joules) instead of the power, as in [44], and the user perception quality is accepted as a given boundary.

In this chapter, by explicitly considering the terminals, user mobility, service decomposition and models of metaverse regions with embedded AROs, a joint optimization framework (OptimT) is constructed for the metaverse MAR application in the edge-supported network. The proposed optimization framework seeks a balance between energy consumption and service delay under a given level of user perception quality. To reveal the influence of capacity and cost of terminals on the metaverse MAR application, this OptimT scheme is further compared with another optimized framework (OptimNT) proposed in our previous work [44], which only focuses on ECs and neglects terminals.

6.2 Related Work

Hereafter, a series of closely related works in the area edge/cloud support of metaverse type of applications over 5G and beyond wireless networks are discussed and compared with the approach proposed in this chapter.

Noticing that despite their limitations MAR terminals have witnessed a series of technical improvements, their joint utilization with ECs for network optimization can bring significant benefits. In [94], the authors aim to minimize the energy consumption of multicore smart devices, which are usually applied for AR applications. Through tracking the response process of an AR application, they manage to measure the terminal's energy consumption by Amdahl's law. Although the law and a similar framework for the terminal energy consumption are also applied in this chapter, we consider a broader use case scenario that includes the edge servers and a more complex balance between energy and latency. The work in [16] shares a similar target like in this chapter, which relates to achieve a balance between latency and energy consumption under an level of acceptable image quality. However, [16] brings in local sensors at MAR devices for recognizing and tracking AROs so that their scheme could realize selective local visual tracking (optical flow) and selective image offloading. Object recognition stage is more focused in [16] with four AR applications requiring different types of AROs. Without support from ECs, they utilize a further cloud server as a heavy database for 3D AROs and leave most tasks at the terminals whilst cloud offloading only triggered when confronted with a calibration. However in this work, we consider an EC supported network and compare the difference between schemes utilizing explicitly the MAR terminals or not. In [91], the energy efficiency is optimized under required service latency for MAR in an EC supported network. Similarly, authors consider the proactively caching and propose a tradeoff between energy and latency in terms of cache size. However, their mobile cache and power management scheme still focuses the energy consumption on terminals. Clearly, above mentioned works do not explicitly consider user mobility, perception quality, service decomposition and metaverse application features like ours.

The problem of efficient resource allocation for supporting metaverse type of applications is starting to attract significant amount of attention and a plethora of aspects have already been considered. In [36], the emphasis is placed in the synchronization of Internet of Things services, in which they employ IoT devices to collect real world data for virtual service providers. Through calculating maximum awards, users could select the ideal virtual service provider. Researchers then proposes the game framework considers such reward allocation scheme and general metaverse sensing model [36]. In [54], the authors also adopt a game theoretic framework by considering tasks offloading between mobile devices based on Coded Distributed Computing in a proposed vehicular metaverse environment. Another framework proposed by [22] manages and allocates different types of metaverse applications so that common resources among them could be shared through a semi-Markov decision process and an optimal admission control scheme. The work in [74] applies a set of proposed resource optimization schemes in a virtual education metaverse. Mores specifically, a stochastic optimal resource allocation scheme is developed with the aim to reduce the overall cost of incurred by a service provider. Similar to the service decomposition in this paper, they only upload and cache some parts of data or services to achieve reduced levels of delay and offer better privacy [74]. The work in [28] is closely relevant since in that paper not only latency but energy consumption is also considered as is the case of our proposed model that uses a multi-objective optimization approach. For ultra-reliable and low-latency communication services, researchers bring in digital twins and deploy a mobility management entity for each access point to determine probabilities of resource allocation and data offloading [28]. Then, by applying a deep learning neuro network the proposed scheme tries to identify a suitable user association and an optimized resource allocation scheme for this association. While, in this chapter, the core idea is to decompose he service and allow a flexible allocation across edge clouds by taking also into account user mobility. The work in[112] considers virtual reality applications in the metaverse and regards the service delivery as a series of events in the market, in which users are buyers and service provides are sellers. Hence, they apply Double Dutch Auction to achieve common price through asynchronous and iterative biding stages [112]. They emphasize the quality of user perception experience by Structural SIMilarity (SSIM) and Video Multi-Method Assessment Fusion. In our proposed framework, we also utilize the SSIM metric to determine the frame quality after integrating background scene and AR contents [112]. The work in [112] further brings in a deep reinforcement learning-based auctioneer to reduce the information exchanging cost. While in this chapter, a multi-objective optimization approach is taken where we aim to balance different objectives functions using the scalarization method whilst considering the inherent user mobility in an explicit manner.

6.3 System Model

6.3.1 Multi Rendering in Metaverse AR

Similarly, with the set $\mathbb{M} = \{1, 2, ..., M\}$ we denote the available locations, including available edge clouds and terminals. Also assuming that each user makes a single request, requests defined by $r \in \mathbb{R}$ in the metaverse region generated by mobile users could be equipped with MAR devices. $f(r)$still represents the initial access router where this user is firstly connected to. For each request, the user terminal could also be viewed as a valid location to cache or to process information locally. Thus, we define with $j_r \in \mathbb{M}$ as the terminal sending the request r. The terminals are brought into consideration in the following formulation. Through defining the location with constraint as $j \in \mathbb{M}, j \neq j_r$, this could only enable the ECs. Clearly, for the OptimNT scheme, this works for all locations. While in the following formulation for the OptimT scheme, we force that the storage intensive functions are executed only on the ECs while the computing intensive ones could also hosted at the end terminals. During the mobility event, a user could move to different potential destinations $k \in \mathbb{K}$ (i.e., changing of the anchoring point). Hereafter, and without loss of generality, we only accept adjacent access routers as available destinations in the mobility event. A series of metaverse regions are set on ECs to interact with users. The corresponding metaverse region serving the user can be found through functions $A(f(r)), A(k)$. As explained earlier, each metaverse region is pre-deployed on a server close to the mobile network and its distance to an EC is also predefined. In this chapter, as already alluded, a set of AROs is assumed to be embedded across the different background metaverse region models and is defined as $\mathbb{N} = \{1, 2, ..., N\}$. A set $\mathbb{S}_r = \{1, 2, ..., S\}$ is defined for multiple rendering of the available metaverse region model to each user. Thus, we denote the decision variable p_{sj} for pre-caching a metaverse region model $s \in \mathbb{S}_r$ at the EC j ($j \in \mathbb{M}, j \neq j_r$). The subset $\mathbb{L}_{rs}$ represents the target AROs required by the user r in the related model $s \in \mathbb{S}_r$ and the size of each target ARO $l \in \mathbb{L}_{rs}$ is denoted as O_l. Lastly, the decision variable h^s_{rl} is brought in for proactively caching an ARO required by a request r. Based on the above, the decision variables p_{sj} and h^s_{rl} can be defined as follows,

$$p^j = \begin{cases} 1, \text{if rendering the related model } s \text{ at node } j, \\ 0, \text{otherwise.} \end{cases} \tag{6.1}$$

$$h^s_{rl} = \begin{cases} 1, \text{if ARO } l \text{ required by request } r \text{ embedded} \\ \quad \text{in the model } s \text{ is cached,} \\ 0, \text{otherwise.} \end{cases} \tag{6.2}$$

Furthermore, the additional set of constraints need to be satisfied,

$$\sum_{r \in \mathbf{R}} h^s_{rl} \leqslant 1,\ \forall j \in \mathbf{M}, j \neq j_r,\ \forall s \in \mathbf{S_r},\ \forall l \in \mathbf{L_{rs}} \tag{6.3}$$

$$\sum_{s \in \mathbf{S_r}} \sum_{l \in \mathbf{L_{rs}}} h^s_{rl} \geqslant 1,\ \forall r \in \mathbf{R} \tag{6.4}$$

$$\sum_{j \in \mathbf{M}, j \neq j_r} p_{sj} \geqslant h^s_{rl},\ \forall r \in \mathbf{R},\ \forall s \in \mathbf{S_r},\ \forall l \in \mathbf{L_{rs}} \tag{6.5}$$

$$h^s_{rl} \leqslant h^s_{rl} \sum_{j \in \mathbf{M}, j \neq J_r} p_{sj},\ \forall r \in \mathbf{R},\ \forall s \in \mathbf{S_r},\ \forall l \in \mathbf{L_{rs}} \tag{6.6}$$

Constraints in (6.3) force each ARO to be pre-cached at most once in a related model. Constraints (6.4) ensure thay a valid request consists of at least one model and an embedded ARO. Constraints in (6.5) guarantee that the allocation of an ARO happens in conjunction to the decision of proactive caching whilst constraints in (6.6) further certify that the rejection of the model's proactive caching causes any ARO planned to be embedded in this model should not be pre-cached as well. Thus, (6.5) only accepts ARO in an pre-cached model and (6.6) rejects all related ones when failing to cache a model, which together could ensure the model and corresponding AROs cannot be handled separately during the formulation.

6.3.2 Wireless Resource Allocation and Channel Model

With B_j we express the bandwidth of the resource block and γ_{rj} denotes the Signal to Interference plus Noise Ratio (SINR) of the user r at the node j. With P_{rj}^{tran} we denote the transmit power of the user r at the node j, and P_i is transmission power at the base station. Furthermore, H_{rj} is the channel gain, N_j is the noise power and a is the path loss exponent whilst d_{rj} is the distance between the user and the base station. Finally, a nominal Rayleigh fading channel is used to capture the channel between the base stations and the users[31]. More specifically, the channel gain H_{rj} can be written as follows [20],

$$H_{rj} = \sqrt{\frac{1}{2}}(t + t^{'}J) \tag{6.7}$$

Where $J^2 = -1$, t and $t^{'}$ are random variables following the standard normal distribution. Based on the above, the SINR γ_{rj} in a 5G wireless network can be written as follows [20][105],

$$\gamma_{rj} = \frac{P_{rj}^{tran} H_{rj}^2 d_{rj}^{-a}}{N_j + \sum_{i \in \mathbf{M}, i \neq j} P_i H_{ri}^2 d_{ri}^{-a}} \tag{6.8}$$

The data rate is denoted as $g \in \mathbb{G}$ and the binary decision variable e_{rg} decides whether to select the data rate g for user r,

$$e^{rg} = \begin{cases} 1, \text{if data rate } g \text{ is selected for user } r, \\ 0, \text{otherwise.} \end{cases} \tag{6.9}$$

Noticing that the chosen data rate can also be written as $B_j \log_2(1 + \gamma_{rj})$, after choosing a data rate as ge_{rg} for the user, the current transmit power P_{rj}^{tran} can be written as follows,

$$P_{rj}^{tran} = \frac{N_j + \sum_{i \in \mathbf{M}, i \neq j} P_i H_{ri}^2 d_{ri}^{-a}}{H_{rj}^2 d_{rj}^{-a}} (2^{\frac{ge_{rg}}{B_j}} - 1) \tag{6.10}$$

Note that $2^{\frac{ge_{rg}}{B_j}} = (1 - e_{rg}) + e_{rg} 2^{\frac{g}{B_j}}$ and should satisfy the following constraint to ensure that a single data rate is selected per user,

$$\sum_{g \in \mathbb{G}} e_{rg} = 1, \forall r \in \mathbb{R} \tag{6.11}$$

6.3.3 Latency, Engergy Consumption and Quality of Perception Experience

Similar to our previous work in [42], the MAR service can be decomposed into computational intensive and storage intensive functionalities and are defined as η and ρ respectively. For these functionalities, their corresponding execution locations are then denoted as x_{ri} and y_{ri} respectively [42]. In a mobility event, the user's moving probability from the starting location to an allowable destination could be known to mobile operators through learning from the historical data and hence is defined as $u_{f(r)k} \in [0,1]$ ($\{f(r),k\} \subset \mathbb{M}$). The size of foreground interactions is denoted as $F_{\eta r}^{fore}$, the size of pointers used for matching AROs is denoted as $F_{\rho r}$ and the size of the related model s used for background content checking is F_{sr}^{back} [42][33]. During the matching and background content validation process, the target AROs and/or the background content are possibly not pre-cached in the local cache and such case is known as a "cache miss" (otherwise there is a "cache hit"). A cache miss in the local cache inevitably triggers the redirection of the request to the metaverse region stored in a core cloud deeper in the network and this extra cost in latency is defined as the penalty D. After rendering, the model and target AROs are integrated into a compressed final frame for transmission and its compressed size is denoted as F_{sr}^{res}.

In this section, a joint optimization scheme is proposed that aims to balance the service delay and the energy consumption under the constraint of the user perception quality of the decomposed Metaverse AR services in the EC supported network. The cache hit/miss is expressed by the decision variable z_{rj} and can be written as follows,

$$z_{rj} = \begin{cases} 1, \text{ if } \sum\limits_{l \in \mathbf{L_{rs}}} \sum\limits_{s \in \mathbf{S_r}} p_{sj} h_{rl}^{s} \geqslant L_{rs}, \\ 0, \text{ otherwise.} \end{cases} \tag{6.12}$$

The cache capacity of an EC and the cache hit/miss relation can be written as follows,

$$\sum_{r \in \mathbf{R}} \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} p_{sj} h_{rl}^{s} O_l \leq \Theta_j, \forall j \in \mathbf{M},\ j \neq j_r \tag{6.13}$$

$$\begin{gathered} \sum_{l \in \mathbf{N}} \sum_{s \in \mathbf{S_r}} h_{rl}^{s} + \varepsilon \leq L_{rs} + U(1 - q_{rj}) \\ \forall j \in \mathbf{M},\ j \neq j_r, r \in \mathbf{R} \end{gathered} \tag{6.14}$$

where Θ_j is the cache capacity at node j. In (6.14), to rewrite the either-or constraint that $\sum_{l \in \mathbf{N}} \sum_{s \in \mathbf{S_r}} h_{rl}^{s} < L_{rs}$ or $z_{rj} = 1$, we define ε as a small tolerance value, U as a large arbitrary number and q_{rj} as a new decision variable satisfying $1 - q_{rj} = z_{rj}$ [42]. Undoubtedly, increased levels of pro-caching decisions related to the background models and embedded

AROs in a request inevitably brings about an extra execution burden for the matching function. Taking the above into account, the actual processing delay of the computational intensive function can be expressed as follows,

$$V_{rj} = \frac{\omega_\eta F_{\eta r}^{fore}}{f_V^j} \tag{6.15}$$

Similarly, the processing delay of the matching and background content checking function is assumed happen only at serves and can be written as,

$$W_{rj} = \frac{\omega_\rho (F_{\rho r} + \sum_{l \in \mathbf{L_{rs}}} \sum_{s \in \mathbf{S_r}} p_{sj} h_{rl}^s O_l + \sum_{s \in \mathbf{S_r}} F_{sr}^{back} p_{sj})}{f_V^j} \tag{6.16}$$

where ω_η and ω_ρ (cycles/bit) represent the computation load of foreground interaction and background matching, f_V^j is the virtual CPU frequency (cycles/sec), $F_{\rho r}$ are the size of uploaded pointers of AROs in foreground interactions [42][33]. When finding the target AROs during matching, their pointers included by foreground interactions should also be transferred to the metaverse for updating. Finally, the final frame integrating the model and target AROs are transmitted back to the user. Hence, the overall transmission delay for each user after processed by functions can be written as follows,

$$\begin{aligned} \sum_{s \in \mathbf{S_r}} \sum_{j \in \mathbf{M}, j \neq j_r} (C_{jA(f(r))} + C_{jA(k)}) p_{sj} + \\ (C_{A(f(r))f(r)} + \sum_{k \in \mathbf{K}} C_{A(k)k} u_{f(r)k}) \end{aligned} \tag{6.17}$$

Note that the product of decision variables, $p_{sj}h_{rl}^s$, $p_{sj}y_{rj}$ and $p_{sj}h_{rl}^s y_{rj}$, create a non-linearity. Observe that $p_{sj}h_{rl}^s$ and $p_{sj}y_{rj}$ appear directly while the product $p_{sj}h_{rl}^s y_{rj}$ appears in $W_{rj}y_{rj}$, which represents the execution of the matching function at the location j ($j \in \mathbb{M}, j \neq j_r$). To express the optimization problem in a nominal linear programming setting, we linearize the above expressions via new auxiliary decision variables. To this end, a new decision variable α_{rsj} is introduced as $\alpha_{rsj} = p_{sj}y_{rj}$ and the constraints should be added as follows,

$$\begin{aligned} &\alpha_{rsj} \leqslant p_{sj}, \\ &\alpha_{rsj} \leqslant y_{rj}, \\ &\alpha_{rsj} \geqslant p_{sj} + y_{rj} - 1 \end{aligned} \tag{6.18}$$

Similarly, a new decision variable β_{rslj} is introduced as $\beta_{rslj} = p_{sj}h^s_{rl}$ and the constraints should be added as follows,

$$\begin{aligned}
&\beta_{rslj} \leqslant p_{sj},\\
&\beta_{rslj} \leqslant h^s_{rl},\\
&\beta_{rslj} \geqslant p_{sj} + h^s_{rl} - 1
\end{aligned} \tag{6.19}$$

The aforementioned constraint (6.6) is affected and should be rewritten as follows,

$$h^s_{rl} \leqslant \sum_{j\in\mathbf{M}} \beta_{rslj},\ \forall r \in \mathbf{R},\ \forall s \in \mathbf{S_r},\ \forall l \in \mathbf{L_{rs}} \tag{6.20}$$

Also, note that p_{sj} is a binary decision variable and causes $p_{sj} = p^2_{sj}$. Thus, we have $p_{sj}h^s_{rl}y_{rj} = \alpha_{rsj}\beta_{rslj}$. A new decision variable λ_{rslj} is introduced as $\lambda_{rslj} = \alpha_{rsj}\beta_{rslj}$ and the following constraints should be added as follows,

$$\begin{aligned}
&\lambda_{rslj} \leqslant \alpha_{rsj},\\
&\lambda_{rslj} \leqslant \beta_{rslj},\\
&\lambda_{rslj} \geqslant \alpha_{rsj} + \beta_{rslj} - 1
\end{aligned} \tag{6.21}$$

Hence, the product $W_{rj}y_{rj}$ can be rewritten as follows,

$$\frac{\omega_\rho(F_{\rho r}y_{rj} + \sum_{l\in\mathbf{L_{rs}}}\sum_{s\in\mathbf{S_r}}\lambda_{rslj}O_l + \sum_{s\in\mathbf{S_r}}F^{back}_{sr}\alpha_{rsj})}{f^j_V} \tag{6.22}$$

By checking whether users share the same metaverse region by $A(f(t)) = A(f(r)), \{t,r\} \subset \mathbb{R}$, we can ensure the user could also view other updates happening in the same metaverse region. Based on the previous modelling of wireless channel, the wireless transmission delay in a mobility event can be written as follows,

$$\begin{aligned}
&\sum_{r\in\mathbf{R}} \frac{F^{fore}_{\eta r} + \sum_{t\in\mathbf{R},A(f(t))=A(f(r))}\sum_{s\in\mathbf{S_r}} p_{sj}F^{res}_{st}}{ge_{rg}} +\\
&\sum_{r\in\mathbf{R}}\sum_{k\in\mathbf{K}} u_{f(r)k}\frac{F^{fore}_{\eta r} + \sum_{t\in\mathbf{R},A(t)=A(k)}\sum_{s\in\mathbf{S_r}} p_{sj}F^{res}_{st}}{ge_{rg}}
\end{aligned} \tag{6.23}$$

Noticing that with constraint (6.11), $\frac{1}{e_{rg}}$ can be replaced by e_{rg} for linearization. By introducing a new decision variable ϕ_{rlsg} with following constraints,

$$\begin{aligned}
&\phi_{rsg} \leqslant e_{rg},\\
&\phi_{rsg} \leqslant p_{sj},\\
&\phi_{rsg} \geqslant e_{rg} + p_{sj} - 1
\end{aligned} \tag{6.24}$$

Thus, the previous formula (6.23) can be updated as follows,

$$\begin{aligned}\frac{1}{g}\sum_{r\in\mathbf{R}}(1+\sum_{k\in\mathbf{K}}u_{f(r)k})(F^{fore}_{\eta r}e_{rg}+\\ \sum_{t\in\mathbf{R},A(f(t))=A(f(r))}\sum_{s\in\mathbf{S_r}}\phi_{rsg}F^{res}_{st})\end{aligned} \tag{6.25}$$

Based on the above derivations and inline with [42], the overall latency can be written as follows,

$$\begin{aligned}L =&(6.25)+\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}(C_{f(r)i}+V_{ri})x_{ri}+\\ &\sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{M}}\sum_{j\in\mathbf{M},j\neq j_r}((6.22)+C_{ij}\xi_{rij}+C_{A(f(r))f(r)}+\psi_{rj}D)+\\ &\sum_{s\in\mathbf{S_r}}\sum_{j\in\mathbf{M},j\neq j_r}(C_{jA(f(r))}+C_{jA(k)})p_{sj}+\\ &\sum_{r\in\mathbf{R}}\sum_{k\in\mathbf{K}}(C_{A(k)k}+C_{ki}x_{ri})u_{f(r)k}\end{aligned} \tag{6.26}$$

where V_{ri} is the processing delay of computational intensive function [42]. L_{max} here denotes the maximum allowed service latency and has $\frac{L}{L_{max}}\in[0,1]$.

The energy efficiency of the system during each service time slot is measured by the production of its total power and running time. The server total power consists of the transmission power and CPU processing power at target ECs. Denote the required CPU processing power of the user r at the node j as P^{cpu}_{rj} and the CPU chip architecture coefficient as k_0 (e.g. 10^{-18}) [28]. Then the power at the EC can be achieved through $k_0(f^j_V)^2$ (J/s) based on measurements in [118][69]. Noticing that for both conditions, the background contents are processed at servers, hence the consumed processing time of the server is,

$$T_{cpu}=\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M},j\neq m_r}V_{rj}x_{rj}+\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}W_{rj}y_{rj} \tag{6.27}$$

The wireless transmission happens no matter if executing foreground interactions at terminals. Since the selected data rate is ge_{rg}, the wireless transmission time is,

$$T_{tran}=\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M},j\neq j_r}\frac{F^{fore}_{\eta r}}{ge_{rg}}+\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M},j=j_r}\frac{F_{\rho r}}{ge_{rg}} \tag{6.28}$$

Finally, the total consumed energy at the server side can be written as follows,

$$\begin{aligned}E_{server} =& \sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}(P_{rj}^{tran}T_{tran}+P_{rj}^{cpu}T_{cpu})\\ =& \sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}(\frac{N_j+\sum_{i\in\mathbf{M},i\neq j}P_iH_{ri}^2d_{ri}^{-a}}{H_{rj}^2d_{rj}^{-a}}(2^{\frac{ge_{rg}}{B_j}}-1))\\ &(\sum_{j\neq j_r}\frac{F_{\eta r}^{fore}}{ge_{rg}}+\sum_{j=m_r}\frac{F_{\rho r}}{ge_{rg}})\\ &+\sum_{r\in\mathbf{R}}\sum_{j\in\mathbf{M}}k_0(f_V^j)^2(W_{rj}y_{rj}+\sum_{j\neq j_r}V_{rj}x_{rj})\end{aligned} \tag{6.29}$$

Regarding the terminal side, in this chapter we follow the Amdahl's law to model the energy consumption, in which considers the potential speedup of potential parallel computations in the function [3]. In this chapter, the metaverse AR functions work by serial and for simplicity the parallel portion is assumed to be zero like [94]. As mentioned earlier, the dynamic foreground interactions including some highly used AROs could be proactively cached at terminals with the matching function. The 3D background model, on the other hand, is much larger and might serve multiple users in a region. Hence, it is not recommended to be stored or processed at the terminal. In addition, the metaverse application should not exceed a certain portion of the whole terminal cpu resources so that other functionalities could work properly [94]. Denoting the consumed portion as $\Gamma_r \in [30\%, 50\%]$ [94], then $\frac{V_{rj}x_{rj}}{\Gamma_r}$ means processing metaverse AR functions at the terminal requires a longer time. The Energy consumption of terminals could be written as follows,

$$\begin{aligned}E_{terminal} =& P_{terminal}T_{terminal}\\ =& \sum_{r\in\mathbf{R}}\sum_{j=j_r}k_0(f_V^j)^2\frac{V_{rj}x_{rj}}{\Gamma_r}\end{aligned} \tag{6.30}$$

Finally, the overall system energy consumption is $E = E_{server} + E_{terminal}$. E_{max} represents the maximum possible energy consumption of the system. It also has $\frac{E}{E_{max}} \in [0,1]$.

SSIM is applied to reveal the quality of perception experience. In this chapter, the video coding scheme (e.g. H.264) and frame resolution (e.g. 1280×720) are assumed as pre-defined [57]. Then SSIM is mainly affected by data rate and a concave function could be applied to reveal the relation between them[57]. Hence, denote the set of SSIM values for each ARO under corresponding data rates as $\mathbb{SSIM}_l, l \in \mathbb{L}_r$. The overall quality of perception experience Q can be written as follows,

$$Q = \sum_{r\in\mathbf{R}}\sum_{l\in\mathbf{L_r}}\sum_{g\in\mathbf{G}}\sum_{c\in\mathbf{SSIM}_l}e_{rg}c \tag{6.31}$$

To maintain the user experience above an acceptable level, the perception quality constraint could be added as follows,

$$\frac{Q}{Q_{max}} \geq Q_{bound} \tag{6.32}$$

where Q_{max} is the maximum quality through selecting max allowable data rate and storing as many AROs as possible.

Using a weighting parameter $\mu \in [0,1]$, the bi-opbjective optimization problem can be written as follows,

$$min\, \mu \frac{L}{L_{max}} + (1-\mu)\frac{E}{E_{max}} \tag{6.33a}$$

$$\text{s.t. } z_{rj} = 1 - q_{rj},\, \forall j \in \mathbf{M}, r \in \mathbf{R} \tag{6.33b}$$

$$\sum_{r \in \mathbf{R}} (x_{rj} + y_{rj}) \leq \Delta_j, \forall j \in \mathbf{M}, j \neq j_r \tag{6.33c}$$

$$\sum_{j \in \mathbf{M}} x_{rj} = 1, \forall r \in \mathbf{R} \tag{6.33d}$$

$$\sum_{j \in \mathbf{M}, j \neq j_r} y_{rj} = 1, \forall r \in \mathbf{R} \tag{6.33e}$$

$$\xi_{rij} \leq x_{ri},\, \forall r \in \mathbf{R}, i, j \in \mathbf{M}, j \neq j_r \tag{6.33f}$$

$$\xi_{rij} \leq y_{rj},\, \forall r \in \mathbf{R}, i, j \in \mathbf{M}, j \neq j_r \tag{6.33g}$$

$$\xi_{rij} \geq x_{ri} + y_{rj} - 1,\, \forall r \in \mathbf{R}, i, j \in \mathbf{M}, j \neq j_r \tag{6.33h}$$

$$\psi_{rj} \leq z_{rj},\, \forall r \in \mathbf{R}, j \in \mathbf{M}, j \neq j_r \tag{6.33i}$$

$$\psi_{rj} \leq y_{rj},\, \forall r \in \mathbf{R}, j \in \mathbf{M}, j \neq j_r \tag{6.33j}$$

$$\psi_{rj} \geq z_{rj} + y_{rj} - 1,\, \forall r \in \mathbf{R}, j \in \mathbf{M}, j \neq j_r \tag{6.33k}$$

$$\begin{aligned} & x_{rj}, y_{rj}, p_{sj}, h^{s}_{rl}, z_{rj}, q_j \in \{0,1\}, \\ & \alpha_{rsj}, \beta_{rslj}, \lambda_{rslj}, \phi_{rslg}, \psi_{rj}, \xi_{rij} \in \{0,1\}, \\ & \forall r \in \mathbf{R}, j \in \mathbf{M}, l \in \mathbf{L_{rs}}, s \in \mathbf{S_r} \end{aligned} \tag{6.33l}$$

(6.3), (6.4), (6.5), (6.20), (6.11), (6.13), (6.14), (6.18), (6.19), (6.21), (6.24), (6.32)

As mentioned earlier, any assignment relating to the storage intensive functions (y_{rj}) are limited to only ECs and hence should apply the constraint $j \neq j_r$. The constraint (6.33b) together with constraints (6.3) to (6.20) reveal the relation between pre-caching decisions and the cache miss/hit for each request [42]. The constraint (6.33c) is the virtual machine limitation while (6.33d) and(6.33e) guarantee the once execution of each function of a request at a single server as explained in [42]. The constraints (6.18) to (6.21) and (6.33f) to (6.33k) are auxiliary and required to solve the product of decision variables for linearization.

6.4 Numerical Investigations

In this section, the effectiveness of the proposed optimization scheme, which will be referred to as Optim in the sequel, is investigated and compared with a number of nominal/baseline schemes.

Same nominal tree-like network topology used in previous chapters is applied for the current metaverse AR service and 30 requests are sent by MAR devices. The remaining available resources allocated for metaverse AR support within an EC are assumed to be CPUs with frequency as 4 to 8 GHz, CPU chip architecture coefficient as 10^{-18} (affected by the chip's design and structure) [28], 4 to 8 cores and $[100,400]$MBytes of cache memory [42]. Similarly, the mobiles AR devices are assumed to own a CPU with 1 GHz frequency, 4 cores and $[0,100]MBytes$ available cache memory for Metaverse AR applications [94]. According to [15], the power of AR applications should not exceed 50% of the mobile device's CPU total power of the mobile device $(2-3W)$ so that other functionalities could operate efficiently. Thus, the service delay of the aforementioned work flow within the above time interval could be regarded as acceptable. Each request requires a single free resource unit for each service function, such as for example a Virtual Machine (VM) [67]. Up to 14 available VMs are assumed in each EC, with equal splitting of the available CPU resources [42]. Note that different view ports lead to different models of the metaverse [33] and up to 4 different models can be cached. All target AROs must be integrated with the corresponding model and rendered within the frame before being streamed to the end user based on a matched result. After triggering the metaverse MAR service, pointers to identify AROs like a name or index are usually a few bytes [115] and hence their transmission and processing are neglected in the following simulations. The set of available data rates is $\{2,3,...,8\}$Mbps and its corresponding SSIM values set is $\{0.955,0.968,...,0.991\}$ [57]. We require the acceptable average SSIM above 0.97 (Q_{bound}). We maintain the assumptions for a nominal 5G base station as in previous chapters like its cell radius (250m), its carrier frequency (2GHz) and etc. And without the loss of generality, each user can utilize only one resource block [59][63][19]. As mentioned earlier, we accept a predefined video coding scheme H.264 with a fixed frame resolution as 1280×720 [57] in RGB (8 bits per pixel). Based on the given resolution, the size of foreground interactions after decoding and compressing can be calculated through multiplying the coefficients $\frac{5}{9}$ and 10^{-3} [33].

Matlab on a personal PC with its CPU of intel i7, 6500U and 2 cores is applied for the simulation. Key simulation parameters are shown below in Table (6.1).

Table 6.1 Simulation parameters

Parameter	**Value**
Number of available ECs	6
Number of available VMs per EC (EC Capacity)	14
Number of requests	30
Number of available models per user	4
AR object size	$(0,10]$ MByte
Total moving probability	$[0,1]$
Cell Radius	250m
Remained Cache Capacity per EC	$[100,400]$ MByte
EC CPU frequency	[4,8] GHz
EC CPU cores	[4,8]
EC CPU core portion per VM	$0.25-0.5$
Remained Cache Capacity per Terminal	$[0,100]$ MByte
Terminal CPU frequency	1 GHz
Terminal CPU cores	4
CPU Architecture Coefficient	10^{-18}
Foreground Interaction Computational Load	4 cycles/bit
Background Content Checking Computational Load	10 cycles/bit
Carrier Frequency	2 GHz
Transmission Power	20dBm
Path Loss Exponent	4
Noise Power	10^{-11}W
Number of Resource Blocks	100
Frame Resolution	1280×720
Average latency per hop	2 ms
Cache miss penalty	25 ms

In following figures and discussion, the optimized scheme considering terminals is denoted as OptimT, while the other one that does not include the terminals is denoted as as OptimNT. The OptimNT scheme could be regarded as an natural extension from our previous work [44]. Two other baseline schemes sharing same caching decisions as the proposed Optim scheme are also implemented for comparison. Those are the RandS and the CEC schemes as applied in previous chapters.

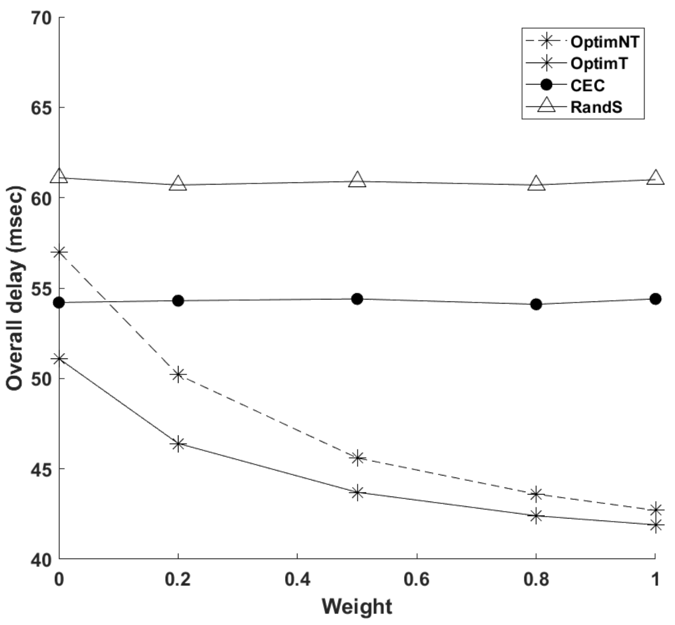


Fig. 6.4 Overall Delay with Weight μ (6 EC, 30 Requests, EC Capacity is 14 and total mobility probability is 1)

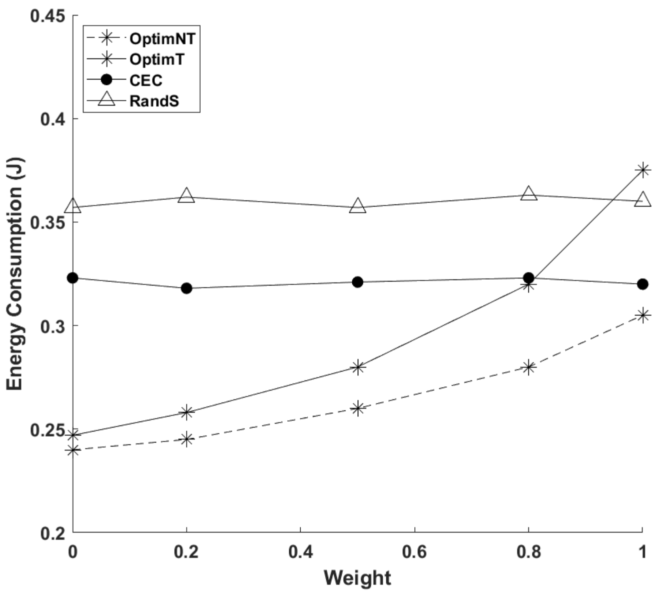


Fig. 6.5 Average Energy Consumption with Weight μ

According to Fig. (6.4), the service delay for each request of the proposed schemes decrease, as expected, with an increasing weight μ. With a larger weight, the proposed schemes tend to select a larger data rate and direct the service to more powerful ECs, which naturally lead to a smaller overall delay. Compared to the OptimNT scheme, for example, the gain in delay of the OptimT scheme ranges from 1.9% to 10.4%. When seeking for the best energy efficiency ($\mu = 0$), since the CPU resources at the terminals are also shared by other functionalities, the OptimT scheme also tries to avoid the occupation of terminals. Hence, in this case, these two schemes share similar solutions and approach to each other. Noticing that the proposed schemes do not care about latency cost, they could choose further and busy EC which even cause the OptimNT scheme become worse than the CEC scheme. Afterwards, as the weight μ increases and the emphasis is placed on latency rather than energy, causes the OptimT scheme to allocate some foreground interactions at terminals and becomes better than OptimNT. Since the baseline schemes neglect energy consumption, service decomposition and mobility, their gaps to the proposed schemes become larger with an increasing weight. However, such gain in delay comes with an extra cost in energy consumption. As shown by Fig. (6.5), the energy consumption per request of the proposed schemes increase with a larger weight. Compared to the OptimNT scheme, the OptimT scheme consumes 2.9% to 23.0% more energy under different weights. Thus, it might not always be worthy to endure lots of energy consumption for narrow gains in delay. By selecting a suitable weight, a balance could be achieved with the OptimT scheme between delay and energy consumption. Through the utilization of terminals, the OptimT scheme becomes the most sensitive to the energy consumption and there could be instances that it might consume more energy that the RandS scheme.

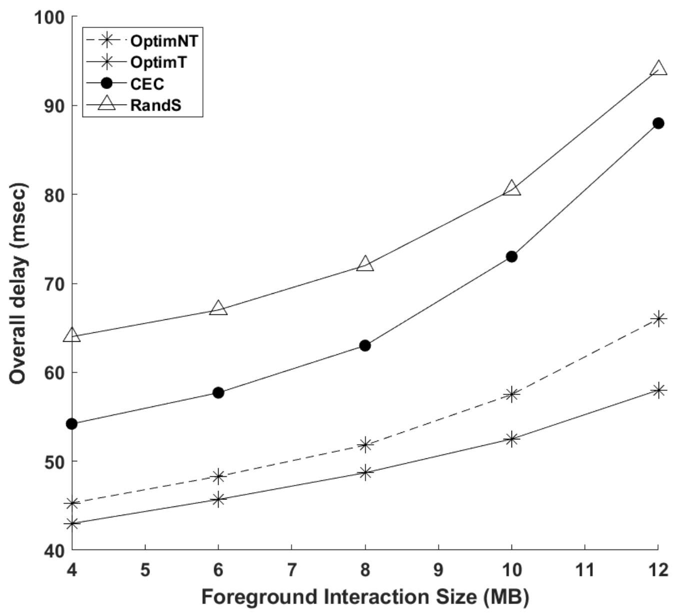


Fig. 6.6 Overall Delay with Foreground Interaction Size (6 EC, 30 Requests, $\mu = 0.5$, EC Capacity is 14 and total mobility probability is 1)

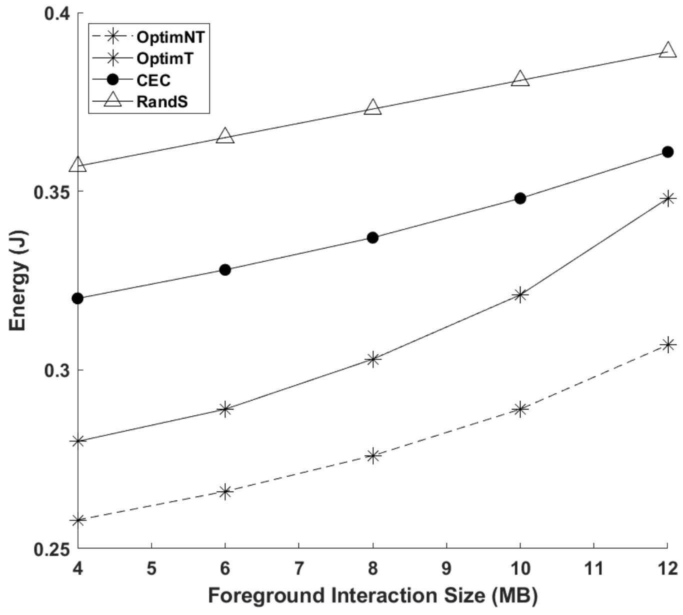


Fig. 6.7 Average Energy Consumption with Foreground Interaction Size (6 EC, 30 Requests, $\mu = 0.5$, EC Capacity is 14 and total mobility probability is 1)

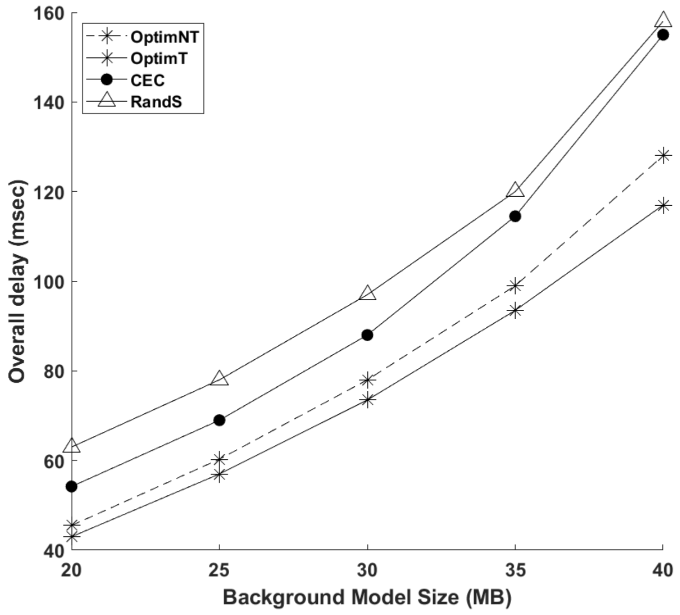


Fig. 6.8 Overall Delay with Background Model Size (6 EC, 30 Requests, $\mu = 0.5$, EC Capacity is 14 and total mobility probability is 1)

Clearly, the user experience could be elevated through viewing more AROs in foreground interactions or more delicate scenes from background models. Fig. (6.6) reveals the variation of delay with the increasing foreground interaction size.When the average foreground interaction size is not too large and there are still enough resources at target

ECs, OptimNT and baseline schemes increase almost linearly at a similar speed. When the size keeps increasing and resources become limited, the CEC scheme becomes the most sensitive one because it only targets on several closest ECs and is easier to trigger the penalty. The optimT scheme, on the other hand, maintains the least latency and the least increasing tendency. To this end, it could save up to 13.8% and 51.7% delay compared respectively to the OptimNT scheme and the CEC scheme. Fig. (6.7) further reveals the variation of energy in this case. Baseline schemes process foreground interactions at ECs without considering energy. Hence, their decisions are not obviously affected by the size of foreground interactions and their energy consumption increases almost linearly. The OptimNT schemes keeps finding more suitable ECs according to current foreground interaction size and remained resources while the OptimT schemes further enables terminals to process some works. Compared to the CEC scheme, they could save over 14.3% energy. It is necessary to point out that the energy consumption will not be taken into account when redirecting the request to the further core cloud and triggering the penalty. According to Fig. (6.8), the background model size is much larger and could cause a significant increase in delay. Note that the terminals could take charge of some foreground interactions and hence could make room for background models in the OptimT scheme. It is still the best scheme in terms of delay and could be up to 9.8% less than the OptimNT scheme. As mentioned earlier, the proactively caching and processing of background models only happen at ECs, these schemes share a similar level of increased energy consumption until triggering the overloading penalty.

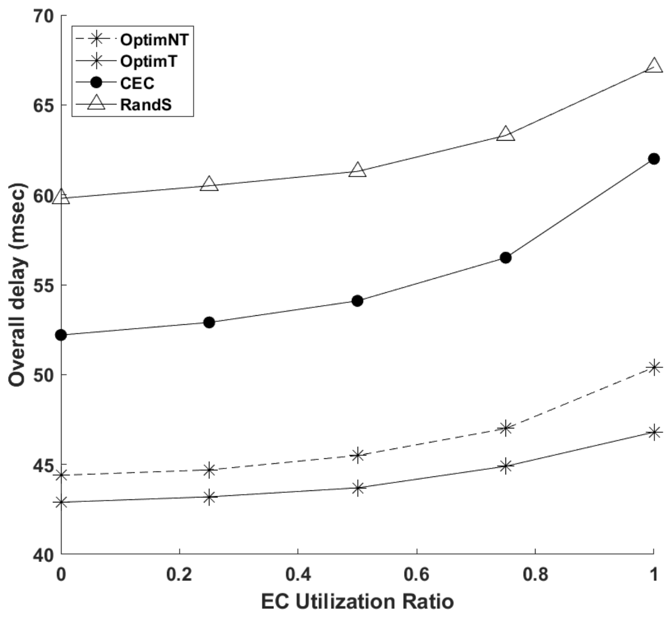


Fig. 6.9 Overall Delay with Average EC Utilization Rate (6 EC, $\mu = 0.5$ and total mobility probability is 1)

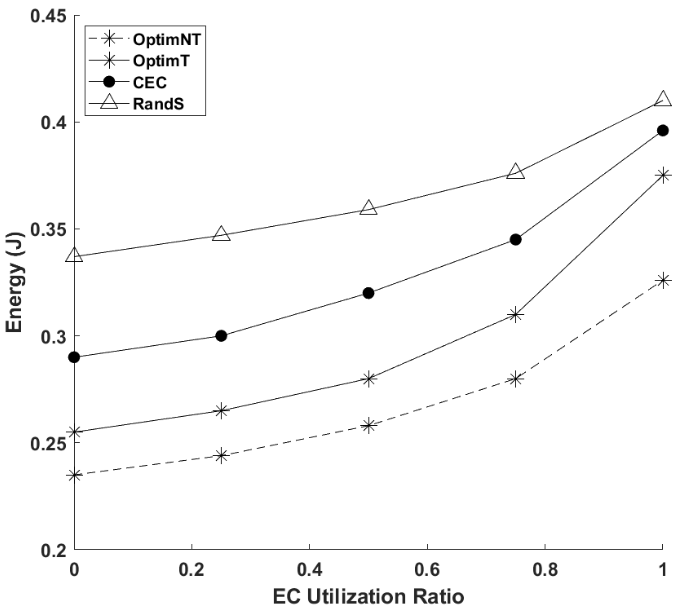


Fig. 6.10 Energy Consumption with Average EC Utilization Rate (6 EC, $\mu = 0.5$ and total mobility probability is 1)

The number of available VMs activated in an EC is known as the EC capacity. For a given EC capacity (e.g. 14), the ratio between the different number of requests (e.g. [30,40]) and the EC capacity could be applied to represent the average EC Utilization in the network. Then, this rate is normalized into $[0,1]$ for better presentation. As shown in Fig. (6.9) and (6.10), the increasing EC Utilization rate indicates a more congested network and hence as expected the delay and energy consumption increase as well. Compared to the OptimT scheme, the OptimNT scheme is still more sensitive in terms of energy but better in terms of delay. Thus, its consideration of terminals benefits delay at the cost of energy. Observe from Table (6.2), that even when there is no mobility event, the proposed OptimT scheme is still slightly better than other baseline schemes because its flexibility of terminals could better avoid potential EC overloading. Therefore, the proposed OptimT and OptimNT schemes have an obvious advantage over baseline schemes and is recommended in a congested network and a high user physical mobility scenario. Especially when the MAR terminal still owns enough energy capacity, its computing resources should not be neglected and hence the OptimT scheme is more suitable in this case.

Table 6.2 Overall Delay in no mobility event
($\mu = 1$, 6 ECs, 30 requests and EC Capacity is 14)

Scheme	**OptimT**	**OptimNT**	**CFS**	**RandS**
Delay (ms)	38.8	40.1	40.7	60.8

Extending MAR applications into the metaverse is expected to incorporate the rendering and updating of high-quality AR metadata in order to provide a more realistic experience. Hence, such forward-looking applications are highly delay- and energy-sensitive and are significantly demanding in terms of caching and computing resources. In this chapter, a joint optimization scheme is proposed by explicitly considering the model rendering performance, user mobility and service decomposition to achieve a balance between energy consumption and service delay under the constraint of user perception quality for metaverse MAR applications. Recent technical improvements in AR devices allow them to process more tasks locally. To this end, we explore their potential in the metaverse and compare the performance with terminal-oblivious schemes which reside on cloud support. A wide range of numerical investigations reveals that the proposed terminal-aware framework provides improved decision making compared to baseline schemes for energy consumption and resource allocation for metaverse MAR applications, especially under a congested network and a high-mobility scenario.

Noticing that obtaining optimal solutions is time consuming but the metaverse is quite different from normal network conditions in previous chapters, a deeper and more efficient heuristic algorithm like deep reinforcement learning could be useful. In addition, metaverse in this chapter is defined as a mixed world with features like foreground interactions and background content creation. However, metaverse is still experiencing evolutionary changes and its cooperation with internal applications is not fixed. Hence, this chapter could be a starting point for further research targeting on above limitations.

Chapter 7

Conclusions

MAR applications are still in a rather embryonic state but they are currently attracting significant attention from both academic and industry stemming from increased capabilities from the network side as well as the terminals. The ability of augmented reality services to overlay digital content over the physical space and surroundings results in an immersive metaverse type of environment which open up a plethora of potential use cases and applications.

In [42] and [46], mobility has been proved to be an important factor in service latency of MAR systems. An optimization framework is proposed that considers a rich set of AR specific constraints, provides service decomposition and allows a trade off between service latency and frame accuracy to improve the communication efficiency. Based on this scheme, a simulated annealing based mobility aware AR (SAMAR) algorithm and a LSTM based neural network for predicting edge clouds for anchoring MAR functions are designed to achieve high-quality solutions whilst being amenable for real-time implementation. Both proposed schemes deliver competitive performance and focusing on the inference stage, the LSTM based scheme shows enhanced performance and shorter execution time.

In [43], enabling content-aware caching by storing high probability 2D FoVs instead of 3D AROs could significantly ease the requirements for storage and computing resources with limited effects on overall quality. This work proposes a joint optimization framework considering the balance between service delay, field-of-view aware proactive caching and FoV allocation. Furthermore, an LSTM-based deep neural network is explored to provide real time decision making. The LSTM network is trained using optimal solutions offline, and can efficiently provide high-quality decision making in real-time during inference.

In [45] and [46], Multi view streaming applications with embedded AROs are highly delay sensitive and confronted with a series of constraints in terms of storage and computing resources. Hence, an optimization framework is proposed to realize optimal service decomposition in an edge-cloud supported 5G and beyond wireless network aiming to balance service latency and content popularity by taking into account the inherent user mobility. In addition, a nominal long short term memory (LSTM) neural network is proposed that is trained using optimal solutions to explore real-time decision making for supporting edge proactive caching for multi view streaming with popular augmented reality objects.

In [44] and [47], experiences in the metaverse are expected to be significantly demanding in terms of energy consumption, persistent high data rate support and advanced edge caching/computing capabilities in 5G and beyond networks. Through considering a more forward looking application that combines the metaverse and MAR, a joint optimization framework is proposed that explicitly considers the model rendering, user mobility and service decomposition to achieve a balance between power consumption, user perception quality and service delay for content rich metaverse type of applications. Based on whether neglecting the capacity of terminals, two versions of optimized schemes are utilized for further comparison.

According to a wide set of simulations in above works, the results show that our proposed schemes are superior to given baseline schemes and could provide a significant reduction in delay without suffering much loss in user experience or other related factors. Thus, they own an obvious advantage over baseline schemes and are suitable for MAR applications during a mobility event in the EC supported networks.

Except for some unique limitations mentioned in previous chapters, above works also share some common limitation requiring more efforts in the future. They are all simulated under a small-scale tree topology to achieve optimal solutions. However, for a large-scale dynamic network, optimal solutions are out of reach and though we try several heuristic algorithms, the quality their predictions for large networks are hard to determine since there are no optimal ones for comparison. In addition, when turning to real time decision and synchronization, it still remains as a big challenge because decision efficiency and quality are hard to maintain both for most proposed algorithms. Our research could be further tested in a more realistic scenario where mobile AR devices and applications are utilized. The proposed LSTM based algorithms could be elevated into a more complex and robust version to better fit a larger and more dynamic network. In addition, user mobility is focused in above mentioned works while current technology achievements further bring the mobility for access points with Unmanned Aerial Vehicle (UAV). In future 6G networks, such mobile events are common for mixed reality and require more advanced machine learning solutions like penetrated Artificial Intelligence in different network layers to achieve millisecond level service latency and render a fully immersive metaverse scene from multiple perspectives.

References

[1] Akhtar, N., Matta, I., Raza, A., Goratti, L., Braun, T., and Esposito, F. (2018). Virtual function placement and traffic steering over 5g multi-technology networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 114–122. IEEE.

[2] Alencar, D., Both, C., Antunes, R., Oliveira, H., Cerqueira, E., and Rosário, D. (2021). Dynamic microservice allocation for virtual reality distribution with qoe support. *IEEE Transactions on Network and Service Management*.

[3] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485.

[4] Amine, K. (2019). Multiobjective simulated annealing: Principles and algorithm variants. *Advances in Operations Research*, 2019.

[5] Avila, L. and Bailey, M. (2016). Augment your reality. *IEEE computer graphics and applications*, 36(1):6–7.

[6] Bachhuber, C., Martinez, A. S., Pries, R., Eger, S., and Steinbach, E. (2019). Edge cloud-based augmented reality. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE.

[7] Ball, M. (2022). *The metaverse: and how it will revolutionize everything*. Liveright Publishing.

[8] Bao, Y., Wu, H., Zhang, T., Ramli, A. A., and Liu, X. (2016). Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1161–1170. IEEE.

[9] Barb, G. and Otesteanu, M. (2019). On the influence of delay spread in tdl and cdl channel models for downlink 5g mimo systems. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0958–0962. IEEE.

[10] Bianchini, R., Fontoura, M., Cortez, E., Bonde, A., Muzio, A., Constantin, A.-M., Moscibroda, T., Magalhaes, G., Bablani, G., and Russinovich, M. (2020). Toward ml-centric cloud platforms. *Communications of the ACM*, 63(2):50–59.

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

[12] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[13] Chakareski, J., Velisavljevic, V., and Stankovic, V. (2015). View-popularity-driven joint source and channel coding of view and rate scalable multi-view video. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):474–486.

[14] Chatzopoulos, D., Bermejo, C., Huang, Z., and Hui, P. (2017). Mobile augmented reality survey: From where we are to where we go. *Ieee Access*, 5:6917–6950.

[15] Chen, H., Dai, Y., Meng, H., Chen, Y., and Li, T. (2018a). Understanding the characteristics of mobile augmented reality applications. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 128–138. IEEE.

[16] Chen, K., Li, T., Kim, H.-S., Culler, D. E., and Katz, R. H. (2018b). Marvel: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 292–304.

[17] Chen, S., Sun, S., and Kang, S. (2020a). System integration of terrestrial mobile communication and satellite communication —the trends, challenges and key technologies in b5g and 6g. *China Communications*, 17(12):156–171.

[18] Chen, Y., Zhang, H., and Hu, Y. (2020b). Optimal power and bandwidth allocation for multiuser video streaming in uav relay networks. *IEEE Transactions on Vehicular Technology*, 69(6):6644–6655.

[19] Chettri, L. and Bera, R. (2019). A comprehensive survey on internet of things (iot) toward 5g wireless systems. *IEEE Internet of Things Journal*, 7(1):16–32.

[20] Cho, Y. S., Kim, J., Yang, W. Y., and Kang, C. G. (2010). *MIMO-OFDM wireless communications with MATLAB*. John Wiley & Sons.

[21] Chowdhury, S. R., Salahuddin, M. A., Limam, N., and Boutaba, R. (2019). Re-architecting nfv ecosystem with microservices: State of the art and research challenges. *IEEE Network*, 33(3):168–176.

[22] Chu, N. H., Hoang, D. T., Nguyen, D. N., Phan, K. T., and Dutkiewicz, E. (2022). Metaslicing: A novel resource allocation framework for metaverse. *arXiv preprint arXiv:2205.11087*.

[23] Chylinski, M., Heller, J., Hilken, T., Keeling, D. I., Mahr, D., and de Ruyter, K. (2020). Augmented reality marketing: A technology-enabled approach to situated customer experience. *Australasian Marketing Journal (AMJ)*, 28(4):374–384.

[24] Cozzolino, V., Tonetto, L., Mohan, N., Ding, A. Y., and Ott, J. (2022). Nimbus: Towards latency-energy efficient task offloading for ar services. *IEEE Transactions on Cloud Computing*.

[25] Cziva, R. and Pezaros, D. P. (2017). On the latency benefits of edge nfv. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 105–106. IEEE.

[26] Dang, T. and Peng, M. (2019). Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks. *IEEE Journal on Selected Areas in Communications*, 37(7):1594–1607.

[27] De Lauretis, L. (2019). From monolithic architecture to microservices architecture. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 93–96. IEEE.

[28] Dong, R., She, C., Hardjawana, W., Li, Y., and Vucetic, B. (2019). Deep learning for hybrid 5g services in mobile edge computing systems: Learn from a digital twin. *IEEE Transactions on Wireless Communications*, 18(10):4692–4707.

[29] Elbadawy, H. M. and Sadek, R. A. (2022). B5g/6g service planning process for the deployment of smart cities as a model for massive urban area applications. In *2022 39th National Radio Science Conference (NRSC)*, volume 1, pages 332–341.

[30] Fausto, F., Reyna-Orta, A., Cuevas, E., Andrade, Á. G., and Perez-Cisneros, M. (2020). From ants to whales: metaheuristics for all tastes. *Artificial Intelligence Review*, 53(1):753–810.

[31] Gemici, Ö. F., Hökelek, İ., and Çırpan, H. A. (2021). Modeling queuing delay of 5g nr with noma under sinr outage constraint. *IEEE Transactions on Vehicular Technology*, 70(3):2389–2403.

[32] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proc. 13th international conference on artificial intelligence and statistics*, pages 249–256.

[33] Guo, F., Yu, F. R., Zhang, H., Ji, H., Leung, V. C., and Li, X. (2020a). An adaptive wireless virtual reality framework in future wireless networks: A distributed learning approach. *IEEE Transactions on Vehicular Technology*, 69(8):8514–8528.

[34] Guo, Y., Wang, S., Zhou, A., Xu, J., Yuan, J., and Hsu, C.-H. (2020b). User allocation-aware edge cloud placement in mobile edge computing. *Software: Practice and Experience*, 50(5):489–502.

[35] Halabian, H. (2019). Optimal distributed resource allocation in 5g virtualized networks. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 28–35. IEEE.

[36] Han, Y., Niyato, D., Leung, C., Miao, C., and Kim, D. I. (2021). A dynamic resource allocation framework for synchronizing metaverse with iot service and data. *arXiv preprint arXiv:2111.00431*.

[37] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE intern. conference on computer vision*.

[38] Herrera, J. G. and Botero, J. F. (2016). Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532.

[39] Hertzmann, A. and Perlin, K. (2000). Painterly rendering for video and interaction. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12.

[40] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[41] Huang, X., Wang, L., Huang, J., Li, D., and Zhang, M. (2009). A depth extraction method based on motion and geometry for 2d to 3d conversion. In *2009 Third International Symposium on Intelligent Information Technology Application*, volume 3, pages 294–298. IEEE.

[42] Huang, Z. and Friderikos, V. (2021). Proactive edge cloud optimization for mobile augmented reality applications. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

[43] Huang, Z. and Friderikos, V. (2022a). Field of view aware proactive caching for mobile augmented reality applications. In *2022 IEEE Global Communications Conference*, pages 1–6. IEEE.

[44] Huang, Z. and Friderikos, V. (2022b). Mobility aware optimization in the metaverse. In *2022 IEEE Global Communications Conference*, pages 1–6. IEEE.

[45] Huang, Z. and Friderikos, V. (2022c). Network resource optimization for multi-view streaming mobile augmented reality. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–7. IEEE.

[46] Huang, Z. and Friderikos, V. (2022d). Optimal proactive caching for multi-view streaming mobile augmented reality. *Future Internet*, 14(6):166.

[47] Huang, Z. and Friderikos, V. (2023a). Optimal mobility aware wireless edge cloud support for the metaverse. *Future Internet*, 15(2):47.

[48] Huang, Z. and Friderikos, V. (2023b). Optimal service decomposition for mobile augmented reality with edge cloud support. *Computer Communications*. ISSN:0140-3664, doi:https://doi.org/10.1016/j.comcom.2023.02.002.

[49] Huang, Z., Friderikos, V., Dohler, M., and Aghvami, H. (2021). Granular vnf-based microservices: Advanced service decomposition and the role of machine learning techniques. In *Design Innovation and Network Architecture for the Future Internet*, pages 250–271. IGI Global.

[50] Huang, Z., Li, W., Hui, P., and Peylo, C. (2014). Cloudridar: A cloud-based architecture for mobile augmented reality. In *Proceedings of the 2014 workshop on Mobile augmented reality and robotic technology-based systems*, pages 29–34.

[51] Huynh, L. N., Lee, Y., and Balan, R. K. (2017). Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95.

[52] Jain, P., Manweiler, J., and Roy Choudhury, R. (2015). Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 331–344.

[53] Jalodia, N., Henna, S., and Davy, A. (2019). Deep reinforcement learning for topology-aware vnf resource prediction in nfv environments. In *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–5. IEEE.

[54] Jiang, Y., Kang, J., Niyato, D., Ge, X., Xiong, Z., and Miao, C. (2021). Reliable coded distributed computing for metaverse services: Coalition formation and incentive mechanism design. *arXiv preprint arXiv:2111.10548*.

[55] Kan, N., Zou, J., Tang, K., Li, C., Liu, N., and Xiong, H. (2019). Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4030–4034. IEEE.

[56] Karmel, A., Chandramouli, R., and Iorga, M. (2016). Nist definition of microservices, application containers and system virtual machines. Technical report, National Institute of Standards and Technology.

[57] Kato, H., Kobayashi, T., Sugano, M., and Naito, S. (2021). Split rendering of the transparent channel for cloud ar. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing*, pages 1–6. IEEE.

[58] Kaur, K., Mangat, V., and Kumar, K. (2020). A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture. *Computer Science Review*, 38:100298.

[59] Korrai, P. K., Lagunas, E., Sharma, S. K., Chatzinotas, S., and Ottersten, B. (2019). Slicing based resource allocation for multiplexing of embb and urllc services in 5g wireless networks. In *IEEE CAMAD*.

[60] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[61] Laghrissi, A. and Taleb, T. (2018). A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys & Tutorials*, 21(2):1409–1434.

[62] Li, L., Qiao, X., Lu, Q., Ren, P., and Lin, R. (2020a). Rendering optimization for mobile web 3d based on animation data separation and on-demand loading. *IEEE Access*, 8:88474–88486.

[63] Li, S., Lin, P., Song, J., and Song, Q. (2020b). Computing-assisted task offloading and resource allocation for wireless vr systems. In *IEEE 6th International Conference on Computer and Communications (ICCC)*.

[64] Liu, L., Li, H., and Gruteser, M. (2019). Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16.

[65] Liu, M. and Liu, Y. (2017). Price-based distributed offloading for mobile-edge computing with computation capacity constraints. *IEEE Wireless Communications Letters*, 7(3):420–423.

[66] Liu, Q. and Han, T. (2018). Dare: Dynamic adaptive mobile augmented reality with edge computing. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE.

[67] Liu, Q., Huang, S., Opadere, J., and Han, T. (2018). An edge network orchestrator for mobile augmented reality. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 756–764. IEEE.

[68] Liu, Z., Xiang, Y., and Qu, X. (2015). Towards optimal cpu frequency and different workload for multi-objective vm allocation. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 367–372. IEEE.

[69] Miettinen, A. P. and Nurminen, J. K. (2010). Energy efficiency of mobile clients in cloud computing. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*.

[70] Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2015). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials*, 18(1):236–262.

[71] Minaee, S., Liang, X., and Yan, S. (2022). Modern augmented reality: Applications, trends, and future directions. *arXiv preprint arXiv:2202.09450*.

[72] Naman, A. T., Xu, R., and Taubman, D. (2013). Inter-frame prediction using motion hints. In *2013 IEEE International Conference on Image Processing*, pages 1792–1796. IEEE.

[73] Nath, S. B., Chattopadhyay, S., Karmakar, R., Addya, S. K., Chakraborty, S., and Ghosh, S. K. (2019). Ptc: Pick-test-choose to place containerized micro-services in iot. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

[74] Ng, W. C., Lim, W. Y. B., Ng, J. S., Xiong, Z., Niyato, D., and Miao, C. (2021). Unified resource allocation framework for the edge intelligence-enabled metaverse. *arXiv preprint arXiv:2110.14325*.

[75] Niu, G. and Chen, Q. (2018). Learning an video frame-based face detection system for security fields. *Journal of Visual Communication and Image Representation*, 55:457–463.

[76] Patel, P., Ranabahu, A. H., and Sheth, A. P. (2009). Service level agreement in cloud computing. *Technical report, Wright State University*.

[77] Patoommakesorn, K., Vignat, F., and Villeneuve, F. (2019). The 3d edge reconstruction from 2d image by using correlation based algorithm. In *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 372–376. IEEE.

[78] Perronnin, F., Liu, Y., Sánchez, J., and Poirier, H. (2010). Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3384–3391. IEEE.

[79] Pervez, F., Raheel, M. S., Gill, J., and Iqbal, K. (2016). Uplink resource allocation of multiple dash streams for qoe-based real and non-real time viewing over lte. In *10th International Conference on Next Generation Mobile Applications, Security and Technologies (NGMAST)*.

[80] Qiao, X., Ren, P., Dustdar, S., Liu, L., Ma, H., and Chen, J. (2019a). Web ar: A promising future for mobile augmented reality—state of the art, challenges, and insights. *Proceedings of the IEEE*, 107(4):651–666.

[81] Qiao, X., Ren, P., Nan, G., Liu, L., Dustdar, S., and Chen, J. (2019b). Mobile web augmented reality in 5g and beyond: Challenges, opportunities, and future directions. *China Communications*, 16(9):141–154.

[82] Ran, X., Chen, H., Zhu, X., Liu, Z., and Chen, J. (2018). Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1421–1429. IEEE.

[83] Rattanarungrot, S. and White, M. (2016). Development of service oriented mobile ar applications for museum learning activities. In *2016 22nd International Conference on Virtual System & Multimedia (VSMM)*, pages 1–8. IEEE.

[84] Rauschnabel, P. A., Felix, R., and Hinsch, C. (2019). Augmented reality marketing: How mobile ar-apps can improve brands through inspiration. *Journal of Retailing and Consumer Services*, 49:43–53.

[85] Rauschnabel, P. A., Felix, R., Hinsch, C., Shahab, H., and Alt, F. (2022). What is xr? towards a framework for augmented and virtual reality. *Computers in Human Behavior*, 133:107289.

[86] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE conference on computer vision and pattern recognition*, pages 779–788.

[87] Ren, J., Gao, L., Wang, X., Ma, M., Qiu, G., Wang, H., Zheng, J., and Wang, Z. (2021). Adaptive computation offloading for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–30.

[88] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[89] Saarnisaari, H., Laiyemo, A. O., and de Lima, C. H. (2019). Random access process analysis of 5g new radio based satellite links. In *2019 IEEE 2nd 5G World Forum (5GWF)*, pages 654–658. IEEE.

[90] Selmadji, A., Seriai, A.-D., Bouziane, H. L., Mahamane, R. O., Zaragoza, P., and Dony, C. (2020). From monolithic architecture style to microservice one based on a semi-automatic approach. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 157–168. IEEE.

[91] Seo, Y.-J., Lee, J., Hwang, J., Niyato, D., Park, H.-S., and Choi, J. K. (2021). A novel joint mobile cache and power management scheme for energy-efficient mobile augmented reality service in mobile edge computing. *IEEE Wireless Communications Letters*, 10(5):1061–1065.

[92] Shea, R., Sun, A., Fu, S., and Liu, J. (2017). Towards fully offloaded cloud-based ar: Design, implementation and experience. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 321–330.

[93] Shu, H. and Chau, L.-P. (2005). Frame size selection in video downsizing transcoding application. In *2005 IEEE International Symposium on Circuits and Systems*, pages 896–899. IEEE.

[94] Song, S., Kim, J., and Chung, J.-M. (2019). Energy consumption minimization control for augmented reality applications based on multi-core smart devices. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4. IEEE.

[95] Sonmez, C., Ozgovde, A., and Ersoy, C. (2019). Fuzzy workload orchestration for edge computing. *IEEE Transactions on Network and Service Management*, 16(2):769–782.

[96] Speicher, M., Hall, B. D., Yu, A., Zhang, B., Zhang, H., Nebeling, J., and Nebeling, M. (2018). Xd-ar: challenges and opportunities in cross-device augmented reality application development. *Proceedings of the ACM on Human-Computer Interaction*, 2(EICS):1–24.

[97] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[98] Sun, C., Zhou, J., Liuliang, J., Zhang, J., Zhang, X., and Wang, W. (2018a). Computation offloading with virtual resources management in mobile edge networks. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

[99] Sun, G., Zhu, G., Liao, D., Yu, H., Du, X., and Guizani, M. (2018b). Cost-efficient service function chain orchestration for low-latency applications in nfv networks. *IEEE Systems Journal*, 13(4):3877–3888.

[100] Sun, Y., Chen, Z., Tao, M., and Liu, H. (2019). Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff. *IEEE Transactions on Communications*, 67(11):7573–7586.

[101] Toczé, K. and Nadjm-Tehrani, S. (2019). Orch: Distributed orchestration framework using mobile edge devices. In *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–10. IEEE.

[102] Tsubaki, I., Shimeno, A., Tsukuba, T., Suenaga, T., and Shioi, M. (2012). 2d to 3d conversion based on tracking both vanishing point and objects. In *The 1st IEEE Global Conference on Consumer Electronics 2012*, pages 110–114. IEEE.

[103] Van Houdt, G., Mosquera, C., and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8):5929–5955.

[104] Wang, X., Yang, L. T., Xie, X., Jin, J., and Deen, M. J. (2017a). A cloud-edge computing framework for cyber-physical-social services. *IEEE Communications Magazine*, 55(11):80–85.

[105] Wang, Y., Haenggi, M., and Tan, Z. (2017b). The meta distribution of the sir for cellular networks with power control. *IEEE Transactions on Communications*, 66(4):1745–1757.

[106] Wang, Y., Tao, X., Zhang, X., Zhang, P., and Hou, Y. T. (2019). Cooperative task offloading in three-tier mobile computing networks: An admm framework. *IEEE Transactions on Vehicular Technology*, 68(3):2763–2776.

[107] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

[108] Wu, C., Wang, M., Ma, B., and Chen, K. (2020). Heterogeneous networks topology optimization based on simulated annealing algorithm. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 2074–2078. IEEE.

[109] Wu, C.-C., Tung, L.-P., Lin, C.-Y., Lin, B.-S. P., and Tseng, Y.-C. (2014). On local cache management strategies for mobile augmented reality. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–3. IEEE.

[110] Xie, R., Wu, J., Wang, R., and Huang, T. (2019). A game theoretic approach for hierarchical caching resource sharing in 5g networks with virtualization. *China Communications*, 16(7):32–48.

[111] Xu, M., Ng, W. C., Lim, W. Y. B., Kang, J., Xiong, Z., Niyato, D., Yang, Q., Shen, X. S., and Miao, C. (2022). A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges. *arXiv preprint arXiv:2203.05471*.

[112] Xu, M., Niyato, D., Kang, J., Xiong, Z., Miao, C., and Kim, D. I. (2021). Wireless edge-empowered metaverse: A learning-based incentive mechanism for virtual reality. *arXiv preprint arXiv:2111.03776*.

[113] Yang, X., Chen, Z., Li, K., Sun, Y., Liu, N., Xie, W., and Zhao, Y. (2018). Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff. *IEEE Access*, 6:16665–16677.

[114] Zeng, X., Cao, K., and Zhang, M. (2017). Mobiledeeppill: A small-footprint mobile deep learning system for recognizing unconstrained pill images. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 56–67.

[115] Zhang, A., Jacobs, J., Sra, M., and Höllerer, T. (2021a). Multi-view ar streams for interactive 3d remote teaching. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–3.

[116] Zhang, R., Liu, J., Liu, F., Huang, T., Tang, Q., Wang, S., and Yu, F. R. (2021b). Buffer-aware virtual reality video streaming with personalized and private viewport prediction. *IEEE JSAC*.

[117] Zhang, W., Han, B., Hui, P., Gopalakrishnan, V., Zavesky, E., and Qian, F. (2018a). Cars: Collaborative augmented reality for socialization. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, pages 25–30.

[118] Zhang, W., Wen, Y., Guan, K., Kilper, D., Luo, H., and Wu, D. O. (2013). Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9):4569–4581.

[119] Zhang, X., Fronz, S., and Navab, N. (2002). Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings. International Symposium on Mixed and Augmented Reality*, pages 97–106. IEEE.

[120] Zhang, X., Toni, L., Frossard, P., Zhao, Y., and Lin, C. (2018b). Adaptive streaming in interactive multiview video systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(4):1130–1144.

[121] Zhang, Z., Yang, Y., Hua, M., Li, C., Huang, Y., and Yang, L. (2019). Proactive caching for vehicular multi-view 3d video streaming via deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 18(5):2693–2706.

[122] Zhao, L., Cui, Y., Liu, Z., Zhang, Y., and Yang, S. (2021). Adaptive streaming of 360 videos with perfect, imperfect, and unknown fov viewing probabilities in wireless networks. *IEEE Transactions on Image Processing*.

[123] Zhao, M., Gong, X., Liang, J., Guo, J., Wang, W., Que, X., and Cheng, S. (2015). A cloud-assisted dash-based scalable interactive multiview video streaming framework. In *2015 Picture Coding Symposium (PCS)*, pages 221–226. IEEE.

[124] Zheng, G., Tsiopoulos, A., and Friderikos, V. (2018). Optimal vnf chains management for proactive caching. *IEEE Transactions on Wireless Communications*, 17(10):6735–6748.

[125] Zuo, Y., Wu, Y., Min, G., and Cui, L. (2019). Learning-based network path planning for traffic engineering. *Future Generation Computer Systems*, 92:59–67.