Technische Universität Dresden

Computing on the Edge of the Network

M. Sc.

Mahshid Mehrabi

der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender: Prof. Dr.-Ing. habil. Leon Urbas
Gutachter: Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek
Gutachter: Prof. Dr. Frank Y. Li
Gutachter: Prof. Dr. Antonio Iera

Tag der Einreichung: 18.05.2021
Tag der Verteidigung: 10.03.2022

Abstract

Enabling Fifth Generation of Cellular Communication Networks (5G) systems requires energy efficient architectures which can provide a reliable service platform to deliver 5G services and beyond. Device-enhanced edge computing is a derivation of Multi-access edge computing (MEC), which provides computing and storage resources very on the end-devices. The importance of this concept has been proved by the rising demands of computation-intensive and ultra-low latency applications which overwhelm the MEC server and the wireless channel. This dissertation presents a computation offloading framework with energy-, mobility-, and incentive-awareness in a multiple-user multiple-task device-enhanced MEC system which considers the interdependency of tasks as well as latency requirements of the applications.

Um Systeme der fünften Generation zellularer Kommunikationsnetze (5G) zu ermöglichen, sind energieeffiziente Architekturen erforderlich, die eine zuverlässige Serviceplattform für die Bereitstellung von 5G-Diensten und darüber hinaus bieten können. Device-enhanced Edge Computing ist eine Ableitung des Multi-Access Edge Computing (MEC), das Rechen- und Speicherressourcen direkt auf den Endgeräten bereitstellt. Die Bedeutung dieses Konzepts wird durch die steigenden Anforderungen von rechenintensiven Anwendungen mit extrem niedriger Latenzzeit belegt, die den MEC-Server allein und den drahtlosen Kanal überfordern. Diese Dissertation stellt ein Berechnungs-Auslagerungsframework mit Berücksichtigung von Energie, Mobilität und Anreizen in einem gerätegestützten MEC-System mit mehreren Benutzern und mehreren Aufgaben vor, das die gegenseitige Abhängigkeit der Aufgaben sowie die Latenzanforderungen der Anwendungen berücksichtigt.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Dr.-Ing. Dr. h.c. Frank Hanns Paul Fitzek, head of the department of the Deutsche Telekom Chair of Communication Networks at Technische Universität Dresden who gave me the possibility to pursue my Ph.D. in the field of communication networks and be part of the European Union's H2020 SECRET project where I have learnt a lot.

I would like to acknowledge my friends and colleagues in the Deutsche Telekom Chair of Communication Networks and the European Union's H2020 SECRET project for supporting me during my PhD era and international experiences. A very special thanks to my colleagues Jan Vincent Latzko and Shiwei Shen for their constructive collaboration.

Last but not least, I would especially like to thank my family. Moving to Germany was the hardest and most important decision I have ever made in my entire life and I am aware of the pressure they had to cope with after my immigration. I especially would like them to know I love them and there are not enough space in this dissertation to write about how much I care about them.

Statement of authorship

I hereby certify that I have authored this thesisindependently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Contents

List of Figures

List of Tables

1 Introduction

Part of the content of this chapter was previously published in:
Mehrabi, Mahshid, et al. "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey." *IEEE Access 7* (2019): 166079-166108 [2].

Enabling Fifth Generation of Cellular Communication Networks (5G) systems requires efficient architectures that will provide a service platform on which to deliver 5G services and beyond. The SECRET project aims to integrate various technologies such as virtualization and Network-coded Cooperation (NCC) as a means to provide a fully flexible and efficient networking approach to provide cost-effective packet delivery [6]. Virtualization has changed our perspective of the network, providing technology tools to enable fully dynamic networks that can break the rigid boundaries of current architectures. In fact, the thesis scenario given by Figure 1.1, demonstrates how concepts such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) introduce the notion of a virtualized Mobile Small Cell (MSC).

However, the key challenge is how to exploit this technology to deliver efficient 5G services and beyond, that have stringent Quality of Service (QoS) requirements in terms of reduced latency and power consumption. Key technologies for 5G and beyond involve task offloading and content caching in a bid to reduce the service delivery time. Content caching aims to exploit the extensive storage capacities in MEC in order to increase the throughput of video files without introducing additional infrastructure cost, whilst task offloading services aims to reduce the power consumption

and service delivery time by eventually enabling the vision of truly virtualized handsets. Although these topics have been well investigated as *isolated* technologies, there are only few works that show how these can be performed within a mobile small cell context that exploits MEC nodes and virtualized mobile small cells.

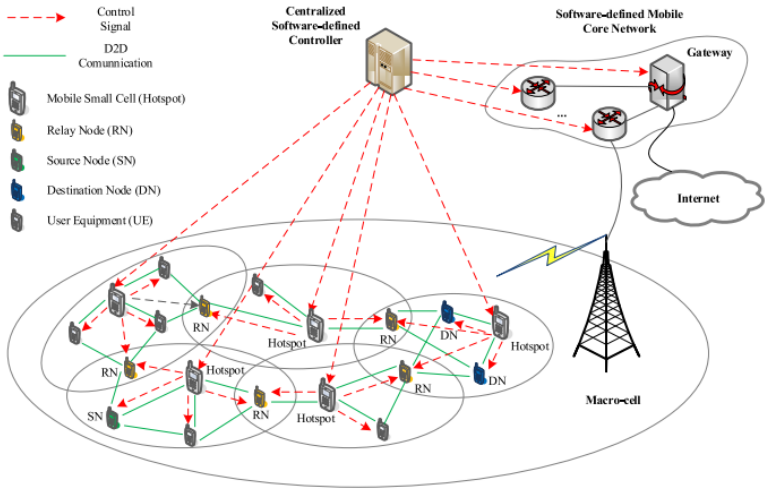


Figure 1.1: Secret scenario architecture [1].

Concurrently, the recent advancement of Central Processing Unit (CPU) built into mobile devices along with D2D communications have enabled the usage of computation and storage resources of adjacent devices. This can be considered as a proper solution for problems caused by ever increasing internet traffic such as breakdown or edge servers overloading which can be even more problematic for applications require ultra-low latency such as Tactile Internet (TI) and Internet of Things (IoT) for robotic control applications [7], online gaming, Augmented Reality (AR) or Virtual Reality (VR). This new computing era which we have coined device-enhanced MEC has more advantages in terms of service coverage, spectral efficiency, mobile devices' battery savings beside the aimed-for latency reduction.

In that context, this thesis addresses the current problems of computation offloading for 5G device-enhanced MEC. In the following, the thesis context and motivation are stated. In Section 1.2, key issues of computation offloading in device-enhance MEC systems are described. The thesis contribution is stated in Section 1.3 and finally, Section 1.4 presents the outline of this thesis.

1.1 Context and motivations

The concept of MEC was firstly proposed by the European Telecommunications Standards Institute (ETSI) as "a new platform to provide IT and Cloud computing capabilities within the Radio Access Network (RAN) in close proximity to mobile subscribers" [8]. The original concept refers to the use of the Base Station (BS) to offload the computation tasks of mobile devices. In comparison to the Mobile Cloud Computing (MCC) which brings rich computational resources to mobile devices, MEC has some advantages in terms of latency, energy for mobile users, context-aware computing and mobile applications security.

Achieving ultra-low latency while saving mobile devices' battery life times is one of the big challenges in the design of the front-haul, where we strive to attain this goal by considering the network as a cloud service, based on conceptual tool such as edge computing and using adjacent mobile devices' resources through D2D communication.

1.1.1 Device-enhanced MEC concept

The fifth generation of cellular mobile communications is an era which targets massive device connectivity with high data rate and reduced latency and cost. The MEC paradigm, has been introduced to bring computational and storage resources in close proximity of the mobile end devices; however, the rapid increase of the number of mobile devices as well as the resulting overall rising Internet traffic along with the computation and storage demands of new emerging services, such as online gaming, IoT, virtual or augmented reality, may overwhelm the installed MEC servers. Due to the cost pressures in the telecommunication industry for installation of higher and higher powered MEC servers, and recent developments of the CPUs of mobile devices, a possible solution can be using adjacent mobile devices' computation and storage capacities for providing services. This resource sharing among mobile devices can be provided using D2D communication. The D2D communication holds significant promise for a variety of practical use-cases such as proximity and local-based services, Vehicle to Vehicle (V2V), among others. In proximity and local-based services, like video stream services in the football stadiums, or concert halls, D2D communication can facilitate connections between different adjacent users to store and share video files and images. In V2V communication it can improve public safety

and intelligent transportation systems [2]. In the following subsection, we will introduce the key challenges of computation offloading in device-enhanced MEC systems.

1.1.2 Key issues of computation offloading in device-enhanced MEC systems

In order to fulfill the low-latency requirements of applications and save the battery life of mobile devices, end devices can offload their computation intensive tasks to more powerful MEC servers or adjacent users using D2D communication links. This can reduce the load on the cellular network infrastructure and usage of cellular bandwidth.

The primary objectives of the existing studies on enhancing MEC computation offloading with end devices are the minimization of latency and energy consumption. The type of application plays an important role in making offloading decisions. Most existing studies have considered computation tasks to be either completely partitionable and non-partitionable, depending on the application scenario; here, individual sub-tasks of partitionable tasks or the non-partitionable tasks can be executed locally depending on the computation resource amount of device and whether the latency requirement of task can be tolerated, offloaded to adjacent User Equipment (UE)s directly via D2D communication or via relays, or offloaded to an MEC server. However, in many IoT scenarios, a data dependency between different IoT sensors' tasks exists for services to be completed, which has been neglected in the surveyed articles. In these kind of services, some information from other tasks are needed to execute the subsequent next tasks.

Furthermore, previous studies mostly consider devices as static; however, in real scenarios, the end devices are mobile. Therefore, in this thesis, we focus on the challenges that mobility, along with increasing the density of devices in the small cells, impact our scenario and try to achieve energy efficiency for our delay-sensitive tasks by considering the aforementioned issues. Mobility of end devices can bring several challenges for service continuity and QoS in MEC systems. User movements can lead to the interruption of D2D links and this will likely increase the latencies and higher battery energy consumption. Therefore, a key prerequisite for enabling satisfactory services is updating the availability and reliability of computation resources. Users' movements can happen either by the requester (the user with computation-

intensive tasks), relays/helpers (the users which share their resources).

In addition, in a realistic cooperation computation offloading scenario, an incentive mechanism should be defined to pursue mobile devices to share their computation or storage resources. We have therefore developed and validated an effective method for providing reliable computation resources for dynamic network scenarios in this dissertation.

In Section 1.2, we briefly describe our contributions in this thesis to solve the aforementioned challenges and problems.

1.2 Thesis contribution

In this section, we summarize the significant contributions of this thesis:

1.2.1 Proposed an optimal edge offloading in a basic three-node device-enhanced MEC system

In Chapter 3, we propose an optimal computation offloading in a basic three- node device-enhanced MEC system as our first contribution. This contribution was published in 2020 IEEE Global Communications Conference, Selected Areas in Communications: Cloud and Fog/Edge Computing, Networking and Storage conference [3]. In this work, we considered a three- tier network consisting of two UEs called UE1 and UE2. UE1 has a computation task following sequential dependency graph and UE2 can play either a helper or relay role and a MEC server node is attached to a BS. In this scenario, tasks can be executed cooperatively using D2D communication. Minimizing the total energy consumption of the device and fulfilling the time deadline constraints of the tasks are our two performance metrics.

The previous studies on computation offloading in device-enhanced MEC networks mostly considered the tasks independently, however, in some IoT applications such as the distributed Ibis application presented in the [9] there exist inter-task dependency relationships. The inter-task dependency challenges for computation offloading in MEC systems investigated in [10, 11]; However, to the best of our knowledge, our joint computation and communication cooperation offloading using D2D communication method was the first work which considered task dependencies in a device-enhanced MEC network. Simulation results show that our proposed method

can save up to 65.47% of energy when compared with on device execution of the tasks and 49.29% when compared with an all server execution strategy for complex tasks scenarios.

1.2.2 Proposed an optimal dynamic computation offloading using a deep learning based mobility prediction algorithm in device-enhanced MEC systems

In Chapter 4, we propose a dynamic computation offloading in a three-tiers device-enhanced MEC system as our second contribution. This contribution was published in 2021 Network Journal [4].
Many previous studies consider a static scenario while mobility of users can cause serious problems for service continuity as well as establishing D2D and cellular links. In order to ensure the service continuity and reliability of computation resources, an online computation offloading framework is proposed considering the dependency relationships between computation tasks.
In this article, there is one UE with computation intensive tasks and multiple helper nodes as well as multiple MEC servers with their respective coverage areas are distributed in the network. The users are mobile, meaning that they can leave the coverage of one small cell and enters the others'. The mobility effects of users in this work has been captured by sojourn time concept which is obtained by the Deep Learning (DL) algorithm PECNet [5]. PECNet uses the historical data of motion path as well as their social interactions.
By introducing the sojourn time concept, there is no handover and service migration during small cell changes, and therefore the extra delay and energy consumption of handover are prevented. Minimizing the total energy consumption for the service execution while satisfying the time deadline constraints of the tasks is the main goal of this work. Simulation results show that our proposed method can save up to 56.34% of energy when compared with on device execution of the tasks and 33.73% when compared with an all server execution strategy for complex tasks scenarios.

1.2.3 Proposed an optimal dynamic and incentive-aware multi-users multi-tasks computation offloading in a device-enhanced MEC system

In Chapter 5, we extended our dynamic scenario to an incentive-aware computation offloading with multi-user multi-task scenario to satisfy the fact that users should be pursued to share their computation and communication resources. Therefore, the proposed computation offloading algorithm in this chapter is a more practical scenario in real world.
In this work, there are several numbers of users which each has to perform an application which is divided into K fine- grained inter-dependent tasks. The mobility model in this section is same as our previous work.
By using resource tit-for-tat and an energy budget constraints [12], we introduce our incentive method. In this work, we have two types of resource constraint namely, computing cycles and cellular bandwidth which account for computation and communication resources, respectively. Our computation offloading decision algorithm's goal is to minimize the sum of energy consumption and the finish time of the tasks. Therefore, our cost function is defined based on the weighted combination of the energy consumption and finish time for executing the tasks. Simulation results show that our proposed method can save up to 56.88% of energy when compared with on device execution of the tasks and 44.82% when compared with an all server execution strategy for complex tasks scenarios.

1.3 Thesis outline

The thesis as a whole is organized as follows. Chapter 2 includes the background and literature review of device-enhanced MEC systems. In Chapter 3, we consider a basic three-node system and present an optimal computation offloading method for inter-dependent and latency critical tasks scenarios. Chapter 4 contains our contribution towards dynamic computation offloading with users' mobility. By taking into account the mobility effects of the users by the sojourn time concept and using a DL algorithm, we predict realistic, socially compliant trajectories and users' destinations. The problem is then formulated as an energy usage minimization optimization while

satisfying the task dependencies and the completion deadlines and solved using a semidefinite relaxation approach. Then, in Chapter 5, we extend the work of Chapter 4 to a more realistic scenario and introduce an incentivization mechanism for cooperations in the offloading algorithm.

The goal is formulated to minimize the sum of energy consumption and the finish time of the tasks as a Mixed Integer Nonlinear Programming (MINLP) and a genetic algorithm is used to solve the optimization problem. Finally, the conclusion and our future research directions are stated in Chapter 6.

2 Background and Literature Review

The content of this chapter was previously published in:
Mehrabi, Mahshid, et al. "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey." *IEEE Access 7* (2019): 166079-166108 [2].

2.1 Introduction

The MEC paradigm, which is also known as Mobile Edge Computing, has been introduced to bring computing and storage resources in close physical proximity of the wireless end devices [13, 14]. For instance, MEC resources can be co-located with the BSs or back-haul entities of cellular wireless communications [15], as illustrated in Figure 2.1. The MEC thus helps to provide low-latency services requiring intensive computations or large data volumes to mobile wireless end devices [16–18]. The number of wireless end devices, such as UE nodes in cellular wireless networks, is expected to further grow and continue to substantially contribute to the overall Internet traffic growth [19]. Also, the computing and data demands of the wireless end devices are projected to grow substantially over the coming years. This growth is in part due to newly emerging service paradigms, such as the TI [20] requiring millisecond latency responsiveness, e.g., for robotic control applications, the IoT [21] connecting enormous numbers of devices, Machine Type Communication (MTC) [22], online gaming, as well as virtual or augmented reality. The increasing computing and

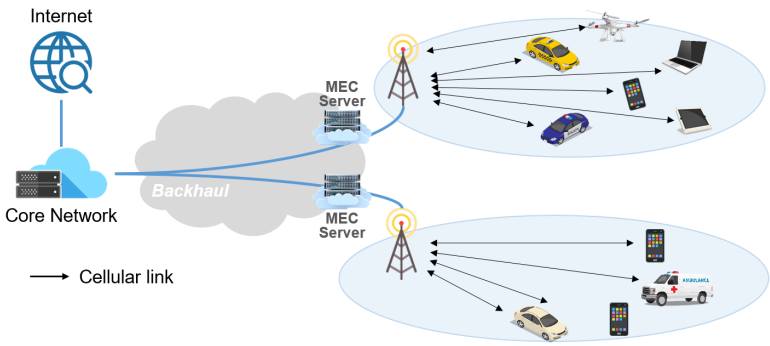


Figure 2.1: Illustration of MEC systems.

data demands due to these emerging services, which will likely be utilized by large numbers of wireless end devices, may overwhelm the installed MEC computing and storage infrastructure. Moreover, cost pressures in the telecommunication industry may limit the installation of ever-increasing MEC compute and storage capacities, as clearly this approach does not scale well.

A possible solution to this dilemma is to utilize the increasingly powerful processing units, e.g., CPUs and special-purpose processing units, and increasing storage capacities of modern wireless end devices for providing services. That is, the community of wireless end devices, which is also referred to as mobile device cloud, can contribute its aggregate computing and storage resources to provide services to individual end devices jointly with the MEC. Effectively, the end devices share their resources and collaborate with the MEC to quickly provide compute- and data-intensive services to their fellow end devices. This sharing among end devices is facilitated by recent advances in D2D communication [23–26]. D2D communication enables an end device to exploit the resources of the end devices in its proximity via direct (D2D) connections, as illustrated by the red links in the right half of Figure 2.2; thus, the traffic load on the cellular network and MEC infrastructure is potentially reduced. The collaboration of (*i*) the MEC, which has installed resources up to the BSs, with (*ii*) the sharing of resources among end devices, which is enabled through D2D communication, gives rise to the paradigm of *device-enhanced MEC*. As illustrated in Figure 2.3, device-enhanced MEC encompasses conventional MEC and D2D communication enabled end device resource sharing and thus extends across the entire scope of Figure 2.3.

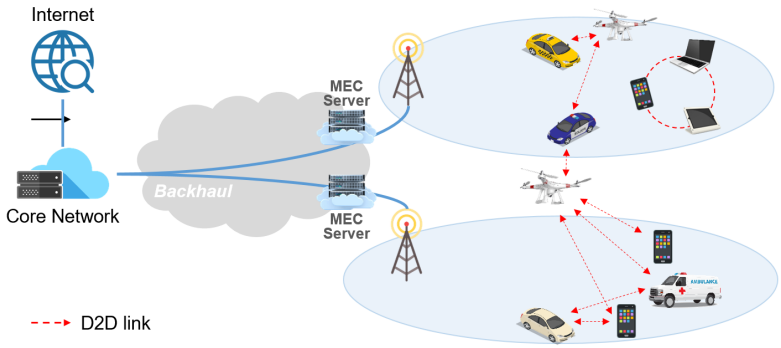


Figure 2.2: Illustration of D2D communication between devices.

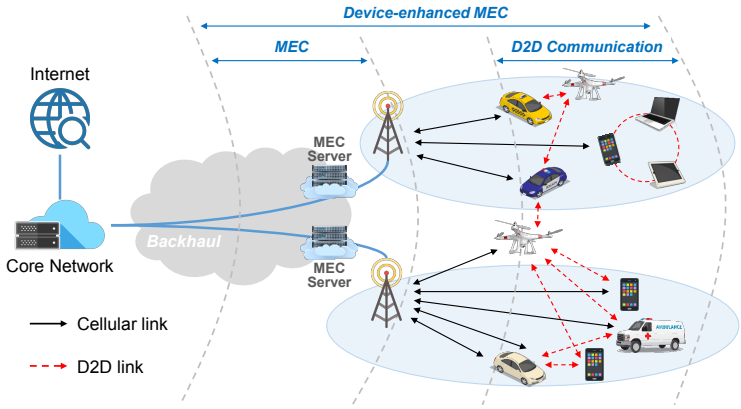


Figure 2.3: Illustration of device-enhanced MEC systems [2].

2.1.1 Emergence of MEC

The demands of popular applications running on mobile end devices have brought several challenges for network operators. The limited battery lifetimes as well as the limited computational and storage resources of mobile end devices have motivated network operators to modify their existing infrastructures. The MCC paradigm was introduced to extend cloud computing features to mobile end devices with the aim of centralizing the management of the computational and storage resources in the core network [27–30]. The MCC benefits mobile end devices by expanding the available computation and storage resources as well as the flexibility to support multiple platforms.

However, the MCC fails to fulfill the low-latency requirements of emerging mobile applications due to the long distances to the devices and the back-haul bandwidth

limitations [31]. To tackle this problem, computing and storage resources should be placed as close as possible to the mobile end devices, e.g., by deploying cloud servers inside cellular BSs or Access Point (AP) depending on the network architecture. This trend of deploying cloud servers close to the mobile end devices was initially called Mobile Edge Computing and standardized by the ETSI Industry Specification Group (ISG). In order to extend the MEC usage to heterogeneous networks technologies, e.g, Wireless Local Area Network (WLAN) (often branded "WiFi" by the WiFi Alliance) and fixed access, ETSI ISG has renamed Mobile Edge Computing to Multi-access Edge Computing in September 2016 [32, 33].

Compared to the centralized MCC, the MEC paradigm with distributed computing and caching resources being placed in close physical proximity to the mobile end devices, e.g., by placing compute and caching servers at BSs, brings several advantages for future low-latency networking, such as the Tactile Internet and IoT applications with millisecond-scale latency requirements. Besides reducing communication delay as the main goal, the MEC paradigm reduces the back-haul data traffic (compared to sending all UE service requests to the core network) [34, 35], extends the UE battery life times by offloading compute intensive tasks to edge servers [36], and provides real-time information of UE locations and behaviors, which are helpful for enabling context-aware services [16, 17]. Also, the MEC can support the wireless power transfer to mobile end devices [37–39].

2.1.2 Key technologies for implementation of MEC concept

To implement the MEC paradigm and make it operational, multiple integrative technologies are involved [40], mainly SDN, NFV, network slicing and Information Centric Network (ICN), as outlined next.

2.1.2.1 Software Defined Networking (SDN)

The main idea for introducing SDN was to enable the use of commodity and off-the-shelf hardware to create intelligent networks that are programmability and application aware [41, 42]. This is achieved by separating the control plane, which manages the network, from the data plane, which transfers actual data streams. Key to assuring interoperability between various equipment manufacturers and vendors is a well-defined open interface between the two planes. Logically centralized SDN con-

trollers help to solve classical networking problems, such as routing, tunneling, and IP address translation, as well as new challenges in future 5G applications, such as UE mobility, adaptation to service degradation, as well as security and integrated protection for IoT systems [43, 44]. Through SDN, network traffic flows can be flexibly steered to and from the MEC [45, 46] so as to seamlessly integrate MEC computations and caching into the provisioning of network services to mobile applications.

2.1.2.2 Network Function Virtualization (NFV)

NFV leverages virtualization techniques to enable the flexible design, deployment, and management of network functions, independent of the underlying physical network equipment [47–49]. These network functions may include classical functions, such as firewalls, deep packet inspection, the elements of the Evolved Packet Core (EPC) which is a framework to provide converged voice and data on LTE networks, but also innovative functions, including network coding, data aggregation, or computation as a service. An intuitive extension of the NFV concept combines single virtual network functions in a sequence to modularize complex functionalities in so-called Service Function Chains (SFC) [50–53].

2.1.2.3 Network Slicing

Network slicing is a virtual network architecture with the concept of running multiple logical network instances on top of a same physical infrastructure in order to provide better resource isolation and satisfy particular applications' demands. Network slicing is an efficient solution to address the future 5G heterogeneous services and requirements that coexist on the same underlying devices. The integration of Network slicing and MEC can be useful to fulfill some of the diverse requirements such as the low latency, high reliability and differentiation in traffic priorities that all are necessary for use cases like autonomous driving, massive IoT services or industry applications [44, 54, 55]. Figure 2.4 shows an example of network slicing concept.

2.1.2.4 Information Centric Networking (ICN)

The Internet, which was originally designed for host-to-host communications, is mainly used today for content distribution. The ICN paradigm aims to narrow the gap between the Internet's original design and the current applications, such as high-definition

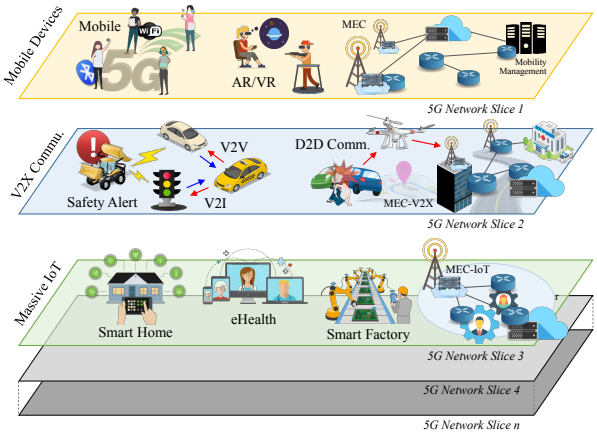


Figure 2.4: Integration of Network slicing and MEC.

video on-demand streams, 3D gaming, as well as augmented and virtual reality, with ever increasing traffic volumes. In order to optimize caching and content distribution, ICN proposes to redesign the Internet architecture as a content-centric network which adopts two design concepts, namely networking named contents (rather than hosts) and in-network caching, e.g., at MEC servers, to relieve the pressure on bandwidth as well as improving data delivery [56–59].

2.1.3 Device-to-Device (D2D) communication

The exponential growth of mobile data traffic and context-aware applications require innovative approaches to utilize the bandwidth more efficiently and to increase coverage, while lowering delay and energy consumption. The star-topology of cellular networks with a centralized control point, e.g., a BS or AP, suffers inefficiencies as all communication has to be relayed by the centralized control point. In contrast, D2D communication is a radio technology that enables direct data exchanges between two adjacent UEs without the involvement of the central control point or core network of the cellular network, i.e., without traversing the BS or AP [24–26, 60]. This direct D2D communication brings several benefits, such as improved spectral efficiency, increased data rates between devices, reduced power consumption, and reduced end-to-end delay. D2D communication has been employed in several studies for computation offloading to other near-by UEs (while not utilizing any MEC resources), e.g., [61–67]. Also, accessing caches at nearby UEs (while not utilizing MEC

caches) has been considered in prior studies, e.g., [68–75], whereby specifically video file caching at other UEs has been considered in [76, 77].

However, D2D communication also poses some implementation challenges. One challenge is the need to collect precise channel information, e.g., for estimating the channel and controlling the communication, which adds overhead. Security is another important challenge in D2D communication. Since a UE's data passes through other UEs, D2D communication is inherently susceptible to security attacks. Selfish exploitative UE behavior is another obstacle for collaborative multi-device D2D communication, as some UEs may use the communication resources of other UEs, e.g., for multi-hop D2D communication via intermediate relay UEs, without contributing their own resources to aid others. Interference and mobility management are also key challenges. Therefore, these D2D communication challenges need to be carefully considered when designing device-enhanced MEC systems that involve D2D communication.

Despite these challenges, D2D communication holds significant promise for a wide range of practical use-case scenarios in future communication systems. We proceed to briefly outline a few example use-case scenarios.

- National security and public safety: The reliance of cellular wireless communication on the availability of the cellular network infrastructure gives rise to severe problems in emergency and disaster scenarios, such as earthquakes and floods. Such disasters often damage the cellular network infrastructure, disrupting cellular wireless communication. In contrast, D2D communication does not require a fixed installed infrastructure and thus can continue to operate when the cellular network infrastructure is damaged. This advantage has made direct D2D communication a key component in projects proposed for future national security and public safety networks by the U.S. National Public Safety Telecommunications Council as well as European Conference of Postal and Telecommunications Administrations [78].

- Proximity and local-based services: The growing interest in multiplayer gaming, advertising, and social network services (e.g., Facebook and Instagram) has increased the need for efficient short-range communications to support interactions between near-by people with low latency and battery consumption while supporting high levels of user privacy [79]. D2D communication can facilitate such connections between different machines in close proximity, such as

a mobile phone connecting to a PC or other mobile phones to store and share video files and images [80].

- Vehicle-to-Vehicle (V2V) communication: Vehicular or Vehicle to Anything (V2X) communications is another important use case of D2D communication which is divided into three categories such as V2V, Vehicle to Infrastructure (V2I), and Vehicle to Network (V2N) communication [81–84]. Recent significant enhancements in computing and communication platforms as well as sensing capabilities of vehicles have shifted attention towards V2X communication to improve public safety and intelligent transportation system [85], collision avoidance systems [86], as well as the charging of electric vehicles [87].

We note that these outlined use-cases and a wide range of other D2D communications use-cases have the potential to significant benefit from jointly exploiting installed MEC computing and caching resources as well as the resources of near-by other mobile end devices, i.e., from device-enhanced MEC. In order to facilitate the further advancement of exploiting device-enhanced MEC through D2D communication, we comprehensively survey in the following two sections the existing research literature on device-enhanced MEC.

2.2 Enhancing MEC computation offloading with end-devices

With device-enhanced MEC, end devices, such as UEs can offload tasks that require heavy computations to powerful MEC servers or to nearby UEs in order to fulfill the low-latency demands of applications and extend their battery life time [88]. The offloading to nearby UEs is conducted over D2D communications, which reduces the load on the cellular network infrastructure and frees up some cellular bandwidth for other usages.

Given the widespread consideration of UEs as end devices in the existing device-enhanced MEC studies, we consider the terms “end device” and “UE” as interchangeable in this thesis. There are two categories for offloading, depending on whether the tasks can be partitioned or not, namely binary offloading and partial offloading. Binary offloading is employed for tasks that cannot be partitioned. Binary offloading either executes the entire task locally or offloads the entire task to an MEC server or

another nearby UE, as illustrated for the offloading from UE1 via UE2 to UE3 in the bottom part of Figure 2.5. Partial offloading is employed for tasks that can be partitioned into independent parts (sub-tasks) and executed in parallel, either locally or at MEC servers or other nearby UEs, as illustrated in the top part of Figure 2.5, where UE4 offloads its sub-tasks to an MEC server and UE5. The offloading to other UEs exploits the idle resources of nearby UEs via D2D communication, which can significantly improve the service to UEs [89, 90]. End devices can generally play three distinct roles in device-enhanced MEC:

- Helper node: A helper node computes offloaded tasks on behalf of UEs that require computation services.
- Relay node: A relay node helps other UEs through communication in order to offload their computation tasks to nearby devices or an MEC server for remote execution.
- Helper and relay node: A device can act as both helper and relay in order to execute and communicate offloaded tasks.

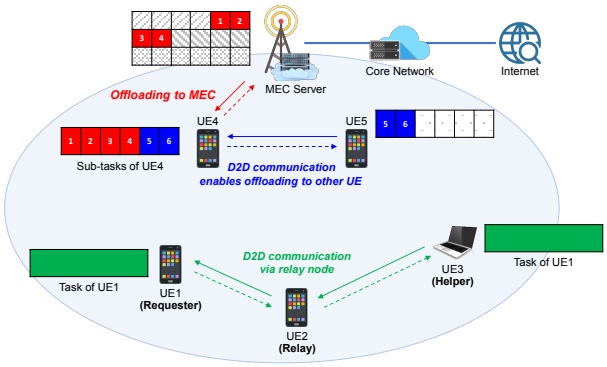


Figure 2.5: Illustration of device-enhanced MEC computation offloading [2].

2.3 Enhancing MEC content caching with end-devices

Mobile video streaming and related social networking already account for a large traffic proportion in wireless networks. The forecast continuous growth of this data-intensive traffic will likely overwhelm installed MEC caching resources or incur sub-

stantial additional investments by wireless operators (or lead to service degradations). Device-enhanced MEC caching exploits the extensive storage capacities in modern wireless end devices to supplement the MEC cache infrastructure. UE requests for data-intensive video streams, web pages, and related social networking applications can be *collaboratively* served by MEC cache servers, the local UE cache, and the caches of other nearby UEs (see Figure 2.6), which are reached via D2D communication [91, 92]. The caching contributions from the UE caches reduce duplicate content transmissions by the BS, which would result when popular content items are requested by the UEs in the range of a BS at different times. In particular, for social networking applications, exploiting the social relationships among UEs and their common interests using local D2D communication can be a key enabler for pre-caching popular content items in the caches of UEs with rich social ties [93].

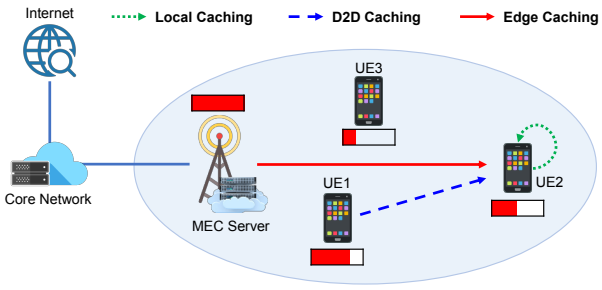


Figure 2.6: Illustration of device-enhanced MEC content caching [2].

Generally, there are two main aspects of caching, namely content placement and content delivery. Caching placement studies strive to design methods for optimally storing (placing) the content item files in caches at BSs and UEs. In contrast, content delivery studies focus on the transmission of the requested files to the end device.

Device-enhanced MEC caching strategies can generally be controlled in a distributed manner or in a centralized manner. The centralized control is typically implemented at the BS. Most existing studies have considered the centralized control since the BS typically tracks the required information, e.g., about UE locations, preferences, and requests, as well as content popularities and channel states. Thus, the BS has the required information to provide an optimal solution for the entire network encompassing the UEs within the range of the BS [94].

The device-enhanced MEC computation offloading concept is continued from now as the main use-case scenario of this work.

2.4 State-of-the-art research works on device-enhanced MEC computation offloading

The main objectives of the existing device-enhanced MEC computation offloading studies have been the minimization of the latency and the energy consumption through optimization of communication and computation resources. We organize this section according to the main objective of the existing device-enhanced MEC computation offloading studies, as summarized in Table 2.1. As Table 2.1 indicates, several studies have considered the joint minimization of latency and consumed energy, while some studies have focused on enhancing the security aspects of device-enhanced MEC computation offloading. The D2D access technology column in Table 2.1 gives the type of frequency resources considered for the D2D communication links in the studies, as well as the channel access method if a specific channel access method is considered in a study. The dash sign '–' indicates that no specific D2D access technology is considered in the study.

2.4.1 Latency minimization

MEC system failures diminish the quality of the service provided to the UEs. MEC server downtimes can incur enormous costs for businesses that rely on MEC server computations. The study [95] proposed two recovery schemes for an MEC server that is overloaded from serving too many computation tasks or for an MEC server that failed. The first scheme offloads the tasks of the overloaded or failed MEC server to available MEC servers within a transfer range. However for situations when there is no available neighboring MEC server, the second proposed scheme uses the UEs that are adjacent to an MEC server as ad-hoc relay nodes in order to provide a connection between the failed MEC server and a new MEC server. The study [95] assumes that an ad-hoc relay node can relay up to three LTE Frequency Division Duplex (FDD) Resource Blocks (RB) at a time. It is shown that the proposed method works well in dense areas. However, the study [95] has only considered the data downlink from the recovery MEC server, while the data uplink to the recovery MEC server has been neglected. The availability of neighboring resources is also not guaranteed by the protection strategies.

Importantly, the study [95] has only considered the UEs as relay nodes towards

the new MEC server and ignored the usage of their computation resources. However, it is beneficial in terms of delay to use the available resources in the vicinity. Considering this fact, a joint task assignment and resource allocation for device-enhanced MEC computing has been proposed in [96]. In this study, a UE can offload its computational-heavy tasks to several nearby end devices, such as smart wearable devices, cell phones, tablets, laptops, as well as infrastructure nodes, such as WiFi APs and cellular BSs, as helper nodes. The task assignment is optimized to minimize the latency, subject to UE and helper energy constraints. Each UE can compute a task locally, or offload the task to a helper node for remote execution. The tasks are considered non-partitionable, however parallel execution of independent UE tasks is possible. A time-slotted communication protocol with three phases is developed. Within the three phases of a time slot, the task is offloaded to one of the helper nodes and the computation results are sent back to the UE. The resulting mixed-integer non-linear minimization problem is solved by relaxing the integer task assignment variables, which results in an efficient, albeit suboptimal solution.

The follow-up study [97] reduced the overall latency by considering controllable computation frequencies instead of fixed processing capacities. Nevertheless, there are still some limitations. The UEs and channel condition are considered static; however, in reality UEs are mobile and channels are dynamic. Therefore, UE mobility and dynamic channels should be addressed through adaptive mechanisms in future research. In addition, only Time-Division Multiple Access (TDMA) is used due to its ease of implementation; other orthogonal multiple access methods for D2D communications should be examined in future research to improve the system performance.

The design of an incentive mechanism to motivate UEs to share their computation resources is a key factor in device-enhanced MEC computation offloading and has been neglected in the studies surveyed so far. The study [98] presented bandwidth incentives for UEs. The considered system contains one BS and numerous UEs. The UEs are either Computing User Equipment (CUE), which have computationally intensive tasks, or Helper User Equipment (HUE), which help by taking over some of the computation sub-tasks. CUEs motivate HUEs to take over some computation sub-tasks as follows. CUEs give some of their available communication bandwidth to the HUEs in exchange for the help with computations. Thus, HUEs essentially trade in some of their computation resources in order to increase their overall communication bandwidths. A CUE can either offload a task to an MEC server using its full available bandwidth or offload a part of the task to the MEC server and the rest to

an HUE; thereby lending some of its bandwidth to that HUE.

An optimization problem has been formulated to model the decisions for pairing a CUE to a suitable HUE, the task offloading, and the partitioning of the MEC server resources among UEs. The study [98] assumed that each HUE can only assist a single CUE. Also, the specifics of the bandwidth lending process were neglected and UE mobility was not considered.

2.4.2 Energy consumption minimization

In order to improve the MEC performance, a joint computation and communication cooperation method has been presented in [89]. The study [89] considers a basic three-node MEC system with two UEs, whereby one UE needs computation resources and the other UE is the helper/relay. Moreover, one AP node is attached to an MEC server. A four-slot protocol is proposed to enable energy-efficient device-enhanced MEC that minimizes the total energy consumption at both UEs, but also considers the UE's latency-constrained computation requirements. UE computation tasks are assumed to be partitionable; thus, a computations task can be partitioned and the different partitions can be executed locally, offloaded to a helper, or offloaded to the MEC server. However, the examined approach does not fully exploit the capacity of the multiple access channel from the multiple UEs to the MEC server. This limits the performance of multi-user MEC systems [99]. Another drawback of this study is the simple evaluation topology, which included only two UEs.

A cellular D2D framework with a massive crowd of devices at the network edge for joint computation and communication resource sharing has been proposed in [100]. The UE energy consumption is minimized by optimizing the task assignment with a graph matching policy, which can achieve good D2D task assignments. However, the energy-efficiency of the D2D clusters is not considered in the study [100], since it mainly deals with the D2D crowd task assignment problem [101]. In addition, in order to make the proposed framework practical, scenarios with changing D2D connections need to be considered in future research. Moreover, to prevent UEs from over-exploiting other UEs and from free-riding behaviors, an incentive mechanism should be added in future research.

The minimization of the energy consumption of computation task offloading in device-enhanced MEC with a large number of UEs poses significant modeling and computational challenges. The two studies [102] and [103] have investigated game-

theoretic models for device-enhanced MEC offloading with large UE numbers. More specifically, the study [102] has formulated the offloading decision problem as a sequential game and examined the stable Nash equilibrium for the system. The study [103] has formulated the problem as a non-cooperative strategic potential game [104]. Both studies found that the game-theory based device-enhanced MEC computation offloading reduces the consumed energy compared to computation on only the MEC servers or computation on only the local UEs.

To take the long-term UE incentive constraints into account and avoid free riding behaviors of UEs which may deter other UEs from collaborating, a D2D framework is presented in [105] to minimize the time-average energy consumption with a Lyapunov optimization based online task offloading. UEs can dynamically share their resources, whereby the sharing is controlled by the BS. The BS establishes in-band LTE-direct Orthogonal Frequency-Division Multiple Access (OFDMA) D2D links between UEs (out-of-band links, e.g., Bluetooth, cannot be controlled by the BS). The working day movement model has been used to characterize the UE mobility patterns. This model which is based on people's daily life activities, including commuting from home to work, spending time at the work place, and commuting back from work to home. The working day movement model has shown close similarity to real-world mobility measurements [106]. Three types of tasks have been considered, namely pure computations tasks, such as image processing, pure communications tasks, such as file downloading, and hybrid tasks requiring both computation and communication resources, such as video streaming. The evaluation model generates the UE application layer tasks according to a Poisson process to represent the stochastic nature of real-life task generation. Tasks are admitted based on a best-effort first-come-first-serve admission policy.

The task admission policy is independent from the scheduling of the task offloading and only operates at the start of a time frame. Interactions between task admission and task offloading should be examined in future research.

The rapid growth of the IoT and fog computing have brought computing devices, which are referred to as fog computing devices, with idle resources close to the UEs. Accessing both the MEC and the fog computing devices can improve energy savings. Towards this goal, an energy-efficient joint computation offloading via cellular networks to the MEC server and via D2D communications to fog computing devices in a 5G network has been presented in [107]. Some UEs are deployed around one MEC server in the considered system. The access technology between UEs and the

MEC server is an LTE radio access network. Fog computing devices with idle computing resources near the UEs functions as helpers. In particular, each UE has a fixed fog computing helper device and communicates with its helper through D2D links. Since the helpers have also limited computing resources, three computation task execution models are considered, depending on the UE demands for computation resources: local, fog computing device, and MEC server execution. The computation offloading framework has two parts, namely a control plane and a data plane. The control plane includes the controller, which is responsible for offloading decisions according to the network status. The data plane includes the task queue buffer and the task data transmission parts in the UEs. Simulation results have demonstrated that the proposed method is effective; however, several issues, such as communication overhead, synchronization, data recovery overhead, security, and incentive mechanisms, are neglected in the framework.

Advanced energy harvesting techniques to power mobile devices with renewable energy, such as solar and wind energy, can extend the battery life time of devices. A new device-enhanced MEC computing and networking framework called D2D Edge Computing and Networking has been proposed in [108] toward designing a green computation MEC system that exploits advanced energy harvesting techniques. The examined D2D Edge Computing and Networking system includes a BS and some UEs, whereby one UE is called the master and the rest are secondary devices. The master device is the UE with a computation-intensive task. The master device is equipped with energy harvesting elements. The offloading process is divided into successive time slots of the same length. The task assignment decision, CPU frequency adjustment, and power control are accomplished at the beginning of each time slot. The task transmission and computation at the master and secondary UEs fill the total task execution time in each time slot. The communication setting between UEs is based on the LTE-D2D standard with the Frequency-Division Multiple Access (FDMA) protocol for dedicated D2D transmissions. The energy cost model for each time slot includes the energy consumed for task transmission and processing at the master and secondary devices. A system operation cost is defined to give a reward or penalty to the D2D-Edge Computing and Networking system. The reward or penalty is a function of the energy consumption and cost for a unit of energy. The joint optimization of the computation offloading and the resource management to reach a good trade-off between low system operation cost and short task execution time is formulated as a constrained Markov Decision Process (MDP). In order to

execute this joint optimization problem, a Q-learning algorithm is employed, which helps to address the stochastic features of harvesting energy and network information. In addition, a low-complexity online Lyapunov optimization based algorithm is developed to tackle the challenges of high dimensionality of the D2D Edge Computing and Networking offloading framework. However, in this study, the system status is considered static in each time interval, which may not be a realistic assumption for scenarios with high UE mobility. The simple system model with only one BS and one master UE device is another drawback of this study.

Based on recent advances in antenna design, the study [109] has proposed an energy efficient offloading scheme using Full Duplex (FD) relays. The network consists of one BS and several UEs forming multiple clusters. One UE with FD antennas is selected as the cluster head, referred to as FD-DCH, in each cluster. This FD-DCH acts as a relay between normal UEs in the cluster (DUEs) and the BS. When DUEs send a proportion of their tasks to their associated FD-DCHs, the tasks will be relayed simultaneously to the BS on the same frequency band used for D2D communication. To avoid interference, it is assumed that DUEs and FD-DCHs work on orthogonal spectrums in both up-link and down-link. The cluster head selection algorithm is based on the Chinese Restaurant Process [110] and the weighted sum method considers several metrics, such as UEs' social behaviors, energy and storage resources, and the transfer rate from the BS to the UEs. The mobility of UEs, which can change the social attributes and consequently the cluster head selection procedure is neglected in this study.

2.4.3 Joint minimization of latency and energy consumption

A simple scenario to minimize the task execution cost which can jointly consider latency and energy consumption minimization for a system with one BS has been proposed in [111]. The problem is transformed into a computation offloading subproblem and a resource allocation subproblem which are solved by the Kuhn-Munkres algorithm [112] and the Lagrangian dual method, respectively. In [111], UE tasks are considered partitionable and parallel execution at the requesting UE and at an MEC server or helper UE is possible.

The total task execution cost problem is further investigated in [113] with the consideration of users movements using a hybrid offloading framework called HyFog. The cost problem has been defined as the weighted sum of the UE computational

time and the UE energy consumption. HyFog chooses between UE task offloading to the MEC or to nearby end devices using D2D communication (cellular D2D or WiFi-direct). The working day movement model has been used as UE mobility pattern. A novel three-layer graph matching algorithm has been developed to represent the choice space consisting of local (UE) task execution, D2D task offloading to nearby UEs, and task offloading to the MEC. The total task execution cost is minimized through problem mapping to a minimum weight matching problem in the three-layer graph and the Edmonds' Blossom algorithm [114]. The study [113] has only focused on spectrum allocation problems. However, the development of mechanisms that overcome the instinctive selfishness of the UEs remains a key challenge. Instinctively, each IoT user typically optimizes its own Quality of Experience (QoE) individually without following the strategies for optimizing the overall system performance [115].

Some IoT applications require ultra-low latency computation services. However, poor channel conditions between end devices and the MEC server may impede latency-constrained IoT applications. To address this problem, the study [116] proposed a forwarding scheme to improve resource sharing for mission-critical IoT devices which fall under the coverage of neighboring end devices. A greedy example heuristic has been proposed to solve the optimization problem for task allocations [116]. In particular, the tasks are allocated according to two main criteria: the proximity of the devices and the number of tasks that have already been allocated to a given device. The evaluations in [116] demonstrated through simulations that by using D2D communication in this way, lower latency, energy consumption, and traffic load through the network can be achieved and improvements in the cooperation of IoT devices at the edge of the network are possible. LTE-Direct with OFDMA and Single-Carrier FDMA have been employed for the down-link and up-link D2D communications, respectively. A round robin scheduler divided the RBs equally between the candidate D2D transmissions (with 6 RBs for D2D). This RB division avoided interference. The random direction movement model, which is a variant of the widely used random waypoint model [117], is considered as the mobility model. An interference coordination scheme that reuses parts of the available frequencies could achieve additional performance gains.

The study [118] has proposed an offloading method with frequency reuse for IoT applications. In the studied architecture, UEs send their computation requests to the MEC server. The MEC server determines the offloading destination according to a

two-step algorithm. The first step processes delay-sensitive tasks, while the second step processes tasks of UEs with energy restrictions. The offloading problem for delay-sensitive tasks is modeled as a delay-aware adjacency graph, which is solved for a maximum matching with minimum cost with Edmonds' Blossom method [114]. The result specifies whether the computation requests are offloaded through D2D communication to near-by UEs or to the MEC server. The MEC server then conducts an analogous graph-based solution procedure for the remaining requests from UEs with energy limitations and allocates the computation resources of the remaining idle near-by UEs and its own resources. If the MEC server becomes overloaded, it offloads the computation tasks of energy-limited UEs to the central cloud.

Common drawbacks of the preceding studies on the joint minimization of latency and energy is their use of conventional cellular and WiFi technologies for D2D communication only, as well as their simulation based evaluation. It is important to examine novel D2D communication technologies as well as to examine the effectiveness of an offloading algorithm through real implementations. The study [119] addressed these drawbacks by proposing the first task offloading framework with Near Field Communication (NFC) based D2D communication and a real implementation evaluation. NFC has several advantages over the longer-range Bluetooth and WiFi technologies due to its short communication range, including lower interference, lower energy consumption, and intrinsic security. The proposed framework circumvents some of the limitations of default Android NFC protocols: the NFC-based task offloading enables bidirectional communications between two UEs and makes the task offloading smoother. The performance evaluation in [119] demonstrated that the NFC interface reduces the UE energy consumption and reduces the execution time of the offloaded task, especially for powerful helper devices. Nevertheless, the NFC-based task offloading in [119] has several limitations. First, the data transfer rate of NFC based communications is only 53 kB/s, because the used hardware can transfer only one message per connection; therefore, the framework is not suitable for data-demanding application scenarios. Moreover, the device heterogeneity and the potential of parallel connections using Bluetooth and/or WiFi-direct as well as user mobility should be examined in future research.

Although the study [119] is based on a practical implementation, the study [119] as well as all prior studies on joint latency and energy minimization lack an incentive mechanism. An incentive mechanism is generally required to make the offloading attractive for users in real D2D systems. A generalized offloading scheme with an in-

centive approach based on credit and reputation to increase the cooperation among UEs via D2D communication has been proposed in [120]. The proposed task offloading system enhances the accessibility of UEs to offloading support and improves their QoS. The social-characteristics of the UEs [121, 122] are exploited to form offloading communities. An offloading community is formed by a group of UEs that trust each other with offloading tasks. A UE gains points when it shares computational resources with other UEs, stays in a certain location for a longer time, or pre-caches some tasks. A UE loses points when utilizing the community resource pool. In [120], the community assignment is based on the frequencies and durations with which the UEs are detected. This assignment approach requires the activation of the UE discovery interfaces. A learning method for predicting communities can improve the discovery process and save energy [123]. Also, new task process acceleration techniques that exploit multiple devices are an important direction for future research.

2.5 Summary and discussion

The main objectives of the existing device-enhanced MEC computation offloading studies have been the minimization of the latency and energy consumption of the UEs as well as the enhancement of security. Most existing studies have considered partitionable and non-partitionable computational tasks, depending on the application scenario. Individual sub-tasks of partitionable tasks or complete non-partitionable tasks can be executed locally (if the UE has sufficient computation resources and the latency of local UE execution can be tolerated), offloaded to adjacent UEs directly via D2D communication or via relays, or offloaded to an MEC server.

The wireless channel characteristics and the UE resource availabilities are generally stochastic and change with time due to the UE mobility. Therefore, offloading decisions should be based on the latest status of the system and be computed online. Overall, Lyapunov optimization based algorithms have so far been the predominant optimization tools for tackling the challenges of the high dimensionality of the offloading frameworks. Lyapunov optimization based algorithms can solve the offloading optimization problems with low-complexity online computations based on the current state of the system, as well as the drift-plus-penalty function for stabilizing the queues.

The task assignment, i.e., the decision on where to execute a computation task

or sub-task can generally either be made in a distributed manner or a centralized manner (at the BSs). The examined task assignments to other UEs or an MEC have typically been based on various aspects of the UE and MEC server resources as well as the UE computation demands. Despite the considerable amount of research devoted to task assignment, the proposed approaches are generally oversimplified. In particular, they did typically not consider the dynamics of wireless communication links. Also, the heterogeneous computational capabilities and time-varying availabilities of the computational resources of the end devices and MEC servers have typically only been partially considered. Future research needs to develop practical approaches that optimize the computation offloading (task assignment) while comprehensively considering the wireless network dynamics and heterogeneity and dynamic availabilities of the end devices and MEC servers.

While the centralized control approach is appropriate for small network sizes, a purely centralized approach may become infeasible or inefficient for large-scale networks. This is because the adaptation to the network dynamics requires frequent data collection from the entire network domain and subsequent centralized processing. This centralized processing translates into long signalling delays, large control signaling overhead, and high computational complexity in large-scale networks [124]. The existing research studies that considered distributed task assignments, neglected the network dynamics; thus they cannot be readily applied to device-enhanced MEC systems [96]. Future research needs to explore hybrid decision approaches that delegate some scope of the decision making to local nodes, while slow-timescale global decisions can still be made at a central controller. While such hybrid approaches have begun to be explored for general wireless resource allocation problems [125–130], they remain an open research area for device-enhanced MEC computation task offloading.

In order to reach the main goal of efficient device-enhanced MEC computation offloading for real world applications and scenarios, future research needs to further examine the interactions between task admission policies and the scheduling of task offloading as well as effective ways to continuously maintain the offloading service when UEs are mobile. In addition, relying only on orthogonal multiple access technologies, such as TDMA and FDMA, may limit the performance of multiuser MEC systems [99]; hence, there should be more focus on using new channel access technologies that exploit the particular network architecture. Generally, WiFi appears to be the most practical medium access technology for D2D communication between

UEs. Nevertheless, emerging physical layer technologies should be evaluated for providing D2D UE communications. Despite a wide range of studies on the design of incentive mechanisms, there is still a pronounced lack of systematic research on participation incentives that consider the interdependent security risks.

The evaluation methodology in most of the existing computation offloading studies is simulation and only few studies have considered practical scenarios. Future research needs to broaden the evaluation to consider mathematical analysis when appropriate to obtain relevant insights through tractable analysis. Also, prototypes of the proposed device-enhanced MEC computation offloading systems should be developed and evaluated through measurements for representative work loads and mobility patterns.

2.6 Conclusion

Device-enhanced MEC augments the MEC computing and storage (caching) resources with the computing and storage resources of the wireless end devices, e.g., UE nodes. Device-enhanced MEC thus enlarges the resource pool that is available for providing services to end devices. This enlargement of the available resource pool is achieved without additional investments in MEC infrastructure; albeit, device-enhanced MEC typically requires some incentives (e.g., payments) to the owners of the participating end devices. Nevertheless, with the ever-increasing computing and storage resources available in mobile end-devices, device-enhanced MEC is an attractive paradigm for improving the service quality without requiring large upfront capital investments in more MEC resources. Also, device-enhanced MEC works particularly well in dense networks, where each end-device has a large number of neighboring end devices within a short D2D communication distance, e.g., in crowded places such as stadiums. Such dense network scenarios pose scalability problems for conventional MEC with a fixed amount of installed resources. Generally, the possibilities for "recruiting" neighboring end devices to contribute computation and storage resources grow in dense networks, as there are more end devices near any given end device in dense networks. Thus, device-enhanced MEC holds a particular promise to mitigate MEC resource shortages in dense networking scenarios.

This chapter was organized to focus on the basic concepts of computation offloading in the device-enhanced MEC environment, D2D communication and the pros

and cons of previous research studies. The studies are sub-categorized according to their main objective. The existing studies have strived to reduce the latency, and to reduce the energy consumption. Also, some studies have focused on enhancing security aspects, while others have focused on maximizing some utility measure. Overall, the device-enhanced MEC studies that have been conducted to date have made significant progress in advancing the protocol development and optimization for offloading computations to MEC resources and other end devices.

Nevertheless, device-enhanced MEC is a nascent research area; most studies have appeared within the past three years. Thus, the existing state-of-the-art research in the device-enhanced MEC area has severe limitations and requires extensive future research to address the numerous open challenges. Overall, only roughly half of the existing studies have accounted for end device mobility. Also, less than roughly a quarter of the existing studies has incorporated an incentive mechanism. Moreover, there is an overarching need to develop effective and efficient control and management frameworks for device-enhanced MEC that can cope with end device mobility and end device heterogeneity while scaling to large network sizes and device densities. Future research should also further improve device-enhanced MEC, e.g., by exploiting emerging Machine Learning (ML) techniques and improved models of the social relationships of end device users. Also, comprehensive performance evaluation frameworks and methodologies should be developed and agreed upon by researches to facilitate the comparison of different approaches to device-enhanced MEC.

Table 2.1: Summary of device-enhanced MEC computation offloading studies [2].

Objective	Study	Year	D2D Access Technology	Mobility of UEs	Incentive Mechanism	Evaluation Environment
Latency minimization	[95]	2017	Cellular resource	No	No	Simulation using Python and Matlab
	[96]	2018	Cellular resource, TDMA	No	No	Simulation
	[97]	2019	Cellular resource, TDMA	No	No	Simulation
	[98]	2019	–	No	Bandwidth incentive	Simulation
Energy consumption minimization	[89]	2018	–	No	No	Simulation
	[100]	2017	Cellular resource	Yes	No	Simulation
	[102]	2018	Cellular resource	No	No	Simulation
	[103]	2019	Cellular resource	No	No	Simulation
	[105]	2016	Cellular resource, OFDMA	Working day movement model	Tit-for-tat	Simulation
	[107]	2019	Cellular resource	No	No	Simulation
	[108]	2019	Cellular resource, FDMA	No	No	Simulation
	[109]	2019	Cellular resource	No	No	Simulation
Joint minimization of latency + energy consumption	[111]	2018	Cellular resource	No	No	Simulation
	[113]	2017	Cellular resource	Working day movement model	No	Simulation
	[116]	2017	Cellular resource, OFDMA downlink, SCFDMA uplink	Random direction movement model	No	Simulation using Matlab
	[118]	2019	Cellular resource	No	No	Simulation
	[119]	2017	Near Field Communication	No	No	Android
	[120]	2017	–	Yes	Credit and reputation	Real and lab tests with Android OS

3 Energy-aware Cooperative Computation Offloading

The content of this chapter was previously published in:
Mehrabi, Mahshid, et al. "Energy-aware cooperative offloading framework for inter-dependent and delay-sensitive tasks." *IEEE Global Communications Conference, 2020* [3].

3.1 Introduction

As mentioned in the previous chapters, computation offloading is a key enabler of providing computation intensive tasks on mobile user devices in 5G networks. Furthermore, by using D2D links between adjacent devices, computation and communication cooperation can be achieved which ease the task offloading and broaden the concept of MEC networks. In this chapter, we will investigate joint computation and communication cooperation task offloading in a basic three node device-enhanced MEC system. The aim is to reduce the battery consumption of the user while satisfying the inter-task dependencies and latency deadline requirements. Simulation results show the superior performance of the proposed algorithm compared to the other state-of-the-art methods.
In order to solve such a problem, we first transform our MINLP algorithm to a Quadratically Constrained Quadratic Programming (QCQP) approach and then using a Semidef-

inite Relaxation Method (SDR), we obtain an approximation of the original problem. Finally, a randomization method is applied to obtain optimum offloading strategy.
This chapter is organized as follows. First, the system model and network condition are introduced in Section 3.2. Then, the optimization problem is formulated in Section 3.3, followed by the randomization method for our offloading decision algorithm. In Section 3.4, numerical results are presented and finally, conclusions and future works are stated in Section 3.5.

3.2 System model

There are two user-equipments called UE and UER in the cluster, which can directly communicate with each other via a D2D network. The UE runs an application while the UER can act as a helper or relay, which can help UE to execute tasks or receiving the task data from the user and then forwarding it to the MEC server.
The MEC server has more powerful resources, such as higher CPU frequency and number of processing units than the two devices. We assume that the MEC server is powered by the energy grid, so the energy usage of the sever is not considered in our work. Both devices can also communicate with the edge server directly via cellular networks. The BS establishes in-band LTE-direct OFDMA for dedicated D2D transmissions. The MEC server is connected to the cloud server via wired networks (such as optical fiber network).

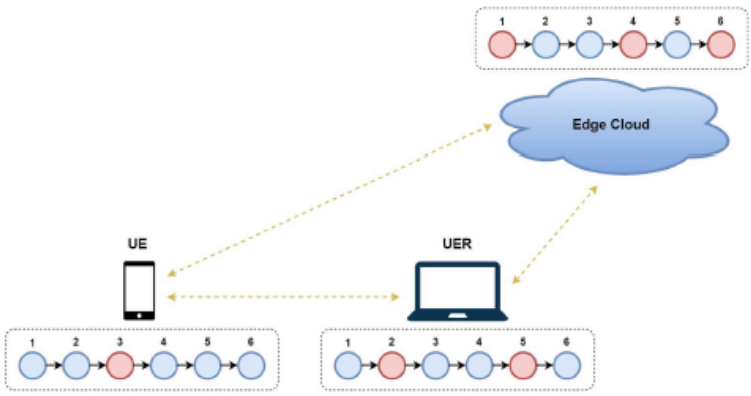


Figure 3.1: A basic three node device-enhanced MEC system [3].

The wireless channel is assumed quasi-static during the execution time and the

Channel State Information (CSI) and computation related parameters are available for devices. The computation result of each task is assumed much smaller than the input bits, therefore, the time for sending the result back is negligible and only the transmission time of the task input and the execution time are taken into consideration. There is a time deadline for completing the UE1's tasks and it is assumed that the data size and computation resource requirements for each task are known in advance. In the following, we elaborate the system model parameters.

3.2.1 Task model

In our scenario, the UE's tasks are sequentially dependent, meaning that each task requires the result of its previous task. The distributed Ibis application presented in [9] would be an example for a sequential task dependency scenario. The application contains the set of tasks presented by $\mathcal{K} = \{1, 2, ..., K\}$. Each task k has parameters of $A_k = \{b_k, c_k, d_k\}$, where b_k is the input data size of the computation task k (in bits), c_k is the computation resources which is required for execution of each bit in task k (in CPU cycles/bit), and d_k, equals to b_k times c_k, is the total amount of required computation capacity for execution of the task k. T_d is the time deadline for the execution of the whole application.
The dependency graph of tasks is shown in Figure 3.2. To apply task dependencies effect to the computation offloading decision algorithm, the starting and finishing time concepts are needed as follows [10]:

- Finish time: It is the time that the task k execution is completed:

$$FT_k = ST_k + T_k^{exe}, \tag{3.1}$$

  where ST_k is the time that the execution of task k predecessors is finished and T_k^{exe} presents the time for task k execution itself.

- Start time: It is the time that the execution of task k can be started:

$$ST_k = \begin{cases} 0, & k = 1 \\ FT_{k-1}, & \forall k \neq 1 \end{cases} \tag{3.2}$$

According to (3.2), since there is no predecessors for the first task, its start time is zero.

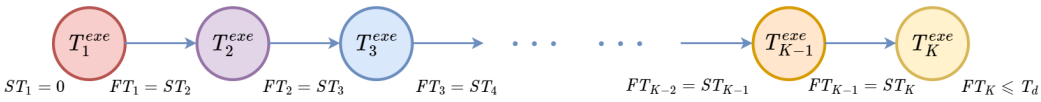


Figure 3.2: The sequential dependency graph [3].

3.2.2 Communication model

The up-link data rate for task k transmission can be achieved using the Shannon theorem and the down-link data rate is neglected in our scenario due to the small size of the executed tasks.

$$r_k = B \log_2 \left(1 + \frac{P_k^{tr} H_k}{\Gamma \sigma^2}\right), \tag{3.3}$$

Here, P_k^{tr} is the sender's transmission power, B and H are the channel bandwidth and channel gain between sender and receiver respectively, and σ^2 denotes the variance of the Gaussian channel noise. Γ is the coding gap as a function of bit error rate (BER) which is determined based on the coding schemes and medium access protocol [131]. To simplify the model, here we assume $\Gamma = 1$.

3.2.3 Computation model

There are three possible scenarios for tasks' execution in our offloading algorithm: local execution on the UE, remote execution on the helper, or remote execution on the MEC server. The latter can be either be a direct offloading from UE or via the relay to the MEC server.
Depending on that which device or server is executing the task, the time and energy needed for execution of task k is calculated based on the computation capability of the host. This can be calculated based on the CPU cycles needed for execution of task which is known in advance in our scenario.

3.2.3.1 Local computing

The time needed for task k execution locally on the UE is then obtained as follows:

$$T_k^L = \frac{d_k}{f^{UE}}, \tag{3.4}$$

where f^{UE} is the computation capacity required for task k execution and it is assumed to remain same during processing of task k. Based on the effective switched capacitance λ, the energy consumption per operation is $\varepsilon = \lambda(f^{UE})^2$, depending on the chip architecture [132]. Therefore:

$$E_k^L = \lambda \left(f^{UE}\right)^2 d_k. \tag{3.5}$$

The equations (3.4) and (3.5) are fully descriptive due to the no transmission posiblility of data in the local mode.

3.2.3.2 Computing on the helper

In the second scenario, the task is offloaded from UE to UER, which plays the role of helper for UE to execute its computation task. The computation execution time consequently consists of two parts, communication time and computation time. Therefore, the total time budget splits into two parts as follows:

$$T_k^H = \frac{b_k}{r_k^{UH}} + \frac{d_k}{f^{UER}}, \tag{3.6}$$

where r_k^{UH} is the transmission data rate between UE to UER and f^{UER} denotes the required computation capacity to execute task k on the helper. The energy consumption in this mode is then calculated as follows:

$$E_k^H = P_{tra}^{UE} \left(\frac{b_k}{r_k^{UH}}\right) + P_{wait}^{UE} \left(\frac{d_k}{f^{UER}}\right). \tag{3.7}$$

Here, P_{tra}^{UE} is the expended transmission power of UE, r_k^{UH} is the transmission data rate between UE to UER and P_{wait}^{UE} is UE's idle circuit power which is the energy that UE consumes while waiting to get the result back. As per the scenario, the return trip for the results is neglected in this, as well as the following strategies due to its smaller size both in terms of energy as well as time.

3.2.3.3 Computing on the edge cloud

Under this final offloading mode for this scenario, the task is offloaded to the edge server, with its more powerful computational abilities and energy supply. UE transmits the task to the MEC server directly and after remote execution, the result is sent back to UE. Therefore, there are two time delay steps for transmission and computation:

$$T_k^S = \frac{b_k}{r_k^{US}} + \frac{d_k}{f^S}, \tag{3.8}$$

where R_k^{US} is the transmission rate from UE to MEC server and f^S is the computation capacity allocated for task k execution on the server. The corresponding energy consumption for the UE in this remote case execution is calculated as follows:

$$E_k^S = P_{tra}^{UE}\left(\frac{b_k}{r_k^{US}}\right) + P_{wait}^{UE}\left(\frac{d_k}{f^S}\right). \tag{3.9}$$

3.2.3.4 Computing on the edge cloud via relay

In this mode, the task is still offloaded to the edge server, but the difference is that UE doesn't send the data to the server directly, but with the help of UER. This time UER plays the role of a relay, receives data from UE, and transmits the data further to the MEC server. This can be a good strategy when the wireless channel state between UE and MEC server is far worse than the channel state between UER and MEC server. Therefore, there are three time delay steps for transmissions and computation:

$$T_k^{HS} = \frac{b_k}{r_k^{UH}} + \frac{b_k}{r_k^{HS}} + \frac{d_k}{f^S} \tag{3.10}$$

Here, r_k^{HS} is the transmission rate from UER to MEC server.

The corresponding energy consumption for the UE in this remote case execution can be then calculated as follows:

$$E_k^{HS} = P_{tra}^{UE}\left(\frac{b_k}{r_k^{UH}}\right) + P_{wait}^{UE}\left(\frac{b_k}{r_k^{HS}} + \frac{d_k}{f^S}\right). \tag{3.11}$$

3.3 Computation offloading problem formulation under sequential task dependency

The goal of our joint communication and computation cooperation offloading algorithm is to minimize the energy consumption of the UE considering the execution deadline of the tasks.

3.3.1 Problem formulation

Considering the offloading scenarios defined above, the total energy consumption and execution time of task k on UE can be formulated as follows:

$$E_k^{exe} = w_k E_k^L + x_k E_k^H + y_k E_k^S + z_k E_k^{HS}, \tag{3.12}$$

$$T_k^{exe} = w_k T_k^L + x_k T_k^H + y_k T_k^S + z_k T_k^{HS}, \tag{3.13}$$

where w_k, x_k, y_k and z_k denote the binary decision variables for the offloading algorithm meaning that only one of them can be 1 and the rest are 0 for each task. The total energy consumption of the UE for the whole application can be then formulated by

$$E_{UE} = \sum_{i=1}^{K} E_k^{exe}. \tag{3.14}$$

Therefore, our user's energy minimization problem with respect to the time deadline for execution and tasks' inter-dependencies can be stated as follows:

$$\min_{\alpha,\beta} \quad \sum_{k=1}^{K} E_k^{exe} \tag{3.15a}$$

$$
\begin{aligned}
\text{subject to:} \quad & w_k, x_k, y_k, z_k \in \{0,1\}, \forall k \in \mathcal{K}, && \text{(3.15b)}\\
& w_k + x_k + y_k + z_k = 1, \forall k \in \mathcal{K}, && \text{(3.15c)}\\
& FT_K \leqslant T_d, && \text{(3.15d)}\\
& ST_1 = 0, && \text{(3.15e)}\\
& ST_k = FT_{k-1}, \forall k \in \mathcal{K}, k \neq 1, && \text{(3.15f)}
\end{aligned}
$$

where $\alpha = [w_1, x_1, y_1, z_1, \ldots, w_K, x_K, y_K, z_K]$ and $\beta = [ST_1, ST_2, \ldots, ST_K]$ are offloading binary decision variables and starting times, respectively. Constraint (3.15d) assures that the application is finished before the time deadline; constraints (3.15e) and (3.15f) define the start time conditions for task k.

3.3.2 Solution of the optimization problem

Due to the existence of a non-linear constraint, the problem is Non-Linear Programming (NLP) and since the first constraint is integer, the problem is then a mixed-integer. Thus, this optimization problem is MINLP, which is an NP-hard problem which means there is no known way to solve it in polynomial time.
In order to solve such a problem, we first transform it into a homogeneous QCQP and then using SDR, we obtain an approximation of the original problem. Finally, a randomization method is applied to obtain optimum offloading strategy.

In the first step, the integer constraints are inverted to the quadratic formats as following:

$$
\begin{aligned}
& w_k (w_k - 1) = 0, x_k (x_k - 1) = 0, y_k (y_k - 1) = 0, \\
& z_k (z_k - 1) = 0, \forall k \in \mathcal{K}.
\end{aligned}
\tag{3.16}
$$

Then, the QCQP transformation of our minimization problem (3.15a) is:

$$
\min_{\alpha,\beta} \quad \sum_{k=1}^{K} E_k^{exe} \tag{3.17a}
$$

$$
\begin{aligned}
& w_k\,(w_k-1)=0, x_k\,(x_k-1)=0, \\
\text{subject to:}\quad & y_k\,(y_k-1)=0, z_k\,(z_k-1)=0, \qquad (3.17b)\\
& \forall k \in K \\
& w_k + x_k + y_k + z_k = 1, \forall k \in K, \qquad (3.17c)\\
& FT_K \leqslant T_d, \qquad (3.17d)\\
& ST_1 = 0, \qquad (3.17e)\\
& ST_k - FT_{k-1} = 0, \forall k \neq 1, \qquad (3.17f)
\end{aligned}
$$

In the next step, we define two vectors called v with the size of $(5K+1)\times 1$ as $v = [\alpha, \beta, 1]^T$ and v' with the size of $5K \times 1$ as $v' = [\alpha, \beta]^T$ with standard unit vectors e_i with the ith entry equal to 1 and size of $(5K+1)\times 1$, and e'_i with the ith entry equal to 1 with the sizes of $5K \times 1$, respectively. Thus, the QCQP transformation of our minimization problem (3.17a) can be formulated as:

$$
\min_{v} \quad (a_0)^T v \qquad (3.18a)
$$

$$
\begin{aligned}
\text{subject to:}\quad & v^T \operatorname{diag}(e_i)\, v - (e_i)^T v = 0, \\
& i = 1, \ldots, 4K, \qquad (3.18b)\\
& \left(a_k^I\right)^T v = 1, \forall k \in K, \qquad (3.18c)\\
& (a_1)^T v \leqslant T_d, \qquad (3.18d)\\
& (e_{4K+k})^T v = 0, \forall k = 1, \qquad (3.18e)\\
& (e'_{4K+k})^T v' - (a_2)^T \operatorname{diag}\left(a_{k-1}^I\right) v' = 0, \\
& \forall k \in K, k \neq 1, \qquad (3.18f)
\end{aligned}
$$

where

$$
\begin{aligned}
& a_0 = \left[E_1^L, E_1^H, E_1^S, E_1^{HS}, \ldots, E_K^L, E_K^H, E_K^S, E_K^{HS}, \mathbf{0}_{1\times(K+1)}\right], \\
& a_k^I = e_{4k-3} + e_{4k-2} + e_{4k-1} + e_{4k}, \\
& a_1 = \left[\mathbf{0}_{1\times(4K-4)}, T_K^L, T_K^H, T_K^S, T_K^{HS}, \mathbf{0}_{1\times(K-1)}, 1, 0\right]^T, \\
& a_2 = \left[T_1^L, T_1^H, T_1^S, T_1^{HS}, \ldots, T_K^L, T_K^H, T_K^S, T_K^{HS}, \mathbf{1}_{1\times K}\right]^T, \\
& a_i^I = e'_{4i-3} + e'_{4i-2} + e'_{4i-1} + e'_{4i} + e'_{4K+i}.
\end{aligned}
$$

The homogeneous QCQP format of the problem can then obtained by defining $u = \left[v^T, 1\right]^T$ as follows:

$$\min_{u} \quad u^T M_0 u \tag{3.19a}$$

$$\begin{aligned}
\text{subject to:} \quad & u^T M_1 u = 0, i = 1, \ldots, 4K, && (3.19b)\\
& u^T M_2 u = 1, \forall k \in \mathcal{K}, && (3.19c)\\
& u^T M_3 u \leqslant T_d, && (3.19d)\\
& u^T M_4 u = 0, \forall k = 1, && (3.19e)\\
& u^T M_5 u = 0, \forall k \in \mathcal{K}, k \neq 1, && (3.19f)
\end{aligned}$$

where

$$M_0 = \begin{bmatrix} \mathbf{0}_{(5K+1)\times(5K+1)} & \frac{1}{2}a_0 \\ \frac{1}{2}\left(a_0\right)^T & 0 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} \operatorname{diag}\left(e_i\right) & -\frac{1}{2}e_i \\ -\frac{1}{2}\left(e_i\right)^T & 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} \mathbf{0}_{(5K+1)\times(5K+1)} & \frac{1}{2}a_k^P \\ \frac{1}{2}\left(a_k^P\right)^T & 0 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} \mathbf{0}_{(5K+1)\times(5K+1)} & \frac{1}{2}a_1 \\ \frac{1}{2}\left(a_1\right)^T & 0 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} \mathbf{0}_{(5K+1)\times(5K+1)} & \frac{1}{2}e_{3K+k} \\ \frac{1}{2}(e_{3K+k})^T & 0 \end{bmatrix}$$

$$M_5 = \begin{bmatrix} \mathbf{0}_{(5K)\times(5K)} & -\frac{1}{2}\left[(a_2)^T \operatorname{diag}\left(a'_{k-1}\right)\right]^T & \frac{1}{2}e'_{3K+k} \\ -\frac{1}{2}\left[(a_2)^T \operatorname{diag}\left(a'_{k-1}\right)\right] & 0 & 0 \\ \frac{1}{2}\left(e'_{3K+k}\right)^T & 0 & 0 \end{bmatrix}$$

It can be observed that the homogeneous QCQP problem (3.19a) is still not convex. Therefore, using the SDR approach, we relax the problem into a Semidefinite Programming (SDP) problem. We further define $U = uu^T$ and by dropping the rank constraint $\text{rank}(U) = 1$, the minimization problem (3.18a) is transformed to:

$$\min_{U} \quad \text{Tr}(M_0 U) \tag{3.20a}$$

$$\begin{aligned}
\text{subject to:} \quad & \text{Tr}(M_1 U) = 0, i = 1, ..., 4K, && \text{(3.20b)}\\
& \text{Tr}(M_2 U) = 1, \forall k \in \mathcal{K}, && \text{(3.20c)}\\
& \text{Tr}(M_3 U) \leqslant T_d, && \text{(3.20d)}\\
& \text{Tr}(M_4 U) = 0, \forall k = 1, && \text{(3.20e)}\\
& \text{Tr}(M_5 U) = 0, \forall k \in \mathcal{K}, k \neq 1, && \text{(3.20f)}\\
& U(5K+1, 5K+1) = 1, && \text{(3.20g)}\\
& U(5K+1, 5K+2) = 1, && \text{(3.20h)}\\
& U(5K+2, 5K+1) = 1, && \text{(3.20i)}\\
& U(5K+2, 5K+2) = 1, && \text{(3.20j)}\\
& U \succeq 0. && \text{(3.20k)}
\end{aligned}$$

SeDuMi [133] as a standard SDP software, is adopted in order to solve the SDP problem and to obtain our binary offloading decisions, and a randomization method is proposed to recover a rank-1 solution of the problem (3.15a) from U^* as the SDP problem (3.20a) optimal solution.

3.3.3 The randomization method for binary offloading decision

We got inspired this randomization method by [10] and modified it in order to apply it to our offloading strategy U^*. We again define $U = uu^T$ and $u(5K+2) = 1$ and by that, the last column of U is:

$$U(i, 5K+2) = u(i), i = 1, \ldots, 5K+2 \tag{3.21}$$

Thus, using the value of $U(i, 5K+2)$ we can determine the offloading strategy U^* for $i = 1, \ldots, 4K$. The optimal solution of the original problem (3.15a) could be extract

directly through U, if the rank of U were 1, otherwise, using a stochastic mapping method [10], we obtain a reasonable solution for our optimization problem (3.15a). The first $4K$ elements of the last column of U^*, $u_0(i), i = 1, \ldots, 4K$ are extracted and the following equation is obtained based on the (3.20c) and (3.20k) constraints:

$$u_0(4i-3) + u_0(4i-2) + u_0(4i-1) + u_0(4i) = 1, i = 1, \ldots, K, u_0(i) \in \mathbb{R}, u_0(i) > 0. \tag{3.22}$$

Therefore, $u_0(i)$ is considered as the probability of $u_0(i) = 1$ for $i = 1, \ldots, 4K$. Then by following the standard uniform distribution, K random numbers between 0 and 1 are generated. The offloading strategy that fulfills the constraint (3.15b) is accepted. For example, we have $u_0(1), u_0(2), u_0(3), u_0(4)$ as the probability of $w_1 = 1, x_1 = 1, y_1 = 1, z_1 = 1$, respectively for the first task. By generating the first random number ξ_1, the strategy for the first task was chosen based on the following equation:

$$\begin{cases} [w_1, x_1, y_1, z_1] = [1, 0, 0, 0], \text{ if } \xi_1 \leqslant u_0(1); \\ [w_1, x_1, y_1, z_1] = [0, 1, 0, 0], \text{ if } u_0(1) < \xi_1 \leqslant u_0(1) + u_0(2); \\ [w_1, x_1, y_1, z_1] = [0, 0, 1, 0], \text{ if } u_0(1) + u_0(2) < \xi_1 \leqslant u_0(1) + u_0(2) + u_0(3); \\ [w_1, x_1, y_1, z_1] = [0, 0, 0, 1], \text{ if } u_0(1) + u_0(2) + u_0(3) < \xi_1. \end{cases} \tag{3.23}$$

Note that the condition (3.15d) is not always fulfilled by the result of the mapping method, therefore, in case it does not satisfy the deadline T_d, it will be discarded. R random samples are generated to get a more accurate offloading strategy and based on the above procedure, potential solutions are achieved. Finally, the solution with minimum energy consumption is selected.

3.4 Numerical results

In this section, simulation results using Python are provided to validate the performance of our joint communication and computation cooperation offloading algorithm compared to the following approaches:

- **All local:** The tasks are fully executed on UE.
- **Computation cooperation:** The UER can just have a helper node role to execute the computation tasks.

- **Communication cooperation:** The UER can just have a relay node role to transmit the tasks to the MEC server.

The simulation parameters are listed in Table 3.1. The average energy consumption

Table 3.1: Simulation parameters [3]

Parameters	Value
Number of tasks (K)	25
Task deadline (T_d)	3 s
Data size of task k (b_k)	200 – 500 kb
Required CPU cycles per bit of task k (c_k)	0 – 50 cycles/bit
Channel bandwidth (B)	5 MHz
Channel gain between the UE and UER (H_{lh})	10^{-7}
Channel gain between the UE and MEC (H_{ls})	10^{-8}
Channel gain between the UER and MEC (H_{hs})	10^{-7}
Variance of the Gaussian channel noise (σ^2)	10^{-9}
Transmission power of UE (P^{UE}_{tra})	0.2 mW
CPU cycles frequency of UE (f_{UE})	0.1×10^9 cycles/s
CPU cycles frequency of UER (f_{UER})	0.5×10^9 cycles/s
CPU cycles frequency of MEC server (f_S)	2×10^9 cycles/s
Effective switched capacitance (λ)	10^{-25} F
Idle circuit power ($P^{\mathrm{UE}}_{\mathrm{wait}}$)	0.1 W

for 100 runs is shown in Figure 3.3. To test the offloading algorithm for different workloads, we sweep the required computations per bit of task data size, the variable c_k.

As can be observed, when the required computations per bit of the task k increases, the average energy consumption of all methods increases as well due to the more computation power required to complete the execution process. As the CPU resource requirements of tasks become larger, the performance benefit of our joint communication and computation cooperation approach is more observable.

In all local mode, the minimum energy consumption can be obtained, when c is less than 10 cycles/bit. By increasing c, all-local still meets the deadline demands but more energy is consumed in comparison with the other methods. For computation-intensive tasks which $c \geqslant 35$ cycles/bit, due to the inability to satisfy the deadline requirements, the all-local method fails as shown in Table 3.2. This clearly shows the need for a computation offloading framework.

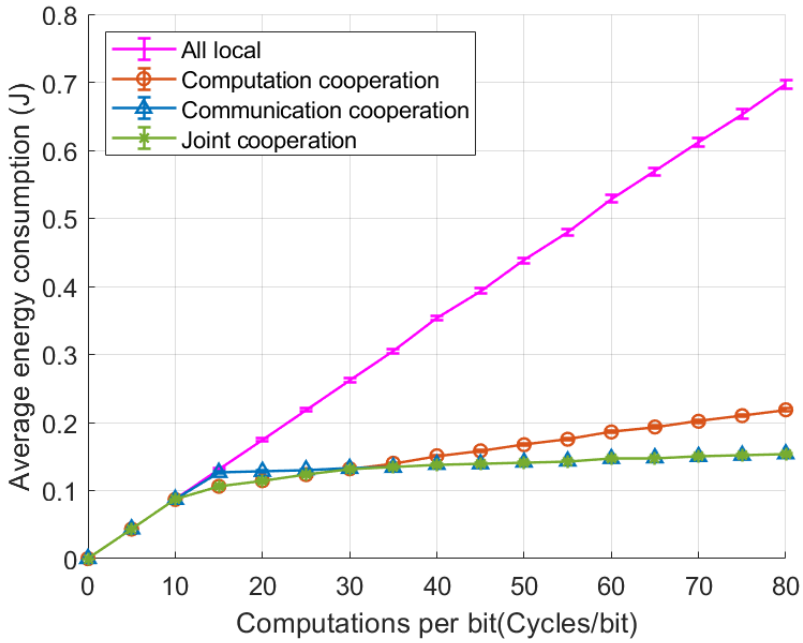


Figure 3.3: Average energy consumption vs tasks' complexity [3].

Table 3.2: Finish time of the local execution (Deadline: 3s) [3].

Computations per bit	Finish time of the last task [s]
0	0
5	0.4343
10	0.8697
15	1.3166
20	1.7473
25	2.1863
30	2.6203
35	3.0444
40	3.5343

The average energy consumption for different methods, with data size changing from 0 to 800 kb is shown in Figure 3.4. By increasing the size of the task, more computation resources are needed to finish the execution process. This leads to the increases of the average energy consumption of all methods. The energy consumption of our joint communication and computation cooperation is still lower compared to the other methods.

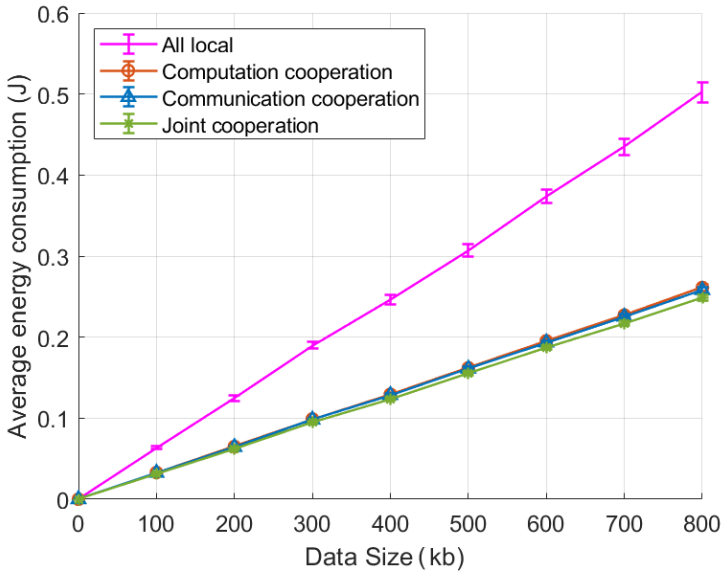


Figure 3.4: Average energy consumption vs tasks' data size [3].

3.5 Conclusion

In this chapter, we studied the optimization of computation offloading decision algorithm in a basic device-enhanced MEC system considering the tasks' sequential dependency and the time deadline required to finish the application. The final goal was minimizing the user's battery life time while executing the application. Simulation results show that our method can achieve the best performance by jointly optimizing the computation and communication resource allocations. For future work, a more realistic scenario with multiple users and general dependency graphs as well as the users' movements and incentive mechanisms for a better cooperation is investigated.

4 Mobility and Energy-aware Cooperative Computation Offloading

This chapter was previously published in:
Mehrabi, Mahshid, et al. "Mobility-and Energy-Aware Cooperative Edge Offloading for Dependent Computation Tasks." *Network 1.2* (2021): 191-214 [4].

4.1 Introduction

In Chapter 3, we first discussed a basic static scenario with three nodes. However, the significant and ongoing increase of the number of both mobile devices as well as applications with low-latency service requirements, such as IoT [134], TI [7], online gaming, and virtual or augmented reality and the MTC [135] have led to the need for reliable servic requirements. An example of such a dense 5G network architecture is shown in Figure 4.1.

Several research studies proposed utilizing D2D to enhance the energy efficiency performance of the system [136–142]; However, the mobility of users is neglected in these research studies which can cause several issues for service continuity and established D2D and cellular links. This handover and service migration can further increase the delay and energy consumption and negate the performance of

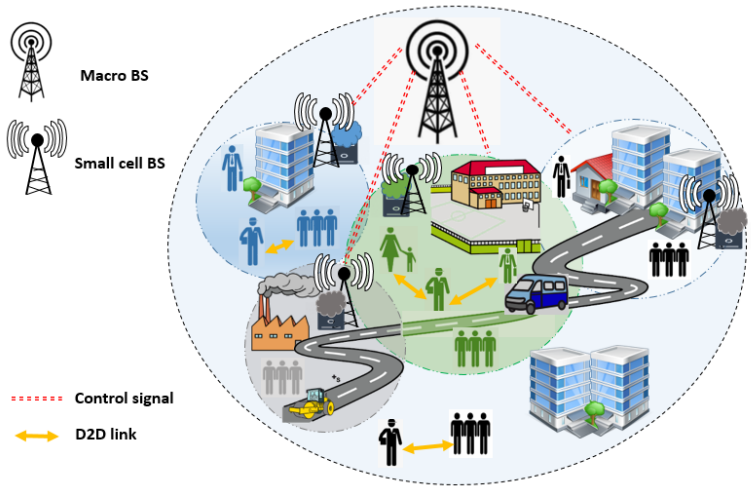


Figure 4.1: 5G ultra-dense network architecture.

the device-enhanced MEC system.

In order to assure the reliability as well as the service continuity of computation resources, in this chapter, we propose an online computation offloading framework considering the dependency relationships between computation tasks.

To the best of our knowledge, a few works studied the inter-dependency and mobility effects on offloading decision algorithm [143, 144] and our work was the first on online task offloading with task dependencies considering joint computation and communication cooperation in device-enhanced MEC systems which is published in [4]. A DL based algorithm called PECNet [5] is applied to predict users' paths by exploiting social interactions as well as the history of users' movement trajectories.

The following part of this chapter is organized as follows. Section 4.2 contains the detailed elaboration of the system model, including the task, mobility, communication and computation models. In Section 4.3, the task offloading decision algorithm is formulated and in Section 4.4, using the QCQP transformation and the SDR approach, we solve the dynamic computation offloading optimization problem. The system performance is investigated in Section 4.5 through various simulation scenarios, and finally, conclusions and possible paths for future work are stated in Section 4.6.

A generic three-layer fog computing network architecture is presented in [144] taking into account the mobility effects of the users by considering the sojourn time

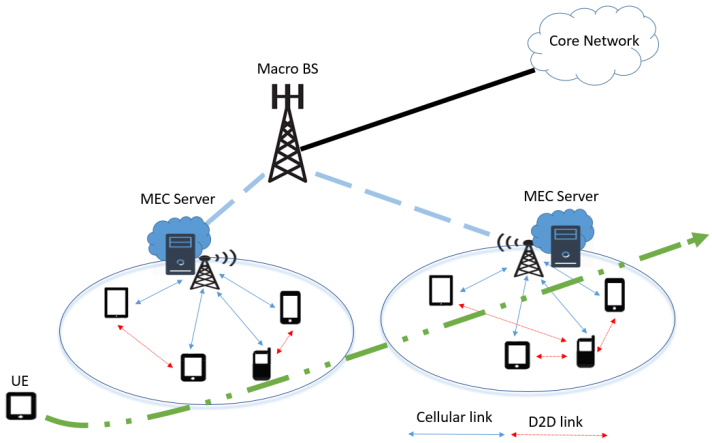


Figure 4.2: System model [4].

with exponential distribution for the coverage of fog nodes. However, in order to have a more accurate prediction, in reality, the sojourn time can be calculated using ML tools. We formulate the joint optimization of offloading decision algorithm and computation resource allocation problem as a MINLP aimed to reduce the migration probability. The previous studies neglected task dependency relationships which can cause several issues considering the movement of the users, especially in a D2D-assisted MEC network. Therefore, to enhance the performance of the offloading decision making in real-world scenarios, we propose an online joint computation and cooperation cooperative offloading framework by taking into account the mobility of users, applying a DL based algorithm to predict the users' trajectory exploiting social interactions between users, in addition to using the history of users' motion paths, and the inter-dependency relations of tasks.

We note for completeness that an approach specifically for mobile video streaming has been examined in [145], while energy harvesting has been the focus in [146] and an industrial IoT setting has been considered in [147]. Joint computation offloading and radio resource management has been explored in [148].

4.2 System model

4.2.1 Overview

A three-layer heterogeneous network with multiple devices and small cells is considered in this work where each small cell has been equipped with a BS and a MEC server attached to it as shown in Figure 4.2. UEs represent the devices with the computation-intensive tasks and the devices with sufficient computation and communication resources are helpers/relays which are denoted as UHs. We randomly distribute all devices and define $\mathcal{U} = \{UH_1, UH_2, \ldots, UH_I\}$ as the set of UHs. The maximum service coverage for each $UH_i \in \mathcal{U}$ can be shown by R_i and a UE can only connect to one helper node at the same time. Same as above, the set $\mathcal{M} = \{S_1, S_2, \ldots, S_M\}$ is defined for the edge servers which R_m is the service coverage of m-th server represented as S_m. In our model, Macro Base Station (MBS) has the information of the channel state and the user positions and controls the offloading decision process. The notations used in this chapter are summarized in Table 4.1.

4.2.2 Task model

One example of a task with a general dependency graph would be a video navigation application which would be be running on a smartphone [149]. This computation-intensive application is denoted as $\mathcal{K} = \{1, 2, \ldots, K\}$. For every task, an associated set of parameters $A_k = \{b_k, c_k, d_k\}$ represents the required computation and memory resources required, respectively, with b_k denoting the bit-wise data size, c_k the necessary CPU cycles per bit of data, and d_k the total amount of computation resources required to execute, which equals $b_k \times c_k$. An application-wide deadline for the entire execution process is introduced as $T_d^{\max}$.

The nature of the task's Directed Acyclic Graph (DAG) with its dependency relationship directly implies an order of execution for the applications' modules. The predecessors of any given task may introduce wait times, i.e. until all predecessors have finished, see Fig 4.3. Thus the concepts of start and finish time of a task are introduced in order to model this influence on the computation offloading decision algorithm:

Table 4.1: Parameter notations [4]

Symbol	Definition
I	Number of helpers/relays
M	Number of MEC servers
K	Number of tasks
$\mathcal{U}$	Set of helper/relay nodes
$\mathcal{M}$	Set of MEC servers
$\mathcal{K}$	Set of computation tasks
b_k	Data size of computation task k
c_k	Required computation resources per bit of task k
d_k	Computation resources required to execute task k
$T_d^{\max}$	Deadline for execution of set of K tasks
B	Bandwidth of wireless channel
H_k	Wireless channel gain
σ^2	AWGN noise variance
P_k^{tr}	Transmission power for task k
P_k^{wait}	*UE*'s idle circuit power
f_k^{UE}	*UE* CPU cycle frequency allocated to execute task k
$f_k^{UH_i}$	CPU cycle frequency of helper node UH_i
$f_k^{S_m}$	CPU cycle frequency of MEC server m
H	Set of helper nodes' selection variables
S	Set of MEC servers' selection variables
HS	Set of relays and MEC servers' selection variables
X_t^{UE}	UE position at time t
$X_t^{UH_i}$	Position of UH_i at time t
R_i	Distance between *UE* and UH_i
$T_{s,i,t}$	*UE* sojourn time in coverage of helper node UH_i

- *Finish time* is the time instant of execution completion of task k:

$$FT_k = ST_k + T_k^{exe}, \tag{4.1}$$

  where ST_k is the start time of task k (see immediately below) and T_k^{exe} is the execution duration (span) for task k.

- *Start time* is the time instant when the execution of task k can commence at the earliest:

$$ST_k = \begin{cases} \max\limits_{j \in \varphi(k)} FT_j & \varphi(k) \neq \varnothing, \\ 0 & \varphi(k) = \varnothing \end{cases} \tag{4.2}$$

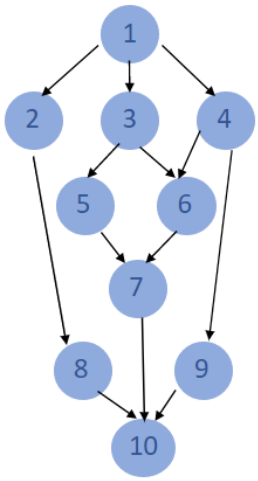


Figure 4.3: Example illustration of general task dependency graph specifying the required task execution order for an application with a total of $K = 10$ tasks. Tasks $k = 2, 3$, and 4 depend on the prior completion of task $k = 1$. The last task $K = 10$ has to be completed by the deadline $T_d^{\max}$ [4].

where $\varphi(k)$ contains all predecessors of the task k. (4.2) implies that the execution of tasks without predecessors can be started without delay, whereas the start time of any subsequent tasks is lower-bounded, specifically by the largest finish time of the union over the computational graph of respective preceding tasks.

4.2.3 Communication model

For the transmission of a task k, the achievable up-link data or information rate can be obtained based on the Shannon-Hartley theorem, i.e.

$$r_k = B \log_2 \left(1 + \frac{P_k^{tr} H_k}{\sigma^2}\right), \tag{4.3}$$

with the channel bandwidth B between sender and receiver. Possible sending entities include UE and UHs; the receivers could be UHs or MEC servers, among others. P_k^{tr} is the transmission power for task k, and H_k denotes the channel gain between sender and receiver during transmission of task k. Default values in the evaluation are set to $H_k = 10^{-7}$ for UE to UH$_i$, $H_k = 10^{-8}$ for UE to server, and $H_k = 10^{-7}$ for UH$_i$ to server); additionally, σ^2 is the variance of the Gaussian channel noise. Since in our scenario, the packets describing the tasks to be executed are of small size, the

down-link data rate is neglected.

4.2.4 Computation model

In total, four decision options in our dynamic offloading decision algorithm need to be determined. These correspond to the various paths of the execution, namely local execution (i.e., on the UE), remote execution (on a helper device) and remote execution on the MEC server, in which case the tasks can be transmitted either directly to the MEC server by UE, or via relays. All of these relevant decision options are defined now.

4.2.4.1 Local execution

For a local execution of task k at the UE, the execution time is fully given by

$$T_k^l = \frac{b_k c_k}{f_k^{UE}}, \tag{4.4}$$

where, the data size for each task k is b_k in bits, c_k the required computational CPU cycles per bit, and f_k^{UE} the computation capacity allocated in the user's device for task execution. Based on (5.10) and the effective switched capacitance, modeled as a (constant) factor depending on the chip architecture and denoted by λ (cf. [150]), the energy consumption in this purely local execution mode can be estimated as

$$E_k^l = \lambda b_k c_k \left(f_k^{UE}\right)^2. \tag{4.5}$$

4.2.4.2 Helper execution

In the second mode, task execution is comprised of two steps. First, task k is transferred from UE to helper node UH_i via D2D communication, after which is executed on UH_i. Thus, the total time required for execution is given by

$$T_k^{h_i} = \frac{b_k}{r_k^{UH_i}} + \frac{d_k}{f_k^{UH_i}}, \tag{4.6}$$

in which $r_k^{UH_i}$ is the communication transmission rate from UE to UH_i and $f_k^{UH_i}$ is the computation capacity of UH_i, provisioned for the execution of task k. Deriving from (4.6), the energy consumption of the helper execution mode is driven by three

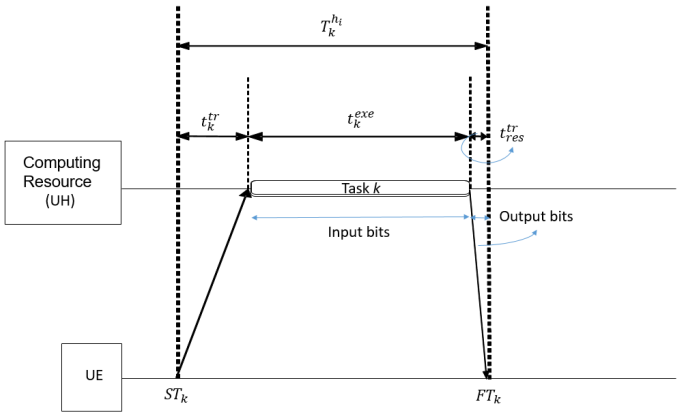


Figure 4.4: Timing diagram for task execution by an external computing resource (helper UH) [4].

phases. The task transmission from UE to helper, the task execution phase, and finally the energy consumed by the UE while waiting to receive the result back from the helper:

$$E_k^{h_i} = P_k^{tr}\left(\frac{b_k}{r_k^{UH_i}}\right) + \lambda\left(f_k^{UH_i}\right)^2 d_k + P_k^{wait}\left(\frac{d_k}{f^{UH_i}}\right), \tag{4.7}$$

where P_k^{tr} represents the transmission power of the UE, $r_k^{UH_i}$ is the transmission rate from the UE to UH_i, and P_k^{wait} is the idle circuit power while the device is waiting for the result.

As described above, a total of I helpers are available for the UE. Therefore, $h_k^i = 1, h_k^i \in H$, means that UE has chosen to offload task k to UH_i, where $H = \{h_k^i \,|\, k \in \mathcal{K}, i \in \mathcal{U}\}$ is the variable set for helper node selection. Taking into consideration that any task k can only be offloaded to one helper at the same time in any efficient manner, consequentially this constraint follows:

$$\sum_{i \in H} h_k^i \leqslant 1. \tag{4.8}$$

To illustrate this concept, we have shown the offloading process of a task k to the helper in Figure 4.4: the time t_k^{tr} is needed to transmit task k to helper node i, and t_k^{exe} is required for the execution of the task on the helper. For completeness, the time t_{res}^{tr} would be needed for the return trip of the result back from helper node i to the UE, but this is neglected in our model due to the typically small size of output.

4.2.4.3 Server execution

As hinted at above, two fundamental paths exist for task execution on the server:

- **Direct offloading to the MEC server:** Without additional hops, the execution delay for task k is as follows

$$T_k^{S_m} = \frac{b_k}{r_k^{S_m}} + \frac{d_k}{f_k^{S_m}}, \tag{4.9}$$

with $r_k^{S_m}$ as the transmission rate from UE to server S_m, and $f_k^{S_m}$ denoting server S_m's CPU computation cycle frequency for execution. For the resulting energy usage, the following equation results:

$$E_k^{S_m} = P_k^{tr}\left(\frac{b_k}{r_k^{S_m}}\right) + P_k^{wait}\left(\frac{d_k}{f_k^{S_m}}\right). \tag{4.10}$$

Generally, the MEC server's energy consumption is generally ignored in (4.10), since these servers do typically not rely on limited battery power.

With M servers available in the area to the UE, the UE has a placement choice to make. This results in $s_k^m = 1$ for one server S_m, with the set $s_k^m \in S, S = \{s_k^m \,|\, k \in \mathcal{K}, m \in \mathcal{M}\}$ containing all selection variables for MEC server nodes. Only one task may be served at any given time at each MEC server, so this following constraint mirrors the one introduced above:

$$\sum_{m \in \mathcal{M}} s_k^m \leqslant 1. \tag{4.11}$$

- **Offloading from *UE* via relay UH_i to the MEC server:** The offloading UE first sends the task to the relay UH_i, which then forwards it to the server S_m. This results in delay consisting of three steps, two for transmissions and one for computation:

$$T_k^{S_{i,m}} = \frac{b_k}{r_k^{UH_i}} + \frac{b_k}{r_k^{S_{m,i}}} + \frac{d_k}{f_k^{S_m}}, \tag{4.12}$$

with transmission rate $r_k^{S_{m,i}}$ from UH_i to server S_m. Then, the energy consump-

tion can be written as

$$E_k^{s_{i,m}} = P_k^{tr}\left(\frac{b_k}{r_k^{UH_i}}\right) + P_k^{wait}\left(\frac{b_k}{r_k^{S_{m,i}}} + \frac{d_k}{f_k^{S_m}}\right) + P_k^{tr,i}\left(\frac{b_k}{r_k^{S_m}}\right), \tag{4.13}$$

where the transmission power of UH_i is given by $P_k^{tr,i}$ and the result is assumed to be sent back directly from server S_m to the UE.

Mirroring above's approach, the selection variables $s_k^{i,m} = 1, s_k^{i,m} \in HS$ for offloading task k to S_m via UH_i are presented, where the set $HS = \{s_k^{i,m} \mid k \in \mathcal{K}, i \in \mathcal{I}, m \in \mathcal{M}\}$ contains all selection variables corresponding to relays and MEC servers, respectively. Again, since every single helper node could only provide service to one task at any one time instance, the constraint follows as

$$\sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{I}} s_k^{i,m} \leqslant 1. \tag{4.14}$$

4.2.5 Mobility model

Recent literature includes learnings that most user trajectories contain similar patterns [143, 151]. We incorporate this insight to achieve effective task offloading with realistic mobility-awareness. To this end, we make use of ML to predict paths of both the UEs and UHs to accurately estimate their available service coverage time. Specifically, we employ the DL based method PECNet [5] to predict socially compliant trajectories which infer users' destinations. DL [152] is a branch of ML methods called Neural Network (NN), and had significant impact in computer vision, natural language processing and control. NNs are (often high-dimensional) mappings from an input space to an output space. Layers are stacked from input to output and connected in various forms (mot notably, feed-forward). A layer itself is made up from simple units, "Neurons", that each perform a nonlinear operation on the sum of their weighted inputs. Every single connection to every neuron has a weight that is calibrated during the learning process to fit data, according to some loss function. DL is typically referred to in situations with more than three layers. Being trained on real world data, PECNet enhances the plausibility of the predicted trajectories in addition to using the historical data of motion paths. Altogether, this yields coherent trajectories for users. PECNet divides the prediction problem into two parts: Firstly, it estimates the potential destinations of users using a Variational Auto-Encoder (VAE) for

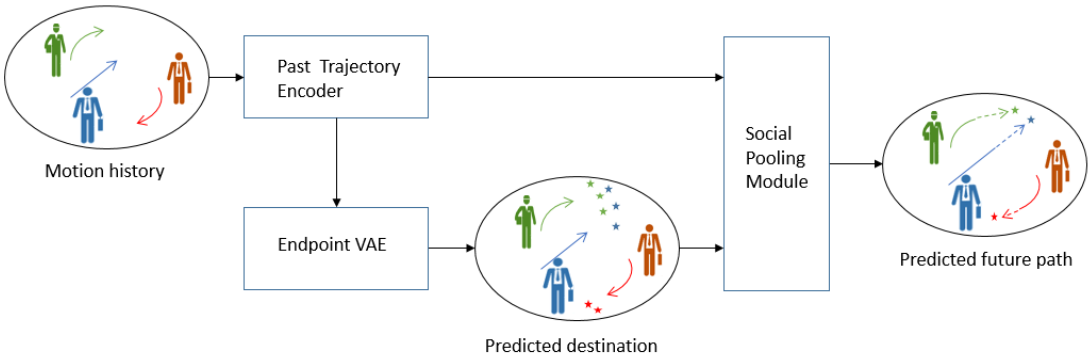


Figure 4.5: PECNet system model [5]: Past trajectory encoder and endpoint estimation variational encoder (VAE) feed into social pooling module to predict paths.

endpoint prediction. Scondly, it predicts socially compliant trajectories while jointly considering the motion history and potential destinations of all users in the scene.

The PECNet system model, which is shown in Figure 4.5, includes three key elements: a past trajectory encoder, an endpoint VAE, and a social pooling module. The user's motion histories are encoded via the past trajectory encoder, feeding into the endpoint VAE for an estimation of the user's destination. After that, the social pooling module uses all users' encoded past trajectoriess and estimated destinations to jointly predict the future paths of all users in the scene. The final output are user-specific paths whose future segments (i.e., the predictions) strongly depend on the past locations (i.e., the inputs).

In our scenario, the initial positions of the UE and the helper node i at time t are defined as $X_t^{\mathrm{UE}} = (x_t, y_t)$ and $X_t^{\mathrm{UH}_i} = \left(x_t^i, y_t^i\right)$, respectively. Then, the trajectories of the UE and helpers from time $(t - n + 1)$ to t are:

$$\mathbf{X}^{\mathrm{UE}} = \left\{X_{t-n+1}^{\mathrm{UE}}, X_{t-n+2}^{\mathrm{UE}}, \ldots, X_t^{\mathrm{UE}}\right\} \tag{4.15}$$

$$\mathbf{X}^{\mathrm{UH_i}} = \left\{X_{t-n+1}^{\mathrm{UH_i}}, X_{t-n+2}^{\mathrm{UH_i}}, \ldots, X_t^{\mathrm{UH_i}}\right\}. \tag{4.16}$$

This information can be directly collected by the edge server. PECNet takes these trajectories of UE as well as helpers, and outputs the predicted movements from time $(t + 1)$ to $(t + l)$:

$$\hat{\mathbf{Y}}^{\mathrm{UE}} = \left\{\hat{Y}_{t+1}^{\mathrm{UE}}, \hat{Y}_{t+2}^{\mathrm{UE}}, \ldots, \hat{Y}_{t+l}^{\mathrm{UE}}\right\} \tag{4.17}$$

$$\widehat{\Upsilon}^{\mathrm{UH_i}} = \left\{ \widehat{Y}^{\mathrm{UH_i}}_{t+1}, \widehat{Y}^{\mathrm{UH_i}}_{t+2}, \ldots, \widehat{Y}^{\mathrm{UH_i}}_{t+l} \right\}, \tag{4.18}$$

where $\widehat{Y}^{\mathrm{UE}}_t = (\hat{x}_t, \hat{y}_t)$ and $\widehat{Y}^{\mathrm{UH}_i}_t = \left(\hat{x}^i_t, \hat{y}^i_t\right)$. A D2D link between the UE and the helper i at time t can be established if the UE is in the coverage area of helper i which has radius R_i, i.e., if

$$\sqrt{\left(\hat{x}_t - \hat{x}^i_t\right)^2 + \left(\hat{y}_t - \hat{y}^i_t\right)^2} \leqslant R_i. \tag{4.19}$$

The sojourn time in helper i's coverage of UE from time t forward is then given as

$$T^{h_i}_{s,t} = \max\ l,\ \text{s.t.}\ \sqrt{(\hat{x}_\tau - \hat{x}^i_\tau)^2 + (\hat{y}_\tau - \hat{y}^i_\tau)^2} \leqslant R_i,\ \ \forall \tau \in [t, t+l], \tau \in \mathbb{Z}. \tag{4.20}$$

The same process can be utilized analogously to obtain the sojourn time of the UE as well as UH_i with respect to the coverage of MEC server S_m.

Due to the users' mobility, the service coverage is limited, introducing a hard problem; in our method, we tackle this by defining the sojourn and finish times of the tasks in such a way that users never choose a destination if the period of the coverage availability is less than the time required for task execution.

4.3 Dynamic computation offloading problem formulation

The joint optimization of the computation and communication cooperation in our framework for the execution of task k considering the offloading decision options defined above and the deadline constraints for execution of tasks can be defined as follows

$$E^{exe}_k = x_k E^l_k + h^i_k E^{h_i}_k + s^m_k E^{s_m}_k + s^{i,m}_k E^{s_{i,m}}_k, \tag{4.21}$$

where x_k, h^i_k, s^m_k and $s^{i,m}_k$ denotes the binary decision variables for encoding the decision options. Only one of these variables, corresponding to the computation entity, can be 1 per each task and the rest are 0. The task execution time can be formulated as

$$T^{exe}_k = x_k T^l_k + h^i_k T^{h_i}_k + s^m_k T^{s_m}_k + s^{i,m}_k T^{s_{i,m}}_k. \tag{4.22}$$

Hence, the finish time of task k is

$$FT_k = T_k^{exe} + ST_k. \tag{4.23}$$

The optimization problem for minimal energy cost can be formulated considering the application's execution deadline constraint, the users' movements, and the task dependency relations as:

$$\begin{aligned}
OP1: \quad & \min_{\alpha,\beta} \quad E_{tot}^{exe} = \sum_{i=1}^{K} E_k^{exe}, \quad \forall k \in \mathcal{K}, \\
s.t: & \; C1: x_k, h_k^i, s_k^m, s_k^{i,m} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \; \forall i \in \mathcal{I}, \; \forall m \in \mathcal{M} \\
& \; C2: x_k + \sum_{i=1}^{I} h_k^i + \sum_{m=1}^{M} s_k^m + \sum_{m=1}^{M}\sum_{i=1}^{I} s_k^{i,m} = 1, \quad \forall k \in \mathcal{K} \\
& \; C3: ST_k = \begin{cases} \max\limits_{j \in \varphi(k)} FT_j & \varphi(k) \neq \varnothing, \quad \forall k \in \mathcal{K} \\ 0 & \varphi(k) = \varnothing, \quad \forall k \in \mathcal{K} \end{cases} \\
& \; C4: FT_K \leqslant T_d^{\max}, \\
& \; C5: FT_k \leqslant T_{s,k}, \quad \forall k \in \mathcal{K},
\end{aligned} \tag{4.24}$$

where $\alpha = [x_1, h_1^1, \ldots, h_1^I, s_1^1, \ldots, s_1^M, s_1^{1,1}, \ldots, s_1^{I,M}, \ldots, x_K, h_K^1, \ldots, h_K^I, s_K^1, \ldots, s_K^M, s_K^{1,1}, \ldots, s_K^{I,M}]$ and $\beta = [ST_1, ST_2, \ldots, ST_K]$ denote the offloading decision variables and start time sets, respectively. C3 demonstrates the start time constraint for each task k according to the predecessors. For tasks with no predecessor, the execution process can get started immediately, while for the rest, the maximum finish time of the respective predecessors determine the start time of the execution. Constraint C4 denotes that the finish time of the last task K which shows the time needed for the whole execution process, should satisfy the time deadline $T_d^{\max}$ constraint of the application. C5 shows that an offloading destination may be selected if only the sojourn time for the device in the range of the computation resource is more than the time needed to finish executing task k.

The sojourn time of the different offloading methods can be obtained as follows

$$T_{s,k} = \begin{cases} T_{s,t}^{h_i}, & if \quad h_k^i = 1 \\ T_{s,t}^{s_m}, & if \quad s_k^m = 1 \\ \min\{T_{s,t}^{h_i}, T_{s,t}^{s_m}, T_{s,t}^{s_{i,m}}\}, & if \quad s_k^{i,m} = 1. \end{cases} \tag{4.25}$$

Where $T_{s,t}^{h_i}$ and $T_{s,t}^{s_m}$ represent the user's sojourn time when it is in the range of helper node UH_i and the MEC server S_m respectively, while $T_{s,t}^{s_{i,m}}$ is helper UH_i sojourn time in the coverage area of the MEC server S_m. According to C5 it is shown that an offloading destination may be selected if only the sojourn time for the device in the range of the computation resource is more than the time needed to finish executing of task k. However, in case the relay option is chosen, the execution completion time of the task should be less than the minimum sojourn time between the UE and UH_i, UH_i and the MEC server S_m, as well as the UE and the MEC server S_m.

The binary constraints make the optimization problem (OP1) a non-convex MINLP problem which cannot be solved in polynomial time [153]. In order to efficiently find a feasible solution, we first transform the problem to an equivalent QCQP format. Then, using a SDR and stochastic mapping method, the binary offloading decisions can be recovered.

4.4 Solution of dynamic computation offloading optimization problem

The first step towards finding the solution for the Optimization Problem (OP1) is converting the integer constraints to a quadratic formulation:

$$x_k\left(x_k-1\right)=0,\ h_k^i\left(h_k^i-1\right)=0,\ s_k^m\left(s_k^m-1\right)=0,\ s_k^{i,m}\left(s_k^{i,m}-1\right)=0,\ \forall k\in K,\forall i\in I,\forall m\in M. \tag{4.26}$$

The QCQP transformation of OP1 can be then expressed as follows considering the quadratic constraint formulations

$$
\begin{aligned}
OP2: \quad & \min_{\alpha,\beta} \quad E_{tot}^{exe} \\
& s.t: C1: x_k\left(x_k-1\right)=0,\ h_k^i\left(h_k^i-1\right)=0, \quad s_k^m\left(s_k^m-1\right)=0,\ s_k^{i,m}\left(s_k^{i,m}-1\right)=0, \\
& \qquad \forall k \in K, \forall i \in I, \forall m \in M \\
& \qquad C2: x_k+\sum_{i=1}^{I} h_k^i+\sum_{m=1}^{M} s_k^m+\sum_{m=1}^{M}\sum_{i=1}^{I} s_k^{i,m}=1, \quad \forall k \in \mathcal{K} \\
& \qquad C3: ST_k=0 \quad \forall j \in \varphi(k), \varphi(k)=\varnothing,\ \forall k \in \mathcal{K} \\
& \qquad\quad ST_k-FT_j \geqslant 0, \quad \forall j \in \varphi(k), \varphi(k) \neq \varnothing,\ \forall k \in \mathcal{K} \\
& \qquad C4: FT_K \leqslant T_d^{\max}, \\
& \qquad C5: FT_k \leqslant T_{s,k}, \quad \forall k \in \mathcal{K},
\end{aligned}
\tag{4.27}
$$

where we reformulated $C1$ and vectorized $C3$.

Furthermore, we introduce a vector v with dimension $((2+I+M+IM)K+1)\times 1$ as $v=[\alpha,\beta,1]^T$, and a standard unit vector e_j with the jth entry equal to 1 and dimension

$$((2+I+M+IM)K+1)\times 1.$$

Thus, the QCQP transformation of OP2 can be written as

$$
\begin{aligned}
OP3: \quad & \min_{v} \quad (n_0)^T v \\
& s.t: C1: v^T \mathrm{diag}(e_j) v-(e_j)^T v=0,\ j=1,\ldots,(1+I+M+IM)K \\
& \qquad C2: (n_{1k})^T v=1,\ \forall k \in \mathcal{K} \\
& \qquad C3: (e_{(1+I+M+IM)K+1})^T v=0, \quad \forall j \in \varphi(k), \varphi(k)=\varnothing,\ \forall k \in \mathcal{K} \\
& \qquad C3: (e_{(1+I+M+IM)K+k})^T v-(n_T)^T \mathrm{diag}(n_3) v \geqslant 0, \forall j \in \varphi(k), \varphi(k) \neq \varnothing\ \forall k \in \mathcal{K} \\
& \qquad C4: (n_2)^T v \leqslant T_d^{\max}, \\
& \qquad C5: (n_T-n_{T_s})^T \mathrm{diag}(n_{1k}) v \leqslant 0, \quad \forall k \in \mathcal{K},
\end{aligned}
\tag{4.28}
$$

where

$$
\begin{aligned}
v =&[x_1, h_1^1, \ldots, h_1^I, s_1^1, \ldots, s_1^M, s_1^{1,1}, \ldots, s_1^{I,M}, \ldots, x_K, h_K^1, \\
&\ldots, h_K^I, s_K^1, \ldots, s_K^M, s_K^{1,1}, \ldots, s_K^{I,M}, ST_1, \ldots, ST_K, 1]^T, \\
n_0 =&[E_1^l, E_1^{h_1}, \ldots, E_1^{h_I}, E_1^{s_1}, \ldots, E_1^{s_M}, E_1^{s_{1,1}}, \ldots, E_1^{s_{I,M}}, \\
&\ldots, E_K^l, E_K^{h_1}, \ldots, E_K^{h_I}, E_K^{s_1}, \ldots, E_K^{s_M}, E_K^{s_{1,1}}, \ldots, E_K^{s_{I,M}}, \mathbf{0}_{1\times(K+1)}]
\end{aligned}
\tag{4.29}
$$

are the decisions set to determine which resource destinations the tasks should be offloaded to, respectively; n_0 is the energy consumption vector associated with a full set of decisions. Note that it is extended by a row of zeros for the optimizer.

$$
\begin{aligned}
n_{1k} =&e_{(1+I+M+IM)(k-1)+1} + \cdots + e_{(1+I+M+IM)k-1} + e_{(1+I+M+IM)k} + e_{(1+I+M+IM)K+k} \\
n_2 =&[0_{1\times(2+I+M+IM)(K-1)}, T_K^l, T_K^{h_1}, \ldots, T_K^{h_I}, T_K^{s_1}, \ldots, T_K^{s_M}, T_K^{I,s_{1,1}}, \ldots, T_K^{I,s_{I,M}}, 0_{1\times(K-1)}, 1, 0]^T \\
n_3 =&e_{(2+I+M+IM)(j-1)+1} + \cdots + e_{(2+I+M+IM)j-1} + e_{(2+I+M+IM)j} + e_{(2+I+M+IM)K+j} \\
n_T =&[T_1^l, T_1^{h_1}, \ldots, T_1^{h_I}, T_1^{s_1}, \ldots, T_1^{s_M}, T_1^{s_{1,1}}, \ldots, T_1^{s_{I,M}}, \ldots, T_K^l, T_K^{h_1}, \ldots, T_K^{h_I}, T_K^{s_1}, \ldots, T_K^{s_M}, T_K^{s_{1,1}}, \\
&\ldots, T_K^{I,s_{I,M}}, 1_{1\times(K+1)}]^T \\
n_{T_s} =&[T_d^{\max}, T_{s,t}^{h_1}, \ldots, T_{s,t}^{h_I}, T_{s,t}^{s_1}, \ldots, T_{s,t}^{s_M}, T_{s,t}^{s_{1,1}}, \ldots, T_{s,t}^{s_{I,M}}, \ldots, T_d^{\max}, T_{s,t}^{h_1}, \ldots, T_{s,t}^{h_I}, T_{s,t}^{s_1}, \ldots, T_{s,t}^{s_M}, \\
&T_{s,t}^{s_{1,1}}, \ldots, T_{s,t}^{s_{I,M}}, 1_{1\times(K+1)}]^T
\end{aligned}
\tag{4.30}
$$

are the constraints reformulated equations, consisting of standard unit vectors and sojourn times.

By defining $g = [v^T 1]$, we change the OP3 to a *homogeneous* QCQP. We further take $n_1' = \text{diag}(n_{1k})$, $n_3' = \text{diag}(n_3)$, $a = (2 + I + M + IM)K$, $b = (1 + I + M + IM)K + 1$, and

$c = (1 + I + M + IM)K + k$ for ease of understanding the equations. Therefore,

$$
\begin{aligned}
OP4: \quad & \min_{g} \quad g^T (M_0)\, g \\
s.t: C1: & \; g^T M_1 g = 0,\ j = 1, \ldots, (1 + I + M + IM)\, K \\
& C2: g^T M_2 g = 1, \forall k \in \mathcal{K}, \\
& C3: g^T M_3 g = 0, \quad \forall j \in \varphi(k), \varphi(k) = \varnothing,\ \forall k \in \mathcal{K} \\
& C3: g^T M'_3 g \geqslant 0, \quad \forall j \in \varphi(k), \varphi(k) \neq \varnothing,\ \forall k \in \mathcal{K} \\
& C4: g^T M_4 g \leqslant T_d^{\max}, \\
& C5: g^T M_5 g \leqslant 0, \quad \forall k \in \mathcal{K},
\end{aligned}
\tag{4.31}
$$

OP4 is in the standard format of the QCQP solvers along with the constraints required for the different device roles in our framework. For this, all constraints are converted into matrix form, i.e.,

$$
M_0 = \begin{bmatrix} 0_{(a+1)\times(a+1)} & \frac{1}{2} n_0 \\ \frac{1}{2} (n_0)^T & 0 \end{bmatrix}
$$

$$
M_1 = \begin{bmatrix} \mathrm{diag}(e_j) & -\frac{1}{2} e_j \\ -\frac{1}{2} (e_j)^T & 0 \end{bmatrix}
$$

$$
M_2 = \begin{bmatrix} 0_{(a+1)\times(a+1)} & \frac{1}{2} n_{1k} \\ \frac{1}{2} (n_{1k})^T & 0 \end{bmatrix}
$$

$$
M_3 = \begin{bmatrix} 0_{(a+1)\times(a+1)} & \frac{1}{2} e_b \\ \frac{1}{2} (e_b)^T & 0 \end{bmatrix}
$$

$$
M'_3 = \begin{bmatrix} 0_{a\times a} & -\frac{1}{2} \left[(n_T)^T n'_3 \right]^T & \frac{1}{2} (e_c)^T \\ -\frac{1}{2} \left[(n_T)^T n'_3 \right] & 0 & 0 \\ \frac{1}{2} (e_c)^T & 0 & 0 \end{bmatrix}
$$

$$
M_4 = \begin{bmatrix} 0_{(a+1)\times(a+1)} & \frac{1}{2} (n_2) \\ \frac{1}{2} (n_2)^T & 0 \end{bmatrix}
$$

$$
M_5 = \begin{bmatrix} 0_{(a+1)\times(a+1)} & \frac{1}{2} \left[(n_T - n_{T_s})^T n'_{1k} \right]^T \\ \frac{1}{2} \left[(n_T - n_{T_s})^T n'_{1k} \right] & 0 \end{bmatrix}.
$$

The final conversion step is defining $G = gg^T$ which is the symmetric, positive semidef-

inite matrix.

$$
\begin{aligned}
OP5: \quad & \min_{G} \quad \mathrm{Tr}(M_0 G) \\
s.t: \; & C1: \mathrm{Tr}(M_1 G) = 0,\ j = 1, \ldots, (1 + I + M + IM)K, \\
& C2: \mathrm{Tr}(M_2 G) = 1, \forall k \in \mathcal{K} \\
& C3: \mathrm{Tr}(M_3 G) = 0, \quad \forall j \in \varphi(k), \varphi(k) = \varnothing,\ \forall k \in \mathcal{K} \\
& C3: \mathrm{Tr}(M_3' G) \geqslant 0, \quad \forall j \in \varphi(k), \varphi(k) \neq \varnothing,\ \forall k \in \mathcal{K} \\
& C4: \mathrm{Tr}(M_4 G) \leqslant T_d^{\max}, \\
& C5: \mathrm{Tr}(M_5 G) \leqslant 0, \quad \forall k \in \mathcal{K} \\
& C6: G[a+1, a+1] = 1 \\
& C7: G[a+1, a+2] = 1 \\
& C8: G[a+2, a+1] = 1 \\
& C9: G[a+2, a+2] = 1 \\
& C10: \mathrm{rank}(G) = 1.
\end{aligned}
\tag{4.32}
$$

4.4.1 Energy-efficient task offloading (EETO) algorithm

Considering the inter-task dependency and the application completion deadline constraints introduced above, a stochastic mapping method is proposed in this section to obtain the optimized offloading strategy. Dropping the last non-convex constraint of rank 1, SDR is employed to get an approximate solution $\tilde{G}$, the last row of which includes $[\alpha, \beta, 1, 1]$. In case $\mathrm{rank}(\tilde{G}) = 1$, α is directly extracted as offloading decision for all tasks from the last row of $\tilde{G}$. Otherwise, a probability-based stochastic mapping method is deployed to recover the solution. The largest value of each offloading decision is selected for task k, from the elements group with index k in α and is labeled as t_1, t_2, t_3, and t_4. Then, t_1, t_2, t_3, and t_4 are mapped based on the probability based stochastic mapping method:

$$q_1 t_1 + q_2 t_2 + q_3 t_3 + q_4 t_4 = 1 \tag{4.33}$$

$$Q_1 = q_1 t_1, Q_2 = q_2 t_2, Q_3 = q_3 t_3, Q_4 = q_4 t_4, \tag{4.34}$$

where Q_1, Q_2, Q_3, and Q_4 are the probabilities of the corresponding offloading decision being 1. We randomly set $t_i = 1$ according to the probabilities Q_i and the rest in α are set to 0. For example, when $I = 2, M = 2$, and $K = 3$, then the dimension of vector α would be $(1 + I + M + IM)K = 27$. We have $1 + I + M + IM = 9$ elements per each task and if the solution were not rank 1, each element could be a value between 0 and 1 and the sum of the 9 elements equals 1. According to ((4.33)) and ((4.34)), we stochastically map their probability that the corresponding offloading decision would be selected with Q_1, Q_2, Q_3, and Q_4, respectively. Then, one of the two numbers is randomly set to 1 while the rest of the 9 elements are set to 0. This means that only one offloading decision would be selected for each task. The same stochastic mapping method is performed for all tasks and after making decisions, the offloading strategy would be obtained $\tilde{\alpha}$. In addition, comparing FT_K and $T_d^{\max}$, the selected strategy could be feasible if only $FT_K \leqslant T_d^{\max}$. The process is repeated L times to obtain more accurate solutions and the one with the minimum energy cost is selected. The algorithm is summarized as Algorithm 1.

4.5 EETO evaluation

4.5.1 Simulation setup

The UE and UHs are initially randomly distributed following the mobility pattern of the publicly available PECNet dataset [5]. The coverage area for the BSs and each user equipment are 400 m and 50 m, respectively. In PECNet model as a common mobility benchmark over 11,000 unique pedestrians in a university campus are considered [5]. The user's mobility speed is initially set to 1 m/s. The other relevant parameters of the simulation are listed in Table 5.1.

We benchmark EETO against four other approaches for computation placement, namely the purely UE using all local (ALO), the remote-centric all sever (ASO), and the two hybrids computation cooperation optimization (CPCO), and communication cooperation optimization (CMCO). In other words, the tasks are all executed locally on the UE in ALO; while with ASO, all tasks are executed remotely on the MEC servers. CPCO allows for the usage of adjacent devices' computation resources which allows the UH_i to act as a helper node. Finally, in CMCO, UH_i can *only* act as transmitter,

Algorithm 1: Energy-Efficient Task Offloading (EETO) Algorithm [4]

Input: L, K, I, M, $T_d^{\max}$, B, σ^2, $P(k)$, b_k, c_k, H_k, P_k^{tr}, P_k^{wait}, f_k^{UE}, $f_k^{UH_i}$, $f_k^{S_m}$, $\forall k \in \mathcal{K}$, $\mathbf{X}^{UE}$, $\mathbf{X}^{UH_i}$
Output: α
Initialize: Predict path of UE and helpers by PECNet with historical path $\mathbf{X}^{UE}$, $\mathbf{X}^{UH_i}$, calculate $T_{s,k}$ with Equation (4.25). Initialize the matrices in Equation eq(4.32). Solve the SDP problem in Equation (4.32), dropping the rank-1 constraint yielding the optimal solution $\tilde{G}$. Extract the first $(1 + I + M + IM)K$ elements of the last row in $\tilde{G}$ as α';
if *rank($\tilde{G}$)==1* then
    $\alpha = \alpha'$;
else
    for l = *1:L* do
        for $k = 1 : K$ do
            $s(k) = (1 + I + M + IM)$ elements in $\alpha'(l)$ related to task k;
            Perform probability based stochastic mapping: set one element of $s(k)$ to 1 and others to 0;
            Compute the current FT_k by $\alpha'(l)$ and Equation (4.23);
            if $FT_k > T_{s,k}$ then
                Discard $\alpha'(l)$;
        if $FT_K > T_d^{\max}$ then
            Discard $\alpha'(l)$;
        else
            $\alpha'(l)$ saved as a feasible solution and calculate total energy consumption by $\alpha'(l)$ and Equation (4.21);
    Select the solution $\tilde{\alpha}$ among $\alpha'(1),\ldots,\alpha'(L)$ that yields the minimum energy cost;
    $\alpha = \tilde{\alpha}$.

forwarding the tasks for execution to the MEC server.
Multiple independent simulation replications were run for each evaluation to eliminate random influence. The resulting 95% confidence intervals however spread out less than 5% from the corresponding sample means, and are omitted from the plots to avoid visual clutter.

4.5.2 Simulation results

Figure 4.6 shows the energy usage versus the computation CPU cycles needed for execution of each bit of task k, which is the same parameter sweep approach as in Chapter 3, but this time for the final five methods. As expected, the energy usage

Table 4.2: Simulation parameters [4]

Parameters	Value
I	10
M	2
K	25
b_k	200 – 400 KB
c_k	30 – 50 cycles/bit
$T_d^{\max}$	4.4 s
B	5 MHz
σ^2	10^{-9} W
P_k^{tr}	0.2 W
P_k^{wait}	0.05 W
f_k^{UE}	$0.1 \cdot 10^9$ cycles/s
$f_k^{UH_i}$	$0.5 \cdot 10^9$ cycles/s
$f_k^{S_m}$	$2 \cdot 10^9$ cycles/s
λ^{UH_i}	$0.8 \cdot 10^{-27}$ F
λ^{UE}	10^{-25} F

of ALO increases linearly with increase of the computations per bit. The energy usage of ASO is nearly constant with a slight increase, which is explainable by the nearly constant energy demand for transmitting the data to the server. When the computations per bit increases, the user's device uses additional energy while idly waiting on the server, a far lower usage than the transmission energy. As an intermediate result it can be stated that for simple tasks, local execution or nearby helper execution is an efficient approach. Consequently with few computations per bit, CPCO outperforms CMCO, because it allows offloading the task to the helper. For more involved tasks, execution at the server is more energy efficient. As expected for this situation, most of the energy is efforted for the transmission of the data to the server. With CMCO's more varied support of transmission paths, CMCO can outperform CPCO when the computations per bit are large. It can be observed from Figure 4.6 that the proposed EETO method is able to achieve the lowest average energy consumption among all approaches. EETO trades off optimally between the two approaches of computation cooperation (CPCO) and communication cooperation (CMCO), and thus it is free to achieve minimal average energy usage across the full range of task complexities, i.e., required computation cycles c_k per bit for task k. In the most assumption-free heterogeneous operating scenarios and with a wide range of task complexities c_k,

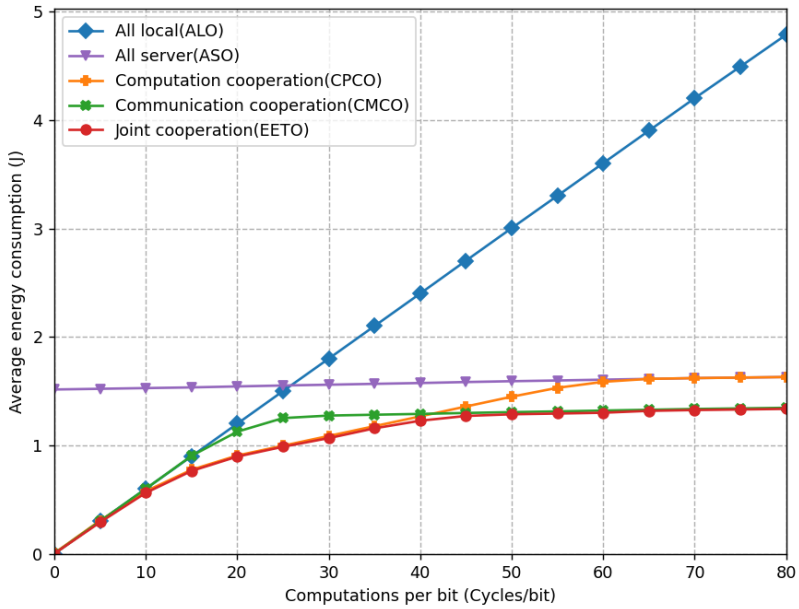


Figure 4.6: Average energy consumption vs. task computation demands c_k; fixed parameters: uniform random data size $d_k = 200$–400 KB, random task dependency, task deadline $T_d^{\max} = 4.4$ s [4].

EETO decides in an optimal way for each task k, depending on the individual per-task characteristics and thus can extract substantial energy usage reductions compared to the CPCO or CMCO benchmarks, which are only free to exploit one single type of cooperation each.

In case of required CPU cycles c_k per bit of less than 10 cycles/bit, ALO achieves minimum energy consumption among all methods, in line with previous observations and intuition gained above. When this parameter c_k of required computations increases, ALO can still satisfy the deadline requirements, but the energy usage increases over the competing algorithms. As is shown in Table 4.3, with computation-intensive workloads corresponding to $c_k \geqslant 20$ cycles/bit, ALO fails to finish the task execution within the deadline. This failure clearly demonstrates the benefits of (energy-aware) offloading methods. In Figure 4.7, we validate that increasing task size demands more energy to offload the task to the helper or server, to the point where additional computational resources are necessary for execution. This results in increased average energy usage, which of course is invariant to the chosen approach. We note how the newly introduced method achieves best in class results compared to the other methods, irrespective of data size.

Table 4.3: Finish time of local computing (ALO); Deadline: $T_d^{\max} = 4.4$ s [4]

Computations per bit c_k	Finish time of the last task [s]
0	0
5	1.2155
10	2.4201
15	3.6314
20	4.8343
25	6.0631
30	7.2743
35	8.4917
40	9.6840

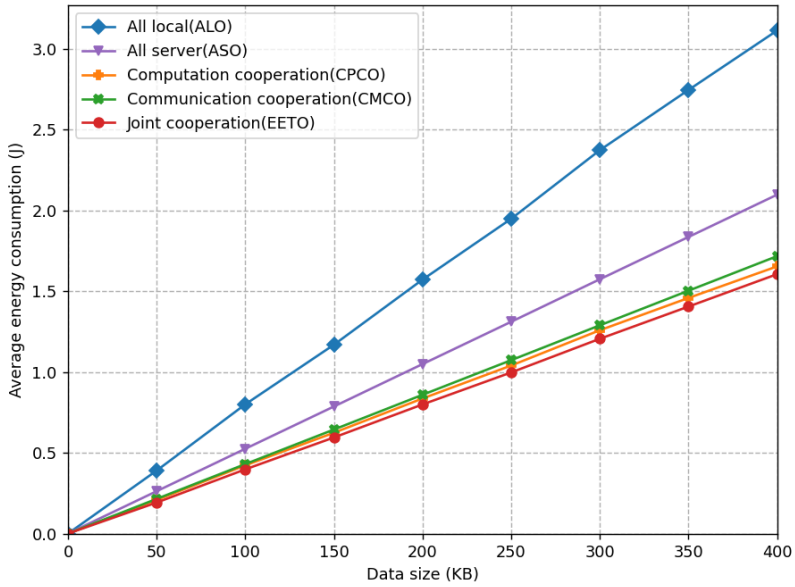


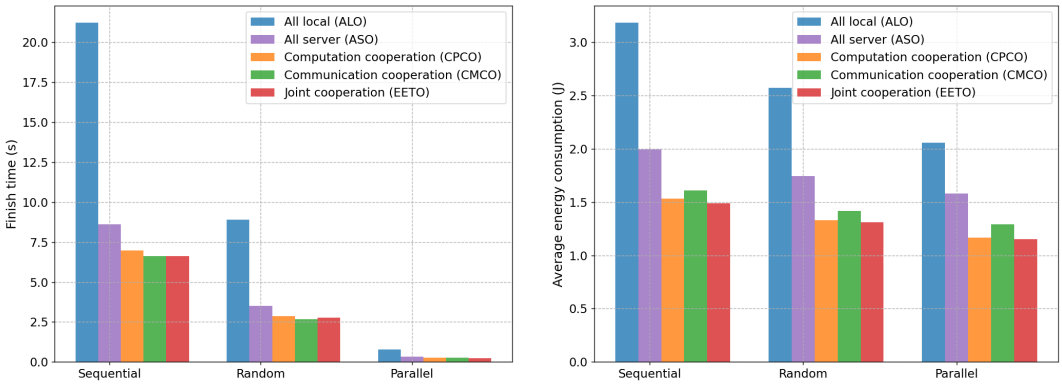
Figure 4.7: Average energy consumption vs data size d_k; fixed parameters: uniform random computation cycles per bit c_k in the range 30–50 cycles per bit, random task dependency, task deadline $T_d^{\max} = 4.4$ [4].

Similar behaviour can be observed for computation intensive tasks, and Table 4.4 shows that with increasing task data size, the ALO mode cannot satisfy the latency requirements. To enable applications with these requirements, the need for an energy-aware offloading method is demonstrated herein again. Finish time and average energy usage of the five analyzed algorithms are shown in Figures 4.8a and 4.8b for

Table 4.4: Finish time for local computing (ALO); Red numbers violate the deadline of $T_d^{\max} = 4.4$ s [4]

Data size (KB)	Finish time of final task (s)
0	0
50	1.6038
100	3.248
150	5.1438
200	6.1856
250	8.256
300	9.9504
350	11.5976
400	12.9008

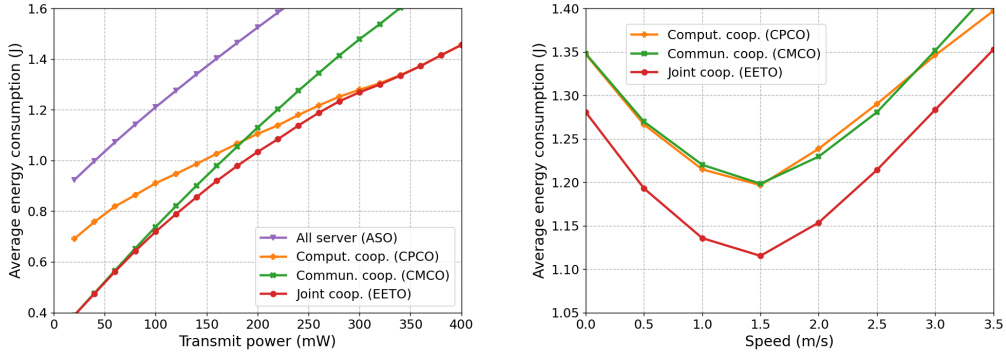
the task dependency relationship graphs corresponding to sequential, random, and parallel dependencies. The proposed EETO algorithm demonstrated the lowest energy consumption in the comparison. With a sequential dependency graphs, each task has exactly one predecessor, whose execution is required to have completed in order to start task execution. In contrast, with random and/or parallel dependency graphs, each task can have zero, one, or multiple predecessors, which requires vastly different offloading decisions to be made when compared to sequential dependency graphs. Since multiple tasks can be executed at the same time in random and parallel graphs, lowered total execution time should be expected to be achievable for all methods for the same number of total tasks. EETO does not always feature the shortest execution time, perhaps counter intuitively. Its goal is to minimize the energy usage while keeping the final finish time below the deadline. In any case, EETO's execution time is very close to the minimum. Figure 4.9a shows the average energy consumption as a function of the transmit power P_k^{tr}. As it can be observed, increasing P_k^{tr} leads to an overall growth of the average energy consumption. Due to the energy-efficient relay task transmission to the MEC servers for low $P_k^{tr} = 45$ mW, EETO and CMCO consume only about two-thirds of the battery of CPCO and less than half of ASO, however, for a high transmission power, EETO and CPCO achieve higher energy savings compared to ASO and CMCO by avoiding the energy-expensive direct transmission to an MEC server and the relay transmission.



(a) Finish time vs. dependency graphs. (b) Average energy vs. dependency graphs.

Figure 4.8: Impact of task dependency graph; fixed parameters: uniform random data size $d_k = 280$–320 KB, $c_k = 30$–50 computation cycles per bit (uniform random), task deadline $T_d^{\max} = 9$ s [4].

Figure 4.9b shows the average energy usage as a function of the user's speed. The UH_i's default speeds are considered 0.7 m/s [5]. As it can be seen in the beginning, energy consumption is decreased as the user's speed increases to 1.5 m/s and then it increases as the UE speed increases since a similar moving speed as the surrounding UH_i leads to a longer sojourn time. Importantly, Figure 4.9b shows that EETO consistently reduces the battery energy consumption compared to CPCO and CMCO across the full range of considered user's speeds.



(a) (b)

Figure 4.9: Average energy consumption vs. transmit power and user's speed). (**a**) Transmit power P_k^{tr}, (**b**) UE speed [4].

As such, the average energy usage levels versus different latency requirements

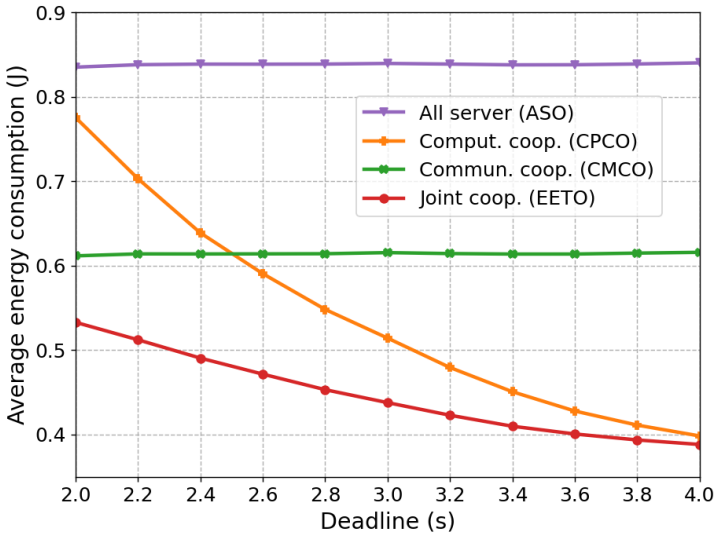


Figure 4.10: Average energy consumption vs execution deadline $T_d^{\max}$; fixed parameters: uniform random data size d_k =150–180 KB; random task dependency [4].

$T_d^{\max}$ are shown in Figure 4.10. With relaxation of the deadline of the final task, energy-efficient helpers are free to take some of the load to execute tasks even with a long processing time. Consequently, more tasks are offloaded to energy-efficient helpers, which again reduces the energy effort of CPCO and EETO. For ASO and CMCO, this relaxation of constraints introduces no exploitable change, since all the tasks are executed on the cloud, and the energy usage remains essentially constant.

4.6 Conclusion

We investigated the optimization of joint computation and communication resource allocation for computation offloading decision in a dynamic device-enhanced MEC system, considering the user mobility, the task dependencies and application deadline with the goal of improving the users' battery life time. We employed a DL based method to predict the trajectory of the users and the mobility impact has been characterized through the concept of sojourn time. Simulation results demonstrated that our proposed algorithm does achieve favorable performance when compared to other algorithms. Indeed, our system always finds a strategy to minimize energy

usage, in most cases outperforming the naive approach of selecting for every parameter configuration the offloading decision that gives the minimum energy. It can save 56.34% of energy when compared with on device execution of the tasks and 33.73% when compared with an all server execution strategy. An important direction for future work is to investigate incentive mechanisms that promote fair cooperation between users.

5 Incentive-aware Cooperative Computation Offloading

5.1 Introduction

In Chapter 3, we first discussed a basic static scenario with three nodes and in Chapter 4, we extended our scenario to a dynamic one with user mobility and handover between small cells. However, the fact that users should be motivated to share their computation and communication resources has been neglected. Therefore, in this chapter, we propose an incentive-aware dynamic computation offloading in a dense small cell which is more practical in real world scenarios. Simulation results show that our proposed method can achieve superior performance compared to the other state-of-the-art algorithms.

In order to solve such a mixed integer non linear algorithm, we apply a genetic algorithm [154] to minimize the energy consumption required for execution of the tasks while satisfying the execution time deadline of users' applications.

This chapter is organized as follows. First, the system model and network conditions are introduced in Section 5.2. Then, the optimization problem is formulated in Section 5.3. The genetic algorithm method for our offloading decision algorithm is explained in detail in Section 5.4. In Section 5.5, numerical results are presented and finally, conclusions and future works are stated in Section 5.6.

5.2 System model

In our system, as it is shown in Figure 5.1, we assume there to be N UEs represented by the set of $\mathcal{N} = \{1, 2, ..., N\}$. Each user has to perform an application which is divided into K fine-grained inter-dependent tasks. Same as in previous chapters, it is assumed that UEs are randomly distributed in a small cell with a MEC server attached to the BS, so we do not set up a favouring scenario. The BS is in charge of the D2D and cellular links control as well as offloading decision making algorithm. In this work, OFDMA is used to address concurrent transmissions of users to the BS and D2D links are dedicated and there is no cellular frequency reuse.
A device can act as a helper or relay for UE n, only if it is in its coverage radius, meaning that the distance $d(\cdot, \cdot)$ between two users is upper-bounded by the threshhold R_i. Therefore, a set of $N - n$ resource devices of

$$\mathcal{H} = \bigcup_{n \in \mathcal{N}} \{i \in \mathcal{N} : d(n, i) \leqslant R_i\}$$

can be defined such that $\mathcal{H} \subset \mathcal{N}$. In essence, the D2D link between two UEs can be established if the distance between them is below the threshold R_i.
The wireless channel is assumed quasi-static during the execution time and the CSI and computation related parameters are available to the devices. The computation result of each task is assumed much smaller than the input bits, therefore, the time for sending the result back is negligible and thus, only the transmission time of the task input and the execution time are taken into consideration. There is a time deadline for completing the UEs' tasks and it is assumed that the data size and computation resource requirements for each task are known in advance. In the following, we elaborate the system model parameters.

5.2.1 Task model

Same as Section 4, the tasks follow general dependency using a directed acyclic graph which is represented by $G = (V, E)$. The set V includes the tasks and $e(i, j)$ indicates the relationship between tasks i and j, meaning that if task i is the predecessor of task j, the edge $e(i, j)$ exits in E [131].
Each task k from device n has a set of parameters $A_{n,k} = \{b_{n,k}, c_{n,k}, d_{n,k}\}$, where $b_{n,k}$

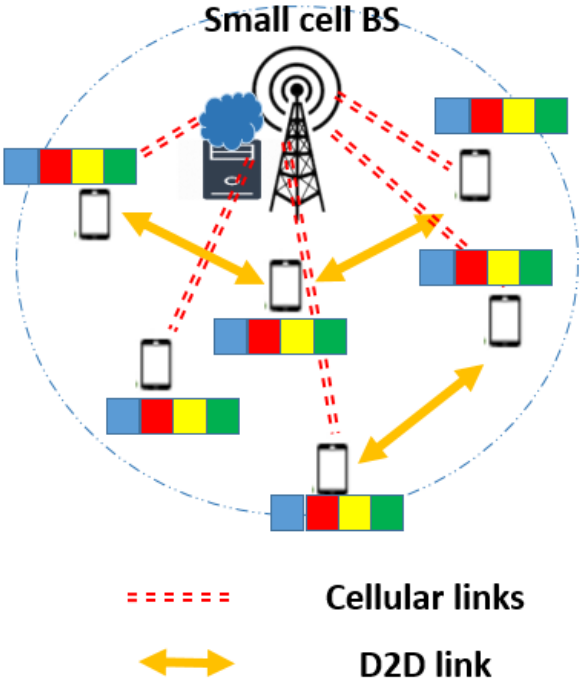


Figure 5.1: The system model containing one small cell and multiple users with multiple tasks.

is the data size of computation task k (in bits) of device n, $c_{n,k}$ is the computation resources which are required for execution of each bit of task k of device n (in CPU cycles/bit), and $d_{n,k}$, equals to $b_{n,k} \times c_{n,k}$, is the total amount of required computation resources for execution of the task k and T_n^{max} is the time deadline for the execution process of the whole application of device n.

5.2.2 Communication model

The up-link data rate for task k of UE n can be calculated using the Shannon theorem and the down-link data rate is neglected in our scenario due to the small size of the executed tasks.

$$r_{n,k} = B \log_2 \left(1 + \frac{P_k^{tr} H_k}{\Gamma \sigma^2}\right), \tag{5.1}$$

Here, P_k^{tr} is the sender's transmission power, B and H are the channel bandwidth and channel gain between sender and receiver, respectively, and σ^2 denotes the variance of the Gaussian channel noise. Γ is the coding gap as a function of bit error rate which is determined based on the coding schemes and medium access protocol [131]. To simplify the model, here we assume $\Gamma = 1$.

5.2.3 Mobility model

The mobility model in this section is same as Chapter 4. ML is employed to predict the users' paths and obtain their available service coverage time. The DL based method PECNet [5] is employed to predict socially compliant trajectories which conclude from users' destinations to help prediction.
In our scenario, the position of the UE n and the helper node i at time t are defined as $X_t^n = (x_t^n, y_t^n)$ and $X_t^i = \left(x_t^i, y_t^i\right)$, respectively. The UE n and its corresponding helpers' paths from $(t - n + 1)$ to t are:

$$\mathbf{X}^n = \{X_{t-n+1}^n, X_{t-n+2}^n, \ldots, X_t^n\} \tag{5.2}$$

$$\mathbf{X}^i = \left\{X_{t-n+1}^i, X_{t-n+2}^i, \ldots, X_t^i\right\}. \tag{5.3}$$

The edge server is responsible to collect this information. The trajectories of UE n and helpers are PECNet inputs and outputs are the predicted movements from time $(t + 1)$ to $(t + m)$:

$$\widehat{\mathbf{Y}}^n = \left\{\widehat{Y}_{t+1}^n, \widehat{Y}_{t+2}^n, \ldots, \widehat{Y}_{t+m}^n\right\} \tag{5.4}$$

$$\widehat{\mathbf{Y}}^i = \left\{\widehat{Y}_{t+1}^i, \widehat{Y}_{t+2}^i, \ldots, \widehat{Y}_{t+m}^i\right\}, \tag{5.5}$$

where $\widehat{Y}_t^n = (\hat{x}_t^n, \hat{y}_t^n)$ and $\widehat{Y}_t^i = \left(\hat{x}_t^i, \hat{y}_t^i\right)$. For establishing a D2D link between the UE n and the helper i at time t, the UE n should be in the coverage area of helper i with radius R_i,

$$\sqrt{\left(\hat{x}_t^n - \hat{x}_t^i\right)^2 + \left(\hat{y}_t^n - \hat{y}_t^i\right)^2} \leqslant R_i. \tag{5.6}$$

The sojourn time of UE in helper i's coverage from time t is then given as:

$$T_{s,i,t} = \max\ m,\ \text{s.t.}\ \sqrt{(\hat{x}_\tau^n - \hat{x}_\tau^i)^2 + (\hat{y}_\tau^n - \hat{y}_\tau^i)^2} \leqslant R_i,\ \ \forall \tau \in [t, t+m] \subset \mathbb{Z}. \tag{5.7}$$

5.2.4 Computation model

There are four possible scenarios in our task offloading algorithm: local execution on the UEs, remote execution on the helpers, remote execution on the MEC server via a direct offloading from UE or via the relay to the MEC server.
Depending on that which device or server is executing the task, the time and energy needed for execution of task k is calculated based on the computation capability of the host. This can be calculated based on the CPU cycles needed for execution of tasks which are known in advance in our scenario.

5.2.4.1 Local computing

The time needed for task k execution locally (on the UE n) is then obtained as follows:

$$T^l_{n,k} = \frac{d_{n,k}}{f^l_{n,k}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \tag{5.8}$$

where $f^l_{n,k}$ is the computation capacity of UE n allocated for task k's execution and it is assumed to remain constant during processing of task k. Based on the effective switched capacitance, the energy consumption per operation is $\varepsilon = \lambda(f^l_{n,k})^2$ as a factor depending on the chip architecture denoted as λ [132]. Therefore:

$$E^l_{n,k} = \lambda \left(f^l_{n,k}\right)^2 d_{n,k}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}. \tag{5.9}$$

The equations (5.8) and (5.9) are fully descriptive due to the no transmission possibility of data in the local mode. We define the binary variable $x_{n,k}$ which when it equals to one, it means that the UE chose to execute task k locally on UE n.

5.2.4.2 Computing on the helper

The task k of device n can be offloaded to one of the helpers in set $\mathcal{H}$. Suppose UE i plays the role of helper to execute the task k of device n: the computation execution time consists of two parts, communication time and computation time. Therefore, the total time budget splits into two parts as follows:

$$T^i_{n,k} = \frac{b_{n,k}}{r^i_{n,k}} + \frac{d_{n,k}}{f^i_{n,k}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}, \tag{5.10}$$

where $r^i_{n,k}$ is the transmission data rate between UE n to UE i and $f^i_{n,k}$ denotes the CPU clock cycles per second allocated to execute task k of UE n on the helper i. The energy consumption in this mode is then calculated as follows:

$$E^i_{n,k} = P^{tr}_{n,k}\left(\frac{b_{n,k}}{r^i_{n,k}}\right) + \lambda\left(f^i_{n,k}\right)^2 d_{n,k}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}. \tag{5.11}$$

Here, $P^{tr}_{n,k}$ is the transmission power of UE n, $r^i_{n,k}$ is the transmission data rate from UE n to UE i. The UE n can chose one helper at the time to offload task k. Therefore, we define a binary variable $a^i_{n,k} = 1$, indicates that the UE n chose to offload task k to UE i, where $i \in \mathcal{H}$. Therefore, a logical offloading selection algorithm should follow the constraint

$$\sum_{i \in \mathcal{H}} a^i_{n,k} \leqslant 1. \tag{5.12}$$

5.2.4.3 Computing on the edge cloud

Under this offloading strategy mode, the task is offloaded to the MEC server, which has more powerful computation abilities and energy supply. The UE transmits the task to the MEC server directly and after remote execution, the result is sent back to UE. The energy consumption of the MEC server is neglected, since it is assumed that the MEC server is powered by the grid. There are two time delay steps for transmission and computation:

$$T^s_{n,k} = \frac{b_{n,k}}{r^s_{n,k}} + \frac{d_{n,k}}{f^s_{n,k}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}, \tag{5.13}$$

where $r^s_{n,k}$ is the transmission rate from UE to MEC server and $f^s_{n,k}$ is the computation capacity allocated for task k of device n execution on the MEC server. The corresponding energy consumption for the UE in this remote case execution is calculated as follows:

$$E^s_{n,k} = P^{tr}_{n,k}\left(\frac{b_{n,k}}{r^s_{n,k}}\right), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}. \tag{5.14}$$

We define the binary variable $y_{n,k}$ which when it equals 1, it means that the UE chose to offload task k directly to the MEC server.

5.2.4.4 Computing on the edge cloud via relay

In this mode, the task is still offloaded to the MEC server, but with the help from a relay. This time UE i receives the data from UE n, and transmits the data further to the MEC server. This can be a good strategy when the wireless channel state between UE and MEC server is far worse than the channel state between the relaying UE i and MEC server. Therefore, there are three time delay steps for transmissions and computation:

$$T^{i,s}_{n,k} = \frac{b_{n,k}}{r^{i}_{n,k}} + \frac{b_{n,k}}{r^{i,s}_{n,k}} + \frac{d_{n,k}}{f^{s}_{n,k}}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}, \tag{5.15}$$

Here, $r^{i,s}_{n,k}$ is the effective transmission rate from UE i to the MEC server (after error correction).

The corresponding energy consumption in this remote case execution can be then calculated as follows:

$$E^{i,s}_{n,k} = P^{tr}_{n,k}\left(\frac{b_{n,k}}{r^{s}_{n,k}}\right) + P^{tr,i}_{n,k}\left(\frac{b_{n,k}}{r^{s}_{n,k}}\right), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}. \tag{5.16}$$

The UE n can chose one relay at the time to offload task k. Therefore, we define a binary variable $a''^{i}_{n,k} = 1$, which indicates that the UE n chose to offload task k to relay i, where $i \in \mathcal{H}$. Therefore, a logical offloading selection algorithm will have to follow the constraint

$$\sum_{i \in \mathcal{H}} a''^{i}_{n,k} \leqslant 1. \tag{5.17}$$

5.3 Task dependency

There will be one or more predecessor tasks for any task $k \neq 1$ which without finishing execution of them, the execution process of task k cannot begin. In our offloading decision algorithm, the tasks dependency is taken into account as follows [155]:
Start time: It is the earliest time that the execution process of task k can get begin, meaning that execution of all predecessors of task k is completed. The start time of task k of UE n can be defined depending on the offloading modes.
Finish time: This is the time that the execution process for task k of UE n is finished,

$$FT_{n,k} = x_{n,k}FT^{l}_{n,k} + a'^{i}_{n,k}FT^{i}_{n,k} + a''^{i}_{n,k}FT^{i,s}_{n,k} + y_{n,k}FT^{s}_{n,k}. \tag{5.18}$$

The determination of these important times branches depending on the offloading mode:

- **Local computing**: In this mode, the biggest time value between the completion of the execution of task's predecessors and the time that UE n is available for execution process is considered as the start time.

$$ST^{l}_{n,k} = \max\left(\max\left(FT_{n,p}\right), FT^{l}_{n,b}, FT^{n}_{i,b'}\right) \tag{5.19}$$

  where p is the predecessor of task k. $FT^{l}_{n,b}$ is the finish time for completion of task b keeping the UE n occupied. If the UE n is acting as a helper at the moment for other UEs, then, we have $FT^{n}_{i,b'}$ for the execution completion process.
  The finish time can be then calculated as follows:

$$FT^{l}_{n,k} = ST^{l}_{n,k} + T^{l}_{n,k} \tag{5.20}$$

- **Computing on the helper**: Same as above, in this mode, the task k of device n has to wait for its predecessors' execution and the availability of the helper i. In case helper i is acting as a relay for the task b' of UE m, we should wait till UE m finishes offloading its task to UE i after which the D2D link is free. Note that task k can be offloaded to the helper simultaneously with offloading task b' to the edge.

$$ST^{i}_{n,k} = \max\left(\max\left(FT_{n,p}\right), FT^{l}_{i,b}, FT^{i}_{m,b'}, ST^{i,s}_{m,b'} + \frac{b_{m,b'}}{r^{i}_{m,b'}}\right) \tag{5.21}$$

  Here, $FT^{i}_{m,b'}$ and $ST^{i,s}_{m,b'}$ denote the finish time of task b' of UE m, when UE n is acting as helper and the start time of task b' of UE m when it acts as a relay for UE m, respectively. The finish time can be then calculated as:

$$FT^{i}_{n,k} = ST^{i}_{n,k} + T^{i}_{n,k} \tag{5.22}$$

- **Direct edge offloading**: In this mode, the UE n should only wait for execution of predecessor tasks.

$$ST^{s}_{n,k} = \max\left(FT_{n,p}\right) \tag{5.23}$$

$$FT^{s}_{n,k} = ST^{s}_{n,k} + T^{s}_{n,k} \tag{5.24}$$

- Edge offloading via relay: In this mode, the predecessors' execution completion as well as the availability of the relay should be considered.

$$ST^{i,s}_{n,k} = \max(\max(FT_{n,p}), FT^{l}_{i,b}, FT^{i}_{m,b'}, ST^{i,s}_{m,b'} + \frac{b_{m,b'}}{r^{i}_{m,b'}}) \tag{5.25}$$

$$FT^{i,s}_{n,k} = ST^{i,s}_{n,k} + T^{i,s}_{n,k} \tag{5.26}$$

5.4 Incentive mechanism

In a feasible cooperative computation offloading algorithm, in order to prevent selfish behaviors of UEs in computation and communication resources usage, a proper incentive mechanism should be defined. By getting inspiration from [12] and using resource tit-for-tat and energy budget constraints, we introduce our incentivising method. In this thesis, we have two types of resource constraint, namely, computing cycles and cellular bandwidth, which account for computation and communication resources, respectively.
Therefore, in order to define the amount of resources that UE n takes from other devices, we introduce $X_n \triangleq \{X^c_n, X^b_n\}$, where X^c_n and x^b_n are the computing and bandwidth resources, respectively. The same goes for $Y_n \triangleq \{Y^c_n, Y^b_n\}$, where, Y^c_n and Y^b_n are the computing and bandwidth resources that UE n contributes to other devices, respectively. Considering each task k of device n, we have the following expressions:

$$X^c_n = \sum_{i \neq n} \sum_{k=1}^{K} a^{i}_{n,k} d_{n,k}, \qquad Y^c_n = \sum_{i \neq n} \sum_{k=1}^{K} a^{i}_{i,k} d_{i,k} \tag{5.27}$$

$$X^b_n = \sum_{i \neq n} \sum_{k=1}^{K} a'^{i}_{n,k} b_{n,k}, \qquad Y^b_n = \sum_{i \neq m} \sum_{k=1}^{K} a'^{i}_{i,k} b_{i,k} \tag{5.28}$$

We have then the average of $\overline{X_n} = \left\{\overline{X^c_n}, \overline{X^b_n}\right\}$ and $\overline{Y_n} = \left\{\overline{Y^c_n}, \overline{Y^b_n}\right\}$ for each device n (e.g., $\overline{X^c_n} = \frac{X^c_n}{K}$, K is the total number of tasks). By defining a resource tit-for-tat constraint [12]:

$$\alpha^c_n \overline{X^c_n} \leqslant \beta^c_n + \overline{Y^c_n} \tag{5.29}$$

$$\alpha_n^b \overline{X_n^b} \leqslant \beta_n^b + \overline{Y_n^b} \tag{5.30}$$

where each α_n and β_n are within [0,1]. The tit-for-tat resource constraints assure a fair resource amount exchange meaning that the more resources a device gets from others, the more it has to share in return [12].

It may happen that there are some UEs with sufficient resources which are mostly idle, therefore, the BS may repeatedly chooses them in order to share their resources by other UEs with computation or communication resource demands. Thus these devices would suffer from over-exploitation and undesired battery life time degradation. To prevent this, here we introduce an energy budget constraint which will be determined by the network operator [12]. This energy budget represented by E_i^{budget} is the average energy budget for helper node $UE_i \in \mathcal{H}$. The total energy which UE_i consumes to execute the offloaded tasks is obtained as

$$E_i = \sum_{n \neq i} \sum_{k=1}^{K} \left(a_{n,k}^i \lambda \left(f_{n,k}^i \right)^2 d_{n,k} + a_{n,k}'^i P_{n,k}^{tr,i} T_{n,k}^{i,s} \right),$$

therefore, the energy budget constraint can be as follows:

$$\overline{E_i} \leqslant E_i^{budget} \tag{5.31}$$

5.5 Problem formulation

The energy consumption for execution of the task k of UE n can be calculated based on equations (5.11), (5.9), (5.14) and (5.16) as follows

$$E_{n,k} = x_{n,k} E_{n,k}^l + a_{n,k}^i E_{n,k}^i + a_{n,k}'^i E_{n,k}^{i,s} + y_{n,k} E_{n,k}^s. \tag{5.32}$$

Our computation offloading decision algorithm's goal is to minimize the sum of energy consumption given in equation (5.32) and the finish time of the tasks. Therefore, we introduce the variable $Z_{n,k}$ as the weighted combination of the energy consumption and finish time for executing the task k of UE n, that contributes to our composite cost function, where ω_k^T and ω_k^E are the weight factors satisfying the constraint $\omega_k^T + \omega_k^E = 1$:

$$Z_{n,k} = \omega^T FT_{n,k} + \omega^E E_{n,k}.$$

Consequently, this variable can be differentiated along the contributing elements for the various offloading modes, i.e.

$$Z_{n,k} = x_{n,k} Z^{l}_{n,k} + a^{i}_{n,k} Z^{i}_{n,k} + a'^{i}_{n,k} Z^{i,s}_{n,k} + y_{n,k} Z^{s}_{n,k}.$$

Based on the system model and constraints described above, we formulate our computation offloading decision algorithm as follows:

$$\begin{aligned}
\min_{q} \quad & \sum_{n=1}^{N} \sum_{k=1}^{K} Z_{n,k} \\
s.t. \quad & \forall k \in \mathcal{K}, i \in \mathcal{N}, n \in \mathcal{N} \\
& C1: \quad x_{n,k}, a^{i}_{n,k}, a'^{i}_{n,k}, y_{n,k} \in \{0,1\} \\
& C2: \quad x_{n,k} + \sum_{i=1}^{I} \left(a^{i}_{n,k} + a'^{i}_{n,k} \right) + y_{n,k} = 1 \\
& C3: \quad x_{n,k} + \sum_{n \in \mathcal{N}-i} \left(a^{n}_{i,k} + a'^{n}_{i,k} \right) \leqslant 1 \\
& C4: \quad \sum_{n=1}^{N} \sum_{k=1}^{K} f^{s}_{n,k} \leqslant F_{max}, \quad if \quad y_{n,k} = 1 \\
& C5: \quad \max\left\{ FT_{n,p} \right\} \leqslant ST_{n,k} \\
& C6: \quad FT_{n,K} \leqslant T^{max}_{n} \\
& C7: \quad FT_{n,k} \leqslant T_{i,s}, \quad if \;\; i \neq n \\
& C8: \quad a_n \overline{X_n} \leqslant \beta_n + \overline{Y_n} \\
& C9: \quad \overline{E_i} \leqslant E^{budget}_{i},
\end{aligned} \tag{5.33}$$

(5.34)

where q is the offloading binary decision variable as follows:

$$\begin{aligned}
\boldsymbol{q} = [& x_{1,1}, a^{1}_{1,1}, a^{2}_{1,1}, ..., a^{I}_{1,1}, a'^{1}_{1,1}, a^{2'}_{1,1}, ..., a'^{I}_{1,1}, y_{1,1}, \; x_{1,2}, a^{1}_{1,2}, ..., a^{I}_{1,2}, a'^{1}_{1,2}, ..., a'^{I}_{1,2}, \\
& y_{1,2}, ..., \; x_{1,K}, a^{1}_{1,K}, ..., a^{I}_{1,K}, a'^{1}_{1,K}, ..., a'^{I}_{1,K}, y_{1,K}, \; x_{2,1}, a^{1}_{2,1}, ..., a^{I}_{2,1}, a'^{1}_{2,1}, ..., a'^{I}_{2,1}, \\
& y_{2,1}, ..., \; x_{N,K}, a^{1}_{N,K}, ..., a^{I}_{N,K}, a'^{1}_{N,K}, ..., a'^{I}_{N,K}, y_{N,K}]^{T}.
\end{aligned} \tag{5.35}$$

The constraints C1 and C2 represent the offloading decision binary variables, meaning only one of them can be 1 and the rest are 0 for each task k of device n. The third constraint depicts that UE n can only process one task (as a helper or relay) at a

time. Constraint C4 assures that the sum of dedicated CPU cycle frequency to each task at the edge is less or equal than the total computation capacity of the server. C5 represents the task dependency relationship constraint. C6 denotes the finish time of the last task of UEs, which is equal to the completion time of the application, which must be within the deadline. C7 represents that an offloading destination may only be chosen if the sojourn time of the UE in the range of the computation resource is longer than the time needed to finish executing task k. And finally, the C8 and C9 are the incentive constraints which are described in Section 5.4.

In order to homogenize the different symbols in the decision variable vector $\boldsymbol{q}$, the local execution binary variable, $x_{n,k}$ can be written as $a^n_{n,k}$ and server execution binary variable can be also rewritten as $a'^n_{n,k}$. Therefore, $a^i_{n,k}$ can indicate local execution and $a'^i_{n,k}$ can represent direct server offloading (only for $i = n$) (here $i \in \{1, 2, ..., N\}$).
The offloading decision variables vector can be then written as follows:

$$\begin{aligned}\boldsymbol{q} = [&a^1_{1,1}, a^2_{1,1}, ..., a^N_{1,1}, a'^1_{1,1}, a'^2_{1,1}, ..., a'^N_{1,1}, a^1_{1,2}, a^2_{1,2}, ..., a^N_{1,2}, a'^1_{1,2}, a'^2_{1,2}, ..., a'^N_{1,2}, ...,\\ &a^1_{1,K}, a^2_{1,K}, ..., a^N_{1,K}, a'^1_{1,K}, a'^2_{1,K}, ..., a'^N_{1,K}, a^1_{2,1}, ..., a^N_{2,1}, a'^1_{2,1}, ..., a'^N_{2,1}, ..., a^1_{N,K},\\ &..., a^N_{N,K}, a'^1_{N,K}, ..., a'^N_{N,K}]^T\end{aligned} \tag{5.36}$$

5.6 Solution of the incentive- and mobility-aware computation offloading optimization problem (IMCO)

A dynamic edge computation offloading method with joint communication and computation cooperation is proposed in this section. To solve our formulated MINLP minimization problem, we propose a Genetic Algorithm (GA) as a metaheuristic evolutionary algorithm which is proper for big search spaces. GAs have been proved to produce better results by operating on a population of solutions rather than a single solution [156, 157]. First, an initial population as solutions are produced and then, by a selection process, two chromosomes are chosen as parents to produce next generation. By crossover and mutation of generations, after some iterations, a near-optimal solution would be obtained by convergence of the chromosomes [157]. We describe the aforementioned terms of a GA in the following subsections in detail.

5.6.1 Chromosomes and encoding method

In GAs, a chromosome contains a set of genes. A gene for a computation offloading problem could be the offloading decision variables for computation task k. Therefore, one chromosome can be the offloading decision variables for the total applications of UEs as it has been shown in Figure 5.2.

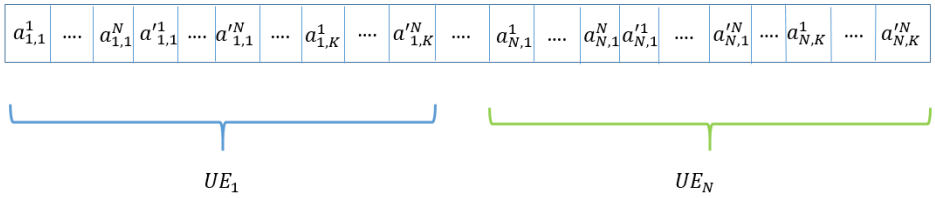


Figure 5.2: An example of a chromosome with offloading variables.

Assuming a number N of UEs in the network with the number K of tasks, each chromosome has $2KN^2$ binary genes or decision variables which slows down our algorithm by increasing the complexity in each iteration and this problem gets worse with increasing number of UEs and computation tasks. Therefore, instead of using binary variables, we use the location of the device or the MEC server where the task is executed on as integer values between 0 and $2N-1$. The integer numbers between $[0, N-1]$ represent the local or helper execution where as the numbers from N to $2N-1$ indicate direct server execution or server offloading via relay options. Since $n \in \{0, 1, 2, ..., N-1\}$ and $k \in \{0, 1, 2, ..., K-1\}$, the size of each chromosome becomes KN.

5.6.2 Fitness function

Our computation offloading algorithm goal is minimizing the total sum of energy consumption given in equation (5.32) and the finish time of the tasks while satisfying the mobility, incentive and applications deadline constraints. Therefore, similar to works in [157–159], our fitness function can be defined as sum of our objective function and the constraints as penalties such that there will be punishments for

each chromosome violating the constraints. The fitness function can be written as:

$$\begin{aligned} Fitness = \sum_{n=0}^{N-1}\sum_{k=0}^{K-1} Z_{n,k} + \theta\Big[& \max\{0, FT_{n,K} - T_n^{max}\} + \max\{0, \left(FT_{n,k}^{i} - T_{i,s}\right)\} \\ & + max\{0, (FT_{n,k}^{i,s} - T_{i,s})\} + \max\{0, \alpha_n\overline{X_n} - \beta_n - \overline{Y_n}\} + \max\{0, \\ & \overline{E_i} - E_i^{budget}\}\Big] \end{aligned} \tag{5.37}$$

Here, θ is a very large number. Since constraints C1 and C2 are met in our chromosomes definition, there is no more need to include them in the penalty function. This is same for C3 where we used same process as [157] which the computation resource allocation constraint is simplified for remote execution by applying uniform frequency assignment.
The constraint C4 in our method is calculated based on the offloading options as described in equations (5.19), (5.21), (5.23), (5.25).

5.6.3 Individual selection

The process selects individuals (choromosomes) at random and pits them in a tournament. The individual with the best fitness values, which has won each tournament, will be selected as parents for the crossover. The advantage of tournament method lies in its lower computational complexity, while keeping the diversity in solutions [157].

5.6.4 Crossover and mutation

The crossover and mutation operations are executed in every iteration of the GA to prevent the convergence towards locally optimal, but globally suboptimal solutions. The crossover operation works as follows: Two off-springs called $o1$ and $o2$ are obtained by applying the following operations on the parents $p1$ and $p2$:

1. First, we initialize $g = 0$, where g represents a gene and $g \in \{0,1, 2, ..., KN\text{-}1\})$, $o1 = p1$ and $o2 = p2$

2. For every g^{th} gene of offspring chromosome, a float number r is selected between [0, 1] randomly.

3. If $r < 0.5$ then the g^{th} gene of the offspring chromosomes is replaced by the g^{th} gene of corresponding parents as follows:

$$o1_g = p2_g$$

$$o2_g = p1_g$$

4. We increase g by 1 and repeat the process until $g < KN - 1$.

In order to mutate, the worst chromosome is chosen and is replaced randomly with new parameters [160]. The GA typically continues until it reaches a predetermined number of iterations. A brief summary of our joint computation and communication cooperation offloading algorithm is shown in Algorithm 2.

Algorithm 2: Incentive- and Mobility-aware Computation Offloading (IMCO) Algorithm

Input: L, K, N, $T_n^{\max}$, B, σ^2, b_k, d_k, $H_{n,i}$, $P_{n,k}^{tra}$, $f_{n,k}^{l}$, $f_{n,k}^{i}$, $f_{n,k}^{s}$, $\mathbf{X}^n$, $\mathbf{X}^i$
Output: q
Initialize: Predict path of UE n and helpers by PECNet with historical path $\mathbf{X}^n$, $\mathbf{X}^i$, calculate $T_{i,s}$;
Encode q into a chromosome;
Create the initial population randomly;
obtain the fitness values for each of chromosomes in the population;
while $L \leqslant Iteration_{max}$ **do**
    Select parents using tournament method;
    Do crossover operation with probability φ;
    Do mutation operation with probability δ;
    endwhile
end

5.7 Numerical results

In this section, simulation results are provided to validate the performance of our joint communication and computation cooperation dynamic computation offloading algorithm (IMCO) compared to the following approaches:

- **All local:** The tasks are fully executed on UEs.
- **All server:** The tasks are fully offloaded to the MEC server.

- **Computation cooperation:** The users can only act as helper nodes to execute the computation tasks.
- **Communication cooperation:** The users can only act as relay nodes to transmit the tasks to the MEC server.

The simulation parameters are listed in Table 5.1. The average cost function for

Table 5.1: Simulation parameters

Parameters	Value
Number of tasks of each user (K)	10
Number of users in the network (N)	5
Task deadline (T_n^{max})	4.5 s
Data size of task k (b_k)	200 – 500 kb
Required CPU cycles per bit of task k (c_k)	110 cycles/bit
Channel bandwidth (B)	5 MHz
Channel gain between the UE_n and UE_i	$1 - 1.5 \times 10^{-2}$
Channel gain between the UE_n and MEC	$1 - 1.5 \times 10^{-5}$
Variance of the Gaussian channel noise (σ^2)	10^{-9}
Transmission power of UEs ($P^{tr}_{n,k}$)	0.1 – 0.15 mW
CPU cycles frequency of UEs($f^{l}_{n,k}$)	$4 - 7 \times 10^8$ cycles/s
Maximum CPU cycles frequency of MEC server (f_S^{max})	10×10^9 cycles/s
Effective switched capacitance (λ)	10^{-27} F
Area considered for UE mobility	500*500 m^2
Energy budget(E_i^{budget})	0.25
Energy weight(ω^E)	0.5
Time weight(ω^T)	0.5
D2D range	100 m
Number of GA iterations	150
Population size	30
Crossover probability φ	0.5
Mutation probability δ	0.05

50 runs is shown in Figure 5.3. As can be observed, when the required computations per bit of the task k increases, the average energy consumption of all methods increases as well due to the more computation power required to complete the execution process. As the CPU resource requirements of tasks become larger, the performance benefit of our joint communication and computation cooperation approach is more observable.

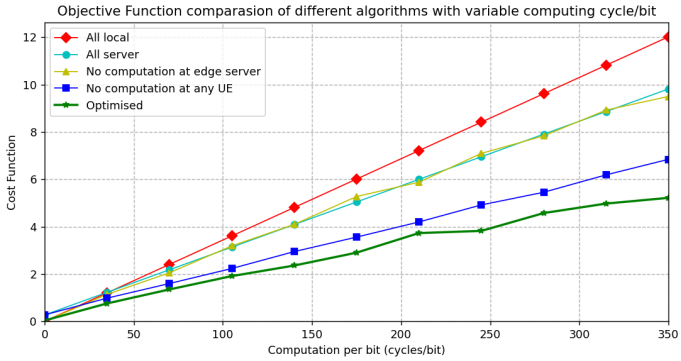


Figure 5.3: Average energy consumption vs. tasks' computation demands.

The average energy consumption for different methods, with data size changing from 0 to 800 kb, is shown in Figure 5.4. By increasing the size of the task, more computation resources are needed to finish the execution process. This leads to the increases of the average energy consumption of all methods. The energy consumption of our joint communication and computation cooperation is still lower compared to the other methods. Figure 5.5 shows the GA convergence speed of our method for

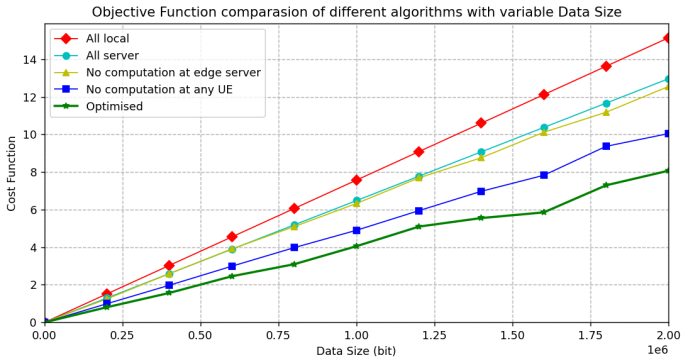


Figure 5.4: Average energy consumption vs. data size.

different number of tasks. As it can be seen, the algorithm for 5 and 10 number of total tasks converges to the minimum amount after approximately 15 and 20 iteration steps, respectively. This shows the effectiveness of our GA and its fast speed for obtaining the results. However, for higher number of total tasks the more iterations are required.

Figure 5.6 shows the cost function for our method with and without cooperation considering task complexity increase. The non-cooperation method does not in-

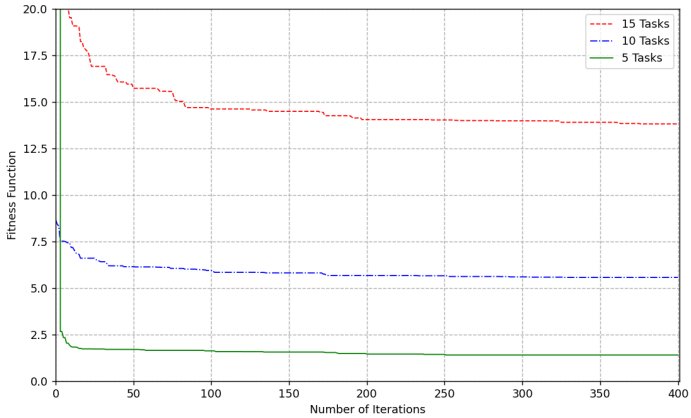


Figure 5.5: Fitness function vs. number of iterations.

clude any resource sharing by helpers and relays and therefore there is no incentivising involved. As can be seen, the optimized method performs better than the non-cooperation for any CPU cycles required for task execution and by increasing the complexity the performance is getting better compared to non-cooperation algorithm which shows the need for our optimized method for having more suitable offloading decision algorithm under worse task situations.

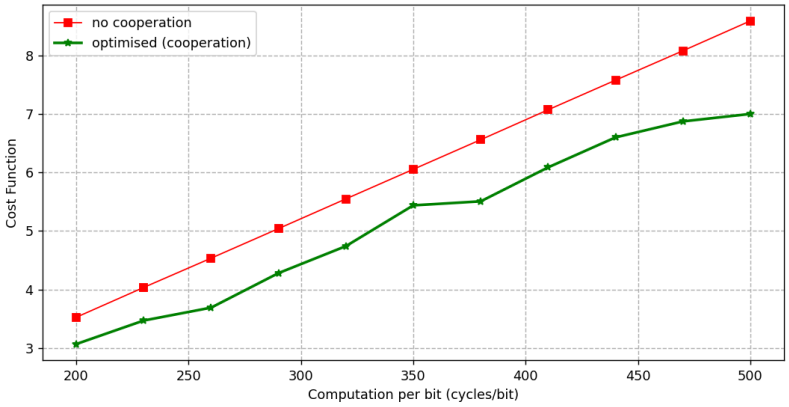


Figure 5.6: Cost function vs. task complexity.

5.8 Conclusion

In this chapter, we studied the optimization of computation offloading decision algorithm in a multi-user multi-task MEC system considering the tasks' dependency and time deadline required to finish the application using GA. The final goal was minimizing the users' battery life time while satisfying the latency, mobility and cooperation requirements. Simulation results show that our method can achieve the best performance by jointly optimizing the computation and communication resource allocations. For future work, the scenario may be made even more realistic by considering the frequency reuse for more efficient bandwidth usage.

6 Conclusion and Future Research Directions

Part of the content of this chapter was previously published in:
Mehrabi, Mahshid, et al. "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey." *IEEE Access 7* (2019): 166079-166108 [2].

This chapter outlines the thesis contributions as well as the major open issues and goes on to outline possible future research directions to make device-enhanced MEC computation offloading a more attractive, efficient, and effective tool. In Section 7.1, the summaries of the contributions in this thesis are presented. In Section 7.2, the future research directions are stated and finally, in Section 7.3 a list of the publications resulted from this thesis are mentioned.

6.1 Summary of contributions

In this thesis, we addressed the issue of multi-user multi-task cooperative computation offloading in a device-enhanced MEC system. We considered that an application such as the distributed Ibis application presented in [9] or a video navigation application running on a smartphone [149] can be divided to several separated tasks which are dependent. We then studied different strategies for each task including local execution on the device, helper execution, direct server execution and server execution

via a relay considering the application requirements, mobility and users' incentives in a device-enhanced MEC network. The offloading decision algorithm optimization problems are MINLP and NP-hard. We therefore, proposed efficient strategies to overcome the difficulty of solving this NP-hard problems [161].

In our first contribution presented in Chapter 3, we considered a basic three node device-enhanced MEC system with the main goal of minimizing the energy consumption of the user while satisfying the task dependencies and the application deadline requirements. In order to solve such a problem, we first transform our mixed integer non linear algorithm to a QCQP approach and then using SDR method, an approximation of the original problem can be obtained. Finally, a randomization method is applied to obtain optimum offloading strategy. Simulation results show that the proposed algorithm can achieve superior performance compared to the other state-of-the-art algorithms. It can save up to 65.47% of energy when compared with on device execution of the tasks and 49.29% when compared with an all server execution strategy for complex tasks scenarios.

In our second contribution presented in Chapter 4, we investigated the problems introduced by mobility to the D2D links and offloading process breakage as well as the handover and service migration further delay and energy cost in a multi-user multi-cell scenario. The mobility effects of users in this chapter captured by sojourn time concept. It was obtained by a DL algorithm called PECNet which uses the historical data of motion paths. Minimizing the total energy consumption for all users in the system while satisfying the time deadline constraints of the tasks was the main goal of this work. The same solving method was used in this chapter to capture the complexity involved in optimization problem and simulation results show the superior performance of our method compared to other algorithms. It can save up to 56.34% of energy when compared with on device execution of the tasks and 33.73% when compared with an all server execution strategy for complex tasks scenarios.

The fact that users should be pursued to share their computation and communication resources was captured in Chapter 5 by extending our scenario to the multi-user multi-task dynamic scenario. Considering the same mobility model using DL method, we formulated our optimization problem to minimize the sum of energy consumption and the finish time of the tasks. In order to solve such a MINLP algorithm, we applied GA to minimize the energy consumption required for execution of the tasks while satisfying the execution time deadline of users' applications. Simulation results show the superior performance of our method compared to other algorithms. It can

save up to 56.88% of energy when compared with on device execution of the tasks and 44.82% when compared with an all server execution strategy for complex tasks scenarios.

6.2 Open challenges and future research directions

Device-enhanced MEC computation offloading is a research area in flux and problem understanding under development. Numerous open problems and challenges remain that need to be resolved in this area. They are grouped into four main clusters, from the control and management of the device-enhanced MEC mechanisms, via improvements of the performance and scalability of the device-enhanced MEC mechanisms, to security and privacy, as well as performance evaluation aspects.

6.2.1 Control and management

6.2.1.1 Device-enhanced MEC management framework

One crucial information element for informed optimization processes is knowledge of the computational capabilities of end devices. Currently, no framework exists to exchange these information, neither in real-time, nor near-real-time manner in order to inform offloading decisions. Proprietary or stand-alone solutions, such as connection sharing in smartphones, do not scale because of the need for per device setup and activation, which prohibits automation. Consequently, an urgent need to develop and evaluate frameworks for control and management of device-enhanced MEC emerged.

One intuitive point of entry for developing such device-enhanced MEC control and management frameworks is to make use of recent successes of SDN control [41,42], especially in scalable control plane operation [162, 163], flow control [164], traffic engineering [165,166], routing [167–169], and Internet of Things management [170, 171]. In addition, SDN principles have been successfully employed for control and management of general wireless networks, see e.g. [43, 172–176]. Extending applications of the SDN controllers, management mechanisms could be incorporated for device-enhanced MEC [177–179]. This is resembling the principles of hybrid SDN [180,181], which allows for the control and management of hybrid systems that

combine conventional devices that are not SDN-enabled as well as SDN-enabled devices.

An equally important aspect for such a framework are considerations of the responsiveness for control and management. Over a localized network area, signaling of real-time computational and caching capabilities can be achieved quickly [182, 183], but for larger network areas, substantial signaling delays may be introduced. Ergo, future control and management frameworks for device-enhanced MEC will have to incorporate some fast localized decision making process with global coordination on a slower timescale, similar to what has recently been used in multi-timescale wireless resource allocation studies [125–130].

In addition, these future control and management frameworks for device-enhanced MEC will need to be robust towards heterogeneity across the variety of the different system characteristics, such as heterogeneity of the Radio Access Technology (RAT), end devices, and applications. For instance, manufacturers' everchanging use of various heterogeneous wireless medium access and transmission technologies has an impact on UE battery life time, link speed, and link reliability. Depending on the use case, the application needs, and the communications scenario, these trade-offs between UE battery lifetime, link speed, and link reliability could lead to a preference of a particular medium access technology, or a combination of medium access technologies, which then in turn may imply heterogeneous achievable communications ranges and UE discoverability. Future device-enhanced MEC control and management frameworks will need to be aware of these trade-offs across the standardized layers of the wireless networking protocol stack, from the physical layer, up to and including the application layer [2].

6.2.1.2 Interference management

Whenever several UEs offload their tasks to MEC servers or adjacent end devices use the same resources (e.g., time slots and frequency channels) at the same time, interference among multiple ongoing D2D communication links and between D2D communication and cellular communication arises. Aggravatingly, this interference worsens with increasing density of UEs in a network cell [184]. Dedicating radio resources exclusively to D2D communications solves the interference problem [185], but this is gained the expense of reduced reuse efficiency. To address this, multiple complementary interference management techniques, such as power control,

mode selection, and radio resource allocation, are generally used jointly to improve the network capacity as well as spectrum reuse efficiency.

A particularly challenging situation for interference management occurs in heterogeneous IoT networks with the limited computational resources of typical IoT nodes, where significant benefit from device-enhanced MEC is to be expected, mainly because of the massive numbers of expected connected IoT devices. Furthermore, the strongly heterogeneous transmit power levels of IoT devices result in equally heterogeneous interference levels. Consequently, several open challenges present themselves to efficiently manage the interference arising from device-enhanced MEC in IoT networks. One possible approach to address them is linking the transmission mode selection algorithm with the mechanisms for device-enhanced MEC computation offloading in order to make dynamic offloading decisions jointly, i.e. in accordance with the interference in the network environments. [186] introduced the general concept of adaptive mode selection. Coupling the mode decision making with the offloading decision making to obtain overall optimized offloading decisions considering the interference levels should be effectively studied in the future research [2].

6.2.2 Performance improvements and scalability

6.2.2.1 Social-aware D2D cooperative communication

The data exchange in D2D networks can be facilitated by exploiting the social characteristics of UEs. One example for social communities can be the location of UEs [187]; therefore, users' movements can lead to social disconnections. Thus, future research should study mechanisms to dynamically capture the location changes.
Some users in the network may produce dishonest information to enhance their own performance; therefore, social network discovery mechanisms must safeguard against this [188].

Furthermore, UEs should consume energy to maintain the social awareness in a D2D cooperative communications; Hence, energy-efficient social awareness in D2D cooperative communication should be investigated in future research studies [2, 189].

6.2.2.2 Learning algorithms

ML is widely accepted as a promising solution to autonomously and optimally configure future wireless networks, among other things. The approach makes use of information learned from recorded or otherwise observed network system behaviors [42, 190–194]. In fact, it has been speculated that most problems considered "hard" can be formulated as ML problems and solved by iteration and policy search [195]. The channel selection problem in D2D communication for example can be modeled as a simple multi-armed bandit game which falls into the category of reinforcement learning algorithms [196]. Much more involved and complex approaches for this and similar problems of control have been developed in the recent years. Similarly, techniques for wireless power control based on distributed Q-learning (another reinforcement learning approach) have been developed in [197]. Attractive as they may be, the required multiple iterations (often into the hundreds of thousands of steps) make ML approaches often a highly time-consuming affair. Future research should focus on time- and resource-efficient ML algorithms [198].

6.2.3 Security and privacy

When offloading computation tasks to adjacent devices, security of privacy is a considerable problem. Side channel attacks continue to be demonstrated with the eventual exploitation of UEs' personal information [199] , thus violating data security and privacy. Such data security breaches, if encountered and employed in the wild, could deter users from adopting task offloading schemes and hinder deployment of any such application, and slow development of the necessary frameworks. Aggravating, such security breaches could counteract the positive effects of offloading incentive mechanisms. As a result, considerable numbers of users may lose interest in participating in cooperation if they are at continuous risk. One further important problem is user mobility, which requires adaptive security mechanisms that account for the varying user locations.

Future research needs to comprehensively and on an ongoing basis address the security and privacy aspects of device-enhanced MEC. One approach could be build with a focus on the social user communities. For instance, UEs could be grouped based on their social relationships, interests, and locations. The resulting groups would be associated with a security level, and a given member of that group may or

may not participate in task offloading. A severer drawback of such groupings of UEs, and any other limitation of exposure, is the potential loss of collaboration opportunities due to hesitation to engage in collaborations with nearby strangers [200]. And of course, throughout the topic, the overhead of security methods for the network communication need to be carefully traded off against their benefits [188]. With implementation of procedures protecting against various attacks however, this overhead should be limited.

6.2.4 Performance evaluation

For the further advancement of the device-enhanced MEC area it will be critical to quantitatively compare various approaches and identify weaknesses that can then be addressed in future research. In order to facilitate quantitative performance comparisons, future research should develop comprehensive evaluation frameworks that specify the set of performance metrics as well as the performance evaluation methodologies that ensure rigorous replicable evaluations. The evaluation frameworks should include workload specifications, as well as wireless channel and mobility models, that the research community agrees on as being representative for common device-enhanced MEC scenarios.

6.3 Publications

This section presents the publications resulted from this thesis.

- Journals:
  —Mehrabi, Mahshid, et al. "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey." *IEEE Access 7* (2019): 166079-166108 [2].
  —Mehrabi, Mahshid, et al. "Mobility-and Energy-Aware Cooperative Edge Offloading for Dependent Computation Tasks." *Network 1.2* (2021): 191-214 [4].

- Conferences:
  —Mehrabi, Mahshid, et al. "A Survey on Mobility Management for MEC-enabled Systems." *IEEE 2nd 5G World Forum (5GWF), 2019* [201].

—Mehrabi, Mahshid, et al. "Accurate Energy-Efficient Localization Algorithm for IoT Sensors." *IEEE International Conference on Communications (ICC), 2020* [202].
—Mehrabi, Mahshid, et al. "Energy-aware cooperative offloading framework for inter-dependent and delay-sensitive tasks." *IEEE Global Communications Conference, 2020* [3].

Acronyms

5G	Fifth Generation of Cellular Communication Networks.
AP	Access Point.
AR	Augmented Reality.
BS	Base Station.
CPU	Central Processing Unit.
CSI	Channel State Information.
CUE	Computing User Equipment.
D2D	Device to Device.
DAG	Directed Acyclic Graph.
DL	Deep Learning.
EPC	Evolved Packet Core.
ETSI	European Telecommunications Standards Institute.
FD	Full Duplex.
FDD	Frequency Division Duplex.

FDMA Frequency-Division Multiple Access.

GA Genetic Algorithm.

HUE Helper User Equipment.

ICN Information Centric Network.
IoT Internet of Things.
ISG Industry Specification Group.

MBS Macro Base Station.
MCC Mobile Cloud Computing.
MDP Markov Decision Process.
MEC Multi-Access Edge Computing.
MINLP Mixed Integer Nonlinear Programming.
ML Machine Learning.
MSC Mobile Small Cell.
MTC Machine Type Communication.

NCC Network-coded Cooperation.
NFC Near Field Communication.
NFV Network Function Virtualization.
NLP Non-Linear Programming.
NN Neural Network.

OFDMA Orthogonal Frequency-Division Multiple Access.

QCQP Quadratically Constrained Quadratic Programming.
QoE Quality of Experience.
QoS Quality of Service.

RAN Radio Access Network.
RAT Radio Access Technology.
RB Resource Blocks.

SDN Software Defined Networking.
SDP Semidefinite Programming.
SDR Semidefinite Relaxation Method.
SFC Service Function Chains.

TDMA Time-Division Multiple Access.
TI Tactile Internet.

UE User Equipment.

V2I Vehicle to Infrastructure.
V2N Vehicle to Network.
V2V Vehicle to Vehicle.
V2X Vehicle to Anything.
VAE Variational Auto-Encoder.
VR Virtual Reality.

WLAN Wireless Local Area Network.

Bibliography

[1] R. Parsamehr, A. Esfahani, G. Mantas, A. Radwan, S. Mumtaz, J. Rodriguez, and J.-F. Martínez-Ortega, "A novel intrusion detection and prevention scheme for network coding-enabled mobile small cells," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1467–1477, 2019.

[2] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. Fitzek, "Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166 079–166 108, 2019.

[3] M. Mehrabi, S. Shen, V. Latzko, Y. Wang, and F. H. Fitzek, "Energy-aware cooperative offloading framework for inter-dependent and delay-sensitive tasks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[4] M. Mehrabi, S. Shen, Y. Hai, V. Latzko, G. P. Koudouridis, X. Gelabert, M. Reisslein, and F. H. Fitzek, "Mobility-and energy-aware cooperative edge offloading for dependent computation tasks," *Network*, vol. 1, no. 2, pp. 191–214, 2021.

[5] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *European Conference on Computer Vision*. Springer, 2020, pp. 759–776.

[6] J. Rodriguez, A. Radwan, C. Barbosa, F. H. P. Fitzek, R. A. Abd-Alhameed, J. M. Noras, S. M. R. Jones, I. Politis, P. Galiotos, G. Schulte, A. Rayit, M. Sousa, R. Al-

heiro, X. Gelabert, and G. P. Koudouridis, "SECRET – Secure network coding for reduced energy next generation mobile small cells: A European training network in wireless communications and networking for 5G," in *Proc. Internet Technologies and Applications*, Wrexham, Sep. 2017, pp. 329–333.

[7] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[8] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[9] R. Kemp, N. Palmer, T. Kielmann, F. Seinstra, N. Drost, J. Maassen, and H. Bal, "eyedentify: Multimedia cyber foraging from a smartphone," in *2009 11th IEEE International Symposium on Multimedia*. IEEE, 2009, pp. 392–399.

[10] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for iot sensors," *Sensors*, vol. 19, no. 5, p. 1105, 2019.

[11] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2019.

[12] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[13] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems," *IEEE Access, in print*, pp. 1–1, 2019.

[14] X. Wang, Y. Ji, J. Zhang, L. Bai, and M. Zhang, "Low-latency oriented network planning for MEC-enabled WDM-PON based fiber-wireless access networks," *IEEE Access, in print*, pp. 1–1, 2019.

[15] M. A. Habibi, M. Nasimi, B. Han, and H. D. Schotten, "A comprehensive survey of RAN architectures towards 5G mobile communication system," *IEEE Access*, vol. 7, pp. 70 371–70 421, 2019.

[16] B. Shi, J. Yang, Z. Huang, and P. Hui, "Offloading guidelines for augmented reality applications on wearable devices," in *Proc. ACM Int. Conf. on Multimedia*, Brisbane, Australia, 2015, pp. 1271–1274.

[17] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider, "Enabling real-time context-aware collaboration through 5G and mobile edge computing," in *Proc. Int. Conf. on Information Technology-New Generations*, Las Vegas, NV, USA, Apr. 2015, pp. 601–605.

[18] Z. Xiang, F. Gabriel, E. Urbano, G. T. Nguyen, M. Reisslein, and F. H. Fitzek, "Reducing latency in virtual machines: Enabling tactile internet for human-machine co-working," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1098–1116, 2019.

[19] Cisco VNI Forecast, "Cisco visual networking index: Global mobile data traffic forecast 2017-2022," Cisco Inc., Tech. Rep., November 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[20] G. P. Fettweis, "The tactile Internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[21] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[22] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh, "Machine-type communications: Current status and future perspectives toward 5G systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10–17, Sep. 2015.

[23] F. Al-Turjman, "5G-enabled devices and smart-spaces in social-IoT: An overview," *Future Generation Computer Systems*, vol. 92, pp. 732–744, Mar. 2019.

[24] J. Iqbal, M. A. Iqbal, A. Ahmad, M. Khan, A. Qamar, and K. Han, "Comparison of spectral efficiency techniques in device-to-device communication for 5G," *IEEE Access*, vol. 7, pp. 57 440–57 449, 2019.

[25] L. Militano, G. Araniti, M. Condoluci, I. Farris, and A. Iera, "Device-to-device communications for *5G Internet* of things," *EAI Endorsed Transactions on Internet of Things*, vol. 1, no. 1, pp. 1–15, 2015.

[26] R. Paul and Y. J. Choi, "Autonomous interface selection for multi-radio D2D communication," *IEEE Access*, vol. 7, pp. 108 090–108 100, 2019.

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

[28] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[29] H. Qi and A. Gani, "Research on mobile cloud computing: Review, trend and perspectives," in *Proc. Int. Conf. on Digital Information and Communication Technology and It's Applications*, Bangkok, Thailand, May 2012, pp. 195–202.

[30] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010.

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

[32] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[33] I. Morris, "ETSI drops mobile from MEC," Sep. 2016. [Online]. Available: https://www.lightreading.com/mobile/mec-(mobile-edge-computing)/etsi-drops-mobile-from-mec/d/d-id/726273

[34] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, May 2017.

[35] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. on Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[36] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, Oct. 2018.

[37] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[38] X. Hu, K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.

[39] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[40] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, and M. Paolino, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[41] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.

[42] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, and S. Schmid, "Adaptable and data-driven softwarized networks: Review, opportunities, and challenges," *Proc. IEEE*, vol. 107, no. 4, pp. 711–731, 2019.

[43] S. Jeon, C. Guimarães, and R. L. Aguiar, "SDN: Based mobile networking for cellular operators," in *Proc. ACM Workshop on Mobility in the Evolving Internet Architecture*, Maui, Hawaii, USA, 2014, pp. 13–18.

[44] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[45] C.-M. Huang, M.-S. Chiang, D.-T. Dao, W.-L. Su, S. Xu, and H. Zhou, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17 741–17 755, 2018.

[46] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "CDS-MEC: NFV/SDN-based application management for MEC in 5G systems," *Computer Networks*, vol. 135, pp. 96–107, 2018.

[47] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[48] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[49] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Network functions virtualization: The long road to commercial deployments," *IEEE Access*, vol. 7, pp. 60 439–60 464, 2019.

[50] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 487–507, 2018.

[51] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2016.

[52] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang, and A. V. Vasilakos, "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet of Things Journal, in print*, 2019.

[53] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[54] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network

slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.

[55] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[56] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "Information-centric networking (ICN) research challenges," pp. 1–38, 2016, Internet Research Task Force (IRTF), Request for Comments 7927.

[57] G. Piro, L. A. Grieco, G. Boggia, and P. Chatzimisios, "Information-centric networking and multimedia services: Present and future challenges," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 4, pp. 392–406, Apr. 2014.

[58] A. V. Vasilakos, Z. Li, G. Simon, and W. You, "Information centric network: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 52, pp. 1–10, Jun. 2015.

[59] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, "Mobility-aware caching for content-centric wireless networks: Modeling and methodology," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 77–83, Aug. 2016.

[60] F. Al-Turjman, "5g-enabled devices and smart-spaces in social-iot: an overview," *Future Generation Computer Systems*, vol. 92, pp. 732–744, 2019.

[61] G. Ahani and D. Yuan, "BS-assisted task offloading for D2D networks with presence of user mobility," in *Proc. IEEE Vehicular Techn. Conf. (VTC2019-Spring)*, 2019, pp. 1–5.

[62] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1816–1820, Oct. 2019.

[63] J. Liu, K. Luo, Z. Zhou, and X. Chen, "ERP: Edge resource pooling for data stream mobile computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4355–4368, Jun. 2019.

[64] W. Xiao, X. Zhu, W. Bao, L. Liu, and J. Yao, "Cooperative data sharing for mobile cloudlets under heterogeneous environments," *IEEE Trans. on Network and Service Management*, vol. 16, no. 2, pp. 430–444, Jun. 2019.

[65] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "D2D computation offloading optimization for precedence-constrained tasks in information-centric IoT," *IEEE Access*, vol. 7, pp. 94 888–94 898, 2019.

[66] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. on Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.

[67] C. You and K. Huang, "Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing," *IEEE Trans. on Wireless Commun.*, vol. 17, no. 6, pp. 4104–4117, Jun. 2018.

[68] K. N. Doan, T. V. Nguyen, H. Shin, and T. Q. S. Quek, "Socially-aware caching in wireless networks with random D2D communications," *IEEE Access*, vol. 7, pp. 58 394–58 406, 2019.

[69] J. Huang, C. Huang, C. Xing, Z. Chang, Y. Zhao, and Q. Zhao, "An energy-efficient communication scheme for collaborative mobile clouds in content sharing: Design and optimization," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 10, pp. 5700–5707, Oct. 2019.

[70] W. Li, C. Wang, D. Li, B. Hu, X. Wang, and J. Ren, "Edge caching for D2D enabled hierarchical wireless networks with deep reinforcement learning," *Wireless Communications and Mobile Computing*, vol. 2019, no. 2561069, pp. 1–12, 2019.

[71] C. Ma, M. Ding, H. Chen, Z. Lin, G. Mao, Y. Liang, and B. Vucetic, "Socially aware caching strategy in device-to-device communication networks," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 5, pp. 4615–4629, May 2018.

[72] A. Said, S. Shah, H. Farooq, A. Mian, A. Imran, and J. Crowcroft, "Proactive caching at the edge leveraging influential user detection in cellular D2D networks," *Future Internet*, vol. 10, no. 10, pp. 93.1–93.17, 2018.

[73] S. Soleimani and X. Tao, "Caching and placement for in-network caching in device-to-device communications," *Wireless Commun. and Mobile Computing*, vol. 2018, no. 9539502, pp. 1–9, 2018.

[74] N. Zhao, X. Liu, Y. Chen, S. Zhang, Z. Li, B. Chen, and M. Alouini, "Caching D2D connections in small-cell networks," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 12, pp. 12 326–12 338, Dec. 2018.

[75] W. Zhang, D. Wu, W. Yang, and Y. Cai, "Caching on the move: A user interest-driven caching strategy for D2D content sharing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2958–2971, Mar. 2019.

[76] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy-efficient mobile edge collaboration for video distribution," *IEEE Trans. on Multimedia*, vol. 19, no. 10, pp. 2197–2209, Oct. 2017.

[77] D. Wu, Q. Liu, H. Wang, Q. Yang, and R. Wang, "Cache less for more: Exploiting cooperative video caching and delivery in D2D communications," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1788–1798, Jul. 2019.

[78] G. Fodor, S. Parkvall, S. Sorrentino, P. Wallentin, Q. Lu, and N. Brahmi, "*Device-to-Device Communications for National Security and Public Safety*," *IEEE Access*, vol. 2, pp. 1510–1520, 2014.

[79] S. Mumtaz and J. Rodriguez, *Smart Device to Smart Device Communication*. Springer, New York, 2014.

[80] L. Lei, Z. Zhong, C. Lin, and X. Shen, "Operator controlled device-to-device communications in LTE-advanced networks," *IEEE Wireless Communications*, vol. 19, no. 3, pp. 96–104, 2012.

[81] M. Höyhtyä, O. Apilo, and M. Lasanen, "Review of latest advances in 3GPP standardization: D2D communication in 5G systems and its energy consumption models," *Future Internet*, vol. 10, no. 1, p. 3, Jan. 2018.

[82] A. Masmoudi, S. Feki, K. Mnif, and F. Zarai, "Efficient scheduling and resource allocation for D2D-based LTE-V2X communications," in *Proc. IEEE Int. Wireless Commun. & Mobile Computing Conf. (IWCMC)*, 2019, pp. 496–501.

[83] G. Nardini, A. Virdis, C. Campolo, A. Molinaro, and G. Stea, "Cellular-V2X communications for platooning: Design and evaluation," *Sensors*, vol. 18, no. 5, pp. 1527.1–1527.22, 2018.

[84] S. Singh, J. Lianghai, D. Calabuig, D. Garcia-Roger, N. H. Mahmood, N. Pratas, T. Mach, and M. C. DeGennaro, "D2D and V2X communications," in *5G System Design: Architectural and Functional Considerations and Long Term Research*. John Wiley & Sons, 2018, pp. 409–449.

[85] T. Bertram, *Fahrerassistenzsysteme 2018: Von der Assistenz zum automatisierten Fahren: 4. Internationale ATZ-Fachtagung Automatisiertes Fahren*. Wiesbaden: Springer Vieweg, 2019.

[86] A. Feroz *et al.*, "Vehicle to vehicle communication for collision avoidance," *International Journal of Emerging Technology and Innovative Engineering*, vol. 5, no. 7, 2019.

[87] Y. Cao, T. Jiang, O. Kaiwartya, H. Sun, H. Zhou, and R. Wang, "Toward preempted EV charging recommendation through V2V-based reservation system," *IEEE Trans. on Systems, Man, and Cybernetics: Systems, in print*, pp. 1–14, 2019.

[88] A. A. Ateya, A. Muthanna, and A. Koucheryavy, "5G framework based on multilevel edge computing with D2D enabled communication," in *Proc. Int. Conf. on Adv. Commun. Techn. (ICACT)*, Feb. 2018, pp. 507–512.

[89] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *Proc. Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Shanghai, China, May 2018, pp. 1–6.

[90] C. You and K. Huang, "Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4104–4117, Jun. 2018.

[91] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 176–189, Jan. 2016.

[92] P. Zhang, X. Kang, Y. Liu, and H. Yang, "Cooperative willingness aware collaborative caching mechanism towards cellular D2D communication," *IEEE Access*, vol. 6, pp. 67 046–67 056, 2018.

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

[94] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018.

[95] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138–147, May 2017.

[96] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in *Proc. IEEE Int. Conf. on Communications*, Kansas City, MO, May 2018, pp. 1–6.

[97] ——, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.

[98] S. Gupta and A. Lozano, "Computation-bandwidth trading for mobile edge computing," in *Proc. IEEE Annual Consumer Communications & Networking Conf.*, Las Vegas, NV, USA, Jan. 2019, pp. 1–6.

[99] F. Wang, J. Xu, and Z. Ding, "Optimized multiuser computation offloading with multi-antenna NOMA," in *Proc. IEEE Global Communications Conf. Workshops*, Singapore, Dec. 2017, pp. 1–7.

[100] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, Aug. 2017.

[101] F. Jiang, H. Wang, H. Ren, and S. Xu, "Energy-efficient resource and power allocation for underlay multicast device-to-device transmission," *Future Internet*, vol. 9, no. 4, p. 84, Nov. 2017.

[102] G. Hu, Y. Jia, and Z. Chen, "Multi-user computation offloading with D2D for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[103] C. Wang, J. Qin, X. Yang, and W. Wen, "Energy-efficient offloading policy in D2D underlay communication integrated with MEC service," in *Proc. Int. Conf. on High Perf. Compilation, Comput. and Commun.*, 2019, pp. 159–164.

[104] G. Scutari, S. Barbarossa, and D. P. Palomar, "Potential games: A framework for vector power control problems with coupled constraints," in *Proc. IEEE Int. Conf. on Acoustics Speech and Signal Proc.*, vol. 4, 2006, pp. 241–244.

[105] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D Fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[106] F. Ekman, A. Keränen, J. Karvo, and J. Ott, "Working day movement model," in *Proc. ACM SIGMOBILE Workshop on Mobility Models*, 2008, pp. 33–40.

[107] Q. Jia, R. Xie, Q. Tang, X. Li, T. Huang, J. Liu, and Y. Liu, "Energy-efficient computation offloading in 5G cellular networks with edge computing and D2D communications," *IET Communications*, vol. 13, no. 8, pp. 1122–1130, May 2019.

[108] G. Qiao, S. Leng, and Y. Zhang, "Online learning and optimization for computation offloading in D2D edge computing and networks," *Mobile Networks and Applications, in print*, 2019.

[109] D. Wang, Y. Lan, T. Zhao, Z. Yin, and X. Wang, "On the design of computation offloading in cache-aided D2D multicast networks," *IEEE Access*, vol. 6, pp. 63 426–63 441, 2018.

[110] C.-Y. Wang, Y. Chen, and K. R. Liu, "Chinese restaurant game," *IEEE Signal Processing Letters*, vol. 19, no. 12, pp. 898–901, 2012.

[111] J. Lin, R. Chai, M. Chen, and Q. Chen, "Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D systems," in *Proc. IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications*, Bologna, Italy, Sep. 2018, pp. 1–5.

[112] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[113] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. on Communications*, Paris, France, May 2017, pp. 1–6.

[114] S. Even, G. Even, and R. M. Karp, *Graph Algorithms*, 2nd ed. Cambridge, UK: Cambridge University Press, 2012.

[115] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[116] A. Orsino, I. Farris, L. Militano, G. Araniti, S. Andreev, I. A. Gudkova, Y. Koucheryavy, and A. Iera, "Exploiting D2D communications at the network edge for mission-critical IoT applications," in *Proc. European Wireless Conf.*, 2017, pp. 68–73.

[117] D. B. Johnson and D. A. Maltz, *Dynamic source routing in ad hoc wireless networks*. Springer, 1996.

[118] R. Ranji, A. M. Mansoor, and A. A. Sani, "EEDOS: An energy-efficient and delay-aware offloading scheme based on device to device collaboration in mobile edge computing," *Telecommunication Systems, in print*, pp. 1–12, 2019.

[119] K. Sucipto, D. Chatzopoulos, S. Kostap, and P. Hui, "Keep your nice friends close, but your rich friends closer–computation offloading using NFC," in *Proc. IEEE Conf. on Computer Communications*, Atlanta, GA, USA, May 2017, pp. 1–9.

[120] H. Flores, R. Sharma, D. Ferreira, V. Kostakos, J. Manner, S. Tarkoma, P. Hui, and Y. Li, "Social-aware hybrid mobile offloading," *Pervasive and Mobile Computing*, vol. 36, pp. 25–43, Apr. 2017.

[121] Y. He, F. R. Yu, N. Zhao, and H. Yin, "Secure social networks in 5G systems with mobile edge computing, caching, and device-to-device communications," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 103–109, Jun. 2018.

[122] H. Zhou, V. C. M. Leung, C. Zhu, S. Xu, and J. Fan, "Predicting temporal social contact patterns for data forwarding in opportunistic mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 372–10 383, Nov. 2017.

[123] H. Flores, P. Nurmi, and P. Hui, "AI on the move: From on-device to on-multi-device," in *Proc. IEEE Int. Conf. on Pervasive Comp and Commun. Workshops (PerCom Workshops)*, 2019, pp. 310–315.

[124] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, p. 2423, 2018.

[125] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2716–2730, 2018.

[126] N. Prasad, M. Arslan, and S. Rangarajan, "A two time scale approach for coordinated multi-point transmission and reception over practical backhaul," in *Proc. Int. Conf. on Commun. Sys. and Netw. (COMSNETS)*, 2014, pp. 1–8.

[127] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57 545–57 561, 2018.

[128] J. Tang, B. Shim, and T. Q. S. Quek, "Service multiplexing and revenue maximization in sliced C-RAN incorporated with URLLC and multicast eMBB," *IEEE J. on Sel. Areas in Commun.*, vol. 37, no. 4, pp. 881–895, Apr. 2019.

[129] M. Wang, N. Karakoc, L. Ferrari, P. Shantharama, A. S. Thyagaturu, M. Reisslein, and A. Scaglione, "A multi-layer multi-timescale network utility maximization framework for the SDN-based layback architecture enabling wireless backhaul resource sharing," *Electronics*, vol. 8, no. 9, pp. 937–1–937–28, 2019.

[130] W. Xia, T. Q. S. Quek, J. Zhang, S. Jin, and H. Zhu, "Programmable hierarchical C-RAN: From task scheduling to resource allocation," *IEEE Trans. on Wireless Commun.*, vol. 18, no. 3, pp. 2003–2016, March 2019.

[131] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.

[132] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[133] Cisco, "Fog computing and the internet of things: Extend the cloud to where the things are," 2016. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[134] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[135] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh, "Machine-type communications: current status and future perspectives toward 5g systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10–17, 2015.

[136] C. Wang, J. Qin, X. Yang, and W. Wen, "Energy-efficient offloading policy in d2d underlay communication integrated with mec service," in *Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications*, 2019, pp. 159–164.

[137] Q. Jia, R. Xie, Q. Tang, X. Li, T. Huang, J. Liu, and Y. Liu, "Energy-efficient computation offloading in 5g cellular networks with edge computing and d2d communications," *IET Communications*, vol. 13, no. 8, pp. 1122–1130, 2019.

[138] G. Qiao, S. Leng, and Y. Zhang, "Online learning and optimization for computation offloading in d2d edge computing and networks," *Mobile Networks and Applications*, pp. 1–12, 2019.

[139] D. Wang, Y. Lan, T. Zhao, Z. Yin, and X. Wang, "On the design of computation offloading in cache-aided d2d multicast networks," *IEEE Access*, vol. 6, pp. 63 426–63 441, 2018.

[140] Y. Cheng, J. Zhang, L. Yang, C. Zhu, and H. Zhu, "Distributed green offloading and power optimization in virtualized small cell networks with mobile edge computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 1, pp. 69–82, 2019.

[141] B. Yang, D. Wu, H. Wang, Y. Gao, and R. Wang, "Two-layer stackelberg game based offloading strategy for mobile edge computing enhanced fiwi access networks," *IEEE Transactions on Green Communications and Networking*, 2020.

[142] R. Ranji, A. M. Mansoor, and A. A. Sani, "Eedos: an energy-efficient and delay-aware offloading scheme based on device to device collaboration in mobile edge computing," *Telecommunication Systems*, vol. 73, no. 2, pp. 171–182, 2020.

[143] C. Wu, Q. Peng, Y. Xia, and J. Lee, "Mobility-aware tasks offloading in mobile edge computing environment," in *2019 Seventh International Symposium on Computing and Networking (CANDAR)*. IEEE, 2019, pp. 204–210.

[144] D. Wang, Z. Liu, X. Wang, and Y. Lan, "Mobility-aware task offloading and migration schemes in fog computing networks," *IEEE Access*, vol. 7, pp. 43 356–43 368, 2019.

[145] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Energy-aware qoe and backhaul traffic optimization in green edge adaptive mobile video streaming," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 3, pp. 828–839, 2019.

[146] T. Zhang and W. Chen, "Computation offloading in heterogeneous mobile edge computing with energy harvesting," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 552–565, 2021.

[147] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial iot in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2019.

[148] H. Chen, D. Zhao, Q. Chen, and R. Chai, "Joint computation offloading and radio resource allocations in small-cell wireless cellular networks," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 3, pp. 745–758, 2020.

[149] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301–313, 2016.

[150] W. Hao and S. Yang, "Small cell cluster-based resource allocation for wireless backhaul in two-tier heterogeneous networks with massive mimo," *Ieee transactions on vehicular technology*, vol. 67, no. 1, pp. 509–523, 2017.

[151] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[152] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[153] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[154] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[155] T. Yang, R. Chai, L. Zhang, and Q. Chen, "Worst case latency optimization-based joint computation offloading and scheduling for interdependent subtasks," in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2020, pp. 1010–1015.

[156] L. Davis, "Handbook of genetic algorithms," 1991.

[157] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, 2020.

[158] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[159] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang, "Exploiting joint computation offloading and data caching to enhance mobile terminal performance," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[160] M. Mehrabi, H. Taheri, and P. Taghdiri, "An improved dv-hop localization algorithm based on evolutionary algorithms," *Telecommunication Systems*, vol. 64, no. 4, pp. 639–647, 2017.

[161] S. Yu, "Multi-user computation offloading in mobile edge computing," *Online]. https://www. researchgate. net/publication/328629402 Multiuser Computation Offloading in Mobile Edge Computing*, 2018.

[162] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[163] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[164] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable OpenFlow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107 346–107 379, 2019.

[165] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[166] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 918–953, 2016.

[167] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 464–486, 2018.

[168] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, 2017.

[169] F. Y. Okay and S. Ozdemir, "Routing in fog-enabled IoT platforms: A survey and an SDN-based solution," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4871–4889, 2018.

[170] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the Internet of Things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[171] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 812–837, 2018.

[172] X. Costa-Perez, A. Garcia-Saavedra, X. Li, T. Deiss, A. de la Oliva, A. di Giglio, P. Iovanna, and A. Moored, "5G-Crosshaul: An SDN/NFV integrated fronthaul/backhaul transport network architecture," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 38–45, 2017.

[173] I. Elgendi, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Traffic offloading techniques for 5G cellular: a three-tiered SDN architecture," *Annals of Telecommunications*, pp. 1–11, 2016.

[174] D. King, A. Farrel, E. N. King, R. Casellas, L. Velasco, R. Nejabati, and A. Lord, "The dichotomy of distributed and centralized control: METRO-HAUL, when control planes collide for 5G networks," *Optical Switching and Networking*, vol. 33, pp. 49–55, 2019.

[175] J. d. C. Silva, J. J. P. C. Rodrigues, J. Al-Muhtadi, R. A. L. Rabelo, and V. Furtado, "Management platforms and protocols for internet of things: A survey," *Sensors*, vol. 19, no. 3, 2019. [Online]. Available: http://www.mdpi.com/1424-8220/19/3/676

[176] A. Thyagaturu, Y. Dashti, and M. Reisslein, "SDN based smart gateways (SmGWs) for multi-operator small cell network management," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 740–753, 2016.

[177] A. A. Abbasi, A. Abbasi, S. Shamshirband, A. T. Chronopoulos, V. Persico, and A. Pescape, "Software-defined cloud computing: A systematic review on latest trends and developments," *IEEE Access*, vol. 7, pp. 93 294–93 314, 2019.

[178] A. Binsahaq, T. R. Sheltami, and K. Salah, "A survey on autonomic provisioning and management of QoS in SDN networks," *IEEE Access*, vol. 7, pp. 73 384–73 435, 2019.

[179] A. A. Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature," *IEEE Access*, vol. 6, pp. 14 159–14 178, 2018.

[180] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[181] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, and S. Hu, "A survey of deployment solutions and optimization strategies for hybrid sdn networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1483–1507, 2018.

[182] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

[183] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[184] K. Doppler, M. Rinne, C. Wijting, C. Ribeiro, and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, Dec. 2009.

[185] D. You and D. H. Kim, "Multi-device-to-multi-device communication in cellular network for efficient contents distribution," in *Proc. IEEE Int. Conf. on Consumer Electronics*, Las Vegas, NV, USA, Jan. 2014, pp. 244–247.

[186] D. You, T. V. Doan, R. Torre, M. Mehrabi, A. Kropp, V. Nguyen, H. Salah, G. T. Nguyen, and F. H. P. Fitzek, "Fog computing as an enabler for immersive media: Service scenarios and research opportunities," *IEEE Access*, vol. 7, pp. 65 797–65 810, 2019.

[187] B. Zhang, Y. Li, D. Jin, P. Hui, and Z. Han, "Social-aware peer discovery for D2D communications underlaying cellular networks," *IEEE Transactions on Wireless Communications*, vol. 14, no. 5, pp. 2426–2439, May 2015.

[188] R. I. Ansari, C. Chrysostomou, S. A. Hassan, M. Guizani, S. Mumtaz, J. Rodriguez, and J. J. P. C. Rodrigues, "5G D2D networks: Techniques, challenges, and future prospects," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3970–3984, Dec. 2018.

[189] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Green cooperative device–to–device communication: A social–aware perspective," *IEEE Access*, vol. 4, pp. 3697–3707, 2016.

[190] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks," *IEEE Access*, vol. 6, pp. 32 328–32 338, 2018.

[191] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2595–2621, Fourth Qu. 2018.

[192] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys Tutorials, in print*, pp. 1–1, 2019.

[193] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, First Qu. 2019.

[194] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, Third Qu. 2019.

[195] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[196] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[197] Z. Fan, X. Gu, S. Nie, and M. Chen, "D2D power control based on supervised and unsupervised learning," in *Proc. IEEE Int. Conf. on Computer and Communications*, Chengdu, China, Dec. 2017, pp. 558–563.

[198] J. Xu, X. Gu, and Z. Fan, "D2D power control based on hierarchical extreme learning machine," in *Proc. IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications*, Bologna, Italy, Sep. 2018, pp. 1–7.

[199] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 465–488, 2017.

[200] M. Peng and K. Zhang, "Recent advances in fog radio access networks: Performance analysis and radio resource allocation," *IEEE Access*, vol. 4, pp. 5003–5009, 2016.

[201] M. Mehrabi, H. Salah, and F. H. Fitzek, "A survey on mobility management for mec-enabled systems," in *2019 IEEE 2nd 5G World Forum (5GWF)*. IEEE, 2019, pp. 259–263.

[202] M. Mehrabi, P. Taghdiri, V. Latzko, H. Salah, and F. H. Fitzek, "Accurate energy-efficient localization algorithm for iot sensors," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.