Binary Multi-User Computation Offloading via Time Division Multiple Access

Binary Multi-User Computation Offloading via Time Division Multiple Access

By Amin Manouchehrpour,

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Master of Applied Science

September 27, 2023

McMaster University

Master of Applied Science (2023)

Hamilton, Ontario (ECE Department)

TITLE: Binary Multi-User Computation Offloading via Time Division Multiple Access

AUTHOR: Amin Manouchehrpour

B.Sc., (Electrical Engineering)

University of Tehran, Tehran, Iran

SUPERVISOR: Dr. Tim Davidson

NUMBER OF PAGES: xvi, 126

To my beloved parents

Mona *and* ***Alireza***

&

to the most inspiring person in my universe, my grandfather

Majid

Abstract

The limited energy and computing power of small smart devices restrict their ability to support a wide range of applications, especially those needing quick responses. Mobile edge computing offers a potential solution by providing computing resources at the network access points that can be shared by the devices. This enables the devices to offload some of their computational tasks to the access points. To make this work well for multiple devices, we need to judiciously allocate the available communication and computing resources among the devices. The main focus of this thesis is on (near) optimal resource allocation in a K-user offloading system that employ the time division multiple access (TDMA) scheme. In this thesis we develop effective algorithms for the resource allocation problem that aim to minimize the overall (cost of the) energy that the devices consume in completing their computational tasks within the specified deadlines, while respecting the devices' constraints. This problem is tackled for tasks that cannot be divided and hence the system must make a binary decision as to whether or not a task should be offloaded. This implies the need to develop an effective decision-making algorithm to identify a suitable group of devices for offloading.

This thesis commences by developing efficient communication resource algorithms that incorporate the impact of integer finite block length in low-latency computational offloading systems with reserved computing resources. In particular, it addresses the challenge of minimizing total energy consumption in a binary offloading scenario involving K users. The approach considers different approximations of the fundamental rate limit in the finite block length regime, departing from the conventional asymptotic rate limits developed by Shannon. Two such

alternatives, namely the normal approximation and the SNR-gap approximation, are explored. A decomposition approach is employed, dividing the problem into an inner component that seeks an optimal solution for the communication resource allocation within a defined set of offloading devices, and an outer component aimed at identifying a suitable set of offloading devices. Given the finiteness of the block length and its integer nature, various relaxation techniques are employed to determine an appropriate communication resource allocation. These include incremental and independent roundings, alongside an extended search that utilizes randomization-based methods in both rounding schemes. The findings reveal that incremental randomized rounding, when applied to the normal approximation of the rate limits, enhances system performance in terms of reducing the energy consumption of the offloading users. Furthermore, customized pruned greedy search techniques for selecting the offloading devices efficiently generate good decisions. Indeed, the proposed approach outperforms a number of existing approaches.

In the second contribution, we develop efficient algorithms that address the challenge of jointly allocating both computation and communication resources in a binary offloading system. We employ a similar decomposition methodology as in the previous work to perform the decision making, but this is now done along with joint computation and communication resource allocation. For the inner resource allocation problem we divide the problem into two components: determining the allocation of computation resources and the optimal allocation of communication resources for the given allocation of computation resources. The allocation of the computation resources implicitly determines a suitable order for data transmission, which facilitates the subsequent optimal allocation of the communication resources.

In this thesis, we introduce two heuristic approaches for allocating the computation resources. These approaches sequentially maximize the allowable transmission time for the devices in sequence, starting from the largest leading to a reduction in total offloading energy. We demonstrate that the proposed heuristics substantially lower the computational burden associated with solving the joint computation–communication resource allocation problem, while maintaining a low total energy. In particular, its use results in substantially lower energy consumption than other simple heuristics. Additionally, the heuristics narrow the energy gap in comparison to a fictitious scenario in which each task has access to the whole computation resource without the need for sharing

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Dr. Timothy N. Davidson, for his unwavering guidance and generous support throughout this journey. I am truly delighted to work under his supervision, and I have learned a lot from him, not only technically but also in terms of his exemplary conduct and interactions. Without his profound insights, continuous support, and encouragement, this work would never have come to fruition

I extend my sincere appreciation to the administrative staff, especially, Cheryl Gies, and Tracey Coop, for their efficient coordination and assistance in navigating the intricacies of this academic pursuit.

To my beloved parents, Mona and Alireza, your unconditional support and unwavering belief in me have been my foundation. I learned to be a fighter during life's storms from my greatest supporter, my father, and I would never ever have been able to finish this journey without my mom's encouragement and her immeasurable love that she has been sending to me my whole life.

I am sincerely thankful for my friends Ali Vakili and Mehrad Mehrabi, who consistently stay in touch with me, bridging the distance and keeping us connected, regardless of the miles between us.

Last but not least, I owe a special debt of gratitude to Zahra. Her exceptional support and unwavering encouragement have played a pivotal role in keeping me motivated and focused. Your assistance have surpassed all expectations, and I am truly indebted to you. Your presence has illuminated my journey.

Abbreviations

AP	Access Point
C-SMMDT	Contiguous Sequentially Maximized Maximum Delivery Time
CPU	Central Processing Units
IEEE	Institute of Electrical and Electronics Engineers
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
NA	Normal Approximation
NOMA	Non-Orthogonal Multiple Access
OMA	Orthogonal Multiple Access
QoS	Quality of Service
SCA	Successive Convex Approximation
SMMDT	Sequentially Maximized Maximum Delivery Time
SNR	Signal to Noise Ratio
TDMA	Time Division Multiple Access
TTA	Two-Term Approximation

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Amin Manouchehrpour, declare that this thesis titled, "Binary Multi-User Computation Offloading via Time Division Multiple Access" and the work presented in it are my own.

Chapter 1

Introduction

The rapid growth in the use of smart devices, the running of computationally expensive applications in our smart phones, and the widespread deployment of wireless communication networks have ushered in a new concept known as computational offloading; e.g., Kumar et al. (2013), Mach and Becvar (2017), and Taleb et al. (2017). This computing paradigm enables devices to offload their computational tasks to external resources, overcoming the limitations posed by their own computational, storage, and energy constraints. Computational offloading is a suitable way to enhance the performance of smart devices and empower them to support a wide range of resource-intensive applications.

As the demand for more advanced and computationally–complex applications continues to rise, the inherent limitations of smart devices has become apparent. Despite advancements in CPUs, storage, and battery life, these devices may require more powerful processing units to meet the computational and energy requirements of emerging latency–sensitive applications, such as augmented reality, artificial intelligence, and real-time data processing. The need for innovative

solutions that can bridge this gap has led to the development of computational offloading techniques.

Computational offloading involves the transfer of computational tasks from a smart device to external resources capable of handling them effectively. By leveraging the computational power and storage capacities of remote servers or cloud-based platforms, and the bandwidth provided by modern wireless (and wire-line) communication networks, smart devices can offload computationally–intensive portions of applications, reducing their processing load and conserving energy. This allows devices with limited resources to perform complex computations that would otherwise be impractical or infeasible.

The success of computational offloading depends on several factors. Network latency, reliability, and bandwidth availability play a crucial role in determining the overall performance and user experience. Efficient load balancing mechanisms and intelligent decision-making algorithms are essential to determine which tasks or data should be offloaded and when, ensuring optimal resource utilization and timely execution.

By embracing computational offloading, smart devices can overcome their inherent limitations and extend their capabilities to support a wide range of sophisticated applications. This paradigm not only enhances the performance and energy efficiency of individual devices, but also enables the development of complex systems and services that rely on distributed computing resources. With computational offloading, smart devices become integral components of a larger computational ecosystem, seamlessly integrating with remote resources to deliver

enhanced functionality and an immersive user experience.

1.1 Mobile cloud computing

One notable approach that tackles these challenges is known as Mobile Cloud Computing (MCC). In MCC devices may take advantage of sophisticated computing resources that are available at dedicated large-scale computing facilities located in the core of the internet. MCC also enables devices to store large volumes of data in a sophisticated way. Researchers such as Dinh et al. (2013), Fernando et al. (2013), Khan et al. (2014), and Rahimi et al. (2014) have extensively explored the potential of MCC, which harnesses the power of shared computational and storage resources located at the core of the network. By leveraging these resources, mobile devices are empowered to execute more sophisticated and resource-intensive applications through various computation offloading techniques.

The proliferation of mobile devices within communication networks presents a new set of challenges for MCC, particularly in terms of latency and security requirements. The very nature of MCC's operation, which involves sharing limited bandwidth among multiple devices, coupled with the significant propagation distance between mobile devices and the centralized cloud center, impedes the seamless support of applications with stringent low-latency and real-time execution demands. This issue has been highlighted in studies conducted by Dinh et al. (2013), Mao et al. (2017b), and Abbas et al. (2018).

The limited bandwidth available in the network poses a significant hurdle to the efficient functioning of MCC. With an increasing number of mobile devices

competing for bandwidth, ensuring fair and equitable distribution becomes a considerable challenge. As a result, the demands of applications requiring real-time responsiveness and low-latency execution may be compromised. The long propagation distance between mobile devices and the cloud center along with the required routing operations further compounds this issue, leading to increased latency and potential delays in executing tasks.

Despite these challenges, ongoing research aims to address the limitations of MCC and improve its capabilities. Efforts are being made to optimize network architectures, develop efficient resource allocation algorithms, and enhance communication protocols to minimize latency and improve overall performance. One particularly interesting advancement is that of edge computing, where computation facilities are provided closer to the edge of the core network. As we will explain in the next section, doing so offers the potential to reduce reliance on distant cloud centers and reduce latency concerns.

1.2 Mobile edge computing

To address some of the challenges encountered by Mobile Cloud Computing (MCC), a paradigm called Mobile Edge Computing (MEC) has emerged as a complementary solution; e.g. Pham et al. (2020). MEC operates by provisioning computation and memory resources at the access points and base stations of the network, rather than relying solely on a centralized cloud infrastructure in the network core (Patel et al. (2014)). This decentralization enables the shifting of computation and storage functions closer to the "edge" of the communication network, as emphasized in works by Mach and Becvar (2017), Mao et al. (2017b), and Abbas et al. (2018).

By leveraging the proximity of computation resources to mobile devices, MEC offers several advantages over traditional MCC approaches. First and foremost, the direct connection between mobile devices and the computation resources located at the network edges reduces propagation time and minimizes networking delays. This proximity enhances the responsiveness and real-time capabilities of applications, making MEC particularly suitable for use cases that demand low-latency and time-sensitive processing. Moreover, MEC brings improvements in terms of network efficiency and bandwidth utilization. By offloading computational tasks to the edge servers located closer to the devices, MEC reduces the need for data to traverse long distances to reach a centralized cloud. This localization of computation not only minimizes the burden on network bandwidth but also alleviates congestion and improves the overall network performance.

The adoption of MEC has sparked significant research and development efforts aimed at optimizing its implementation. Various architectural enhancements, efficient resource management mechanisms, and communication protocols have been proposed to fully exploit the benefits of MEC; e.g., Kumar et al. (2013) and Mao et al. (2017a).

1.3 Types of computation offloading

As we discussed in the previous sections, computation offloading enables each device to execute some portion of its tasks in a much larger processing unit. While this approach is effective, the process of deciding whether each device should offload its task or not, and if it does offload what portion of the task should be offloaded, can be quite challenging. since there are many constraints that influence the

decision. To address that challenge we can classify approaches to computation offloading into two main strategies, partial and binary offloading.

Partial offloading is a strategy in which only a portion of a computational task is offloaded to a remote server, while the remaining part is executed locally on the mobile device. This approach allows for a balanced distribution of computational workload and reduces the communication overhead associated with offloading large volumes of data. By intelligently partitioning the task, partial offloading offers the flexibility to exploit the available computing resources both locally and remotely, resulting in improved performance and energy efficiency. However, it is limited to computational tasks that can be appropriately partitioned.

Binary offloading is a different approach that involves the decision offload the whole of a computational task to a remote server, or to complete the whole task locally. Binary offloading is particularly suitable for tasks that are tightly coupled and cannot be appropriately partitioned, and for computationally intensive tasks that significantly exceed the capabilities of the mobile device. In this thesis we will focus on binary offloading binary.

1.4 Developing effective computation offloading systems

The development of an effective computation offloading system requires consideration of a variety of aspects, including computing resource allocation among the offloading devices, communication resource allocation, task partitioning, dynamic offloading decision-making, energy efficiency, security and privacy concerns,

quality of service (QoS), load balancing, and latency optimization. Each of these aspects presents unique challenges and requires careful consideration:

- Computing resource allocation: This process involves determining the optimal allocation of computational tasks between the mobile devices and the remote server or cloud infrastructure. It involves considering factors such as task characteristics, network conditions, and device capabilities.

- Communication resource allocation: This process seeks to allocate the communication resources in a way that facilitates offloading. This includes the cost of transmitting data between the mobile device and the remote server, as well as the impact of network latency, bandwidth limitations, and the management of interference from other devices.

- Task partitioning: This process involves identifying the appropriate portions of the tasks to offload from the mobile device, considering their computational complexity, data dependencies, and potential benefits of offloading.

- Dynamic offloading decision making: This process involves developing algorithms or decision-making mechanisms to dynamically determine when and which tasks should be offloaded based on real-time conditions, such as network availability, device resources, and application requirements.

- Energy efficiency: A goal of many offloading systems is to optimize the energy consumption in order to prolong the mobile device's battery life. This involves considering the energy costs associated with communication, computation, and the overall offloading process.

- Security and privacy concerns: Another aspect of offloading system is addressing security and privacy challenges related to computation offloading. This includes protecting sensitive data during transmission, ensuring secure authentication, and mitigating risks associated with remote processing

- Quality of service (QoS): One of the important goals of the communication system is to maintain acceptable levels of performance and user experience during computation offloading. This includes considerations such as response time, throughput, reliability, and meeting application-specific QoS requirements.

- Load balancing: Incorporating load balancing algorithms helps to effectively distribute the computational load efficiently across multiple remote servers or cloud resources to ensure fair resource utilization, minimize response time, and avoid bottlenecks or overloaded nodes.

- Latency reduction: This process involves reducing the latency introduced by offloading tasks to remote servers, which can affect real-time and low-latency applications. This includes optimizing the network communication, task scheduling, and processing time at the remote server to reduce overall latency.

Table 1.1, which is adapted from Kumar et al. (2013), summarises some of the differences between MCC and MEC systems in terms of the computing servers, proximity to end users, typical latency, and more. In comparison to MCC, MEC offers several advantages, including reduced latency, energy conservation for mobile devices, support for context-aware computing, and improved privacy and security

for mobile applications. In this thesis, we specifically target latency-sensitive tasks, making Mobile Edge Computing (MEC) the appropriate paradigm, as distinct from Mobile Cloud Computing (MCC).

Table 1.1: Comparison of MEC and MCC, adapted from Kumar et al. (2013)

Category	**MEC (Mobile Edge Computing)**	**MCC (Mobile Cloud Computing)**
Server hardware	Small-scale data centers with moderate resources	Large-scale data centers (each contains a large number of highly-capable servers)
Server location	Co-located with wireless gateways, WiFi routers, and LTE BSs	Installed at dedicated buildings with size comparable to several football fields
Deployment	Densely deployed by telecom operators, MEC vendors, enterprises, and home users. Requires lightweight configuration and planning	Deployed by IT companies such as Google and Amazon at limited locations worldwide. Requires sophisticated configuration and planning
Distance to end users	Relatively short distances (tens to hundreds of meters)	Can span large distances, potentially across country borders
Backhaul usage	Infrequent use, used to alleviate congestion	Frequent use, which can potentially cause congestion
System management	Hierarchical control with centralized and distributed elements	Centralized control
Supportable latency	Less than tens of milliseconds	Larger than 100 milliseconds
Applications	Latency-critical and computation-intensive applications, e.g., AR, automatic driving, and interactive online gaming	Latency-tolerant and computation-intensive applications, e.g., online social networking, and mobile commerce/health/learning

1.5 Constraints on Computing Resource Allocation in MEC Servers

While MEC (Mobile Edge Computing) offers numerous advantages for offloading computational tasks to edge servers, there are certain limitations that need to be considered. One key challenge is the efficient allocation of computing resources within the MEC server infrastructure. This allocation is subject to a number of constraints:

1. **Limited Processing Power**: MEC servers typically have limited processing power compared to cloud data centers. These servers are often equipped with lower-end processors and have restricted computational capabilities. The limited processing power can constrain the types and complexity of tasks that can be efficiently executed at the edge.

2. **Memory and Storage Constraints**: MEC servers also have limited memory and storage capacities compared to their cloud counterparts. This limitation can impact the ability to store and process large volumes of data locally at the edge. Tasks that require significant memory or storage resources may experience performance degradation or even fail to execute entirely within the constrained environment of MEC servers.

3. **Dynamic Workload Variation**: The workload on MEC servers can be highly dynamic and unpredictable. The number and types of computational tasks being offloaded can vary significantly, leading to resource contention and potential performance bottlenecks. MEC servers must efficiently manage

and allocate resources to handle these workload fluctuations while ensuring optimal performance for all offloaded tasks.

4. **Heterogeneous Environments**: MEC deployments often encompass a diverse range of edge devices and servers with varying capabilities and configurations. Managing the computing resource allocation across such heterogeneous environments can be challenging. Optimizing the resource allocation requires considering factors such as device capabilities, network conditions, and task characteristics. It requires efficient load balancing mechanisms and intelligent resource allocation algorithms that can adapt to the dynamic nature of the MEC ecosystem.

5. **Network Latency and Bandwidth**: MEC servers are located at the network edge, closer to the end-users and devices. While this proximity reduces network latency to some extent, it is still a factor that needs to be considered. The limited network bandwidth and potential congestion can impact the performance of offloaded tasks. Applications with strict latency requirements, such as real-time and low-latency applications, may be particularly sensitive to these limitations and may not achieve the desired performance gains through MEC offloading.

In this thesis, our focus centers around a centralized MEC-based system, where the MEC server plays the pivotal role in decision-making for task offloading and resource allocation. In our two main contributions our algorithms inherently address the dynamic workload variation in a heterogeneous environment, in the presence of latency constraints. In the first contribution the emphasis is on a communication

resource allocation algorithm that addresses the bandwidth limitations, whereas the second contribution provides a joint computation–communication resource allocation that also seeks to make the most of the limited processing power of the MEC server.

1.6 Low-Latency Communications

When computation offloading is employed in a real time application, such as autonomous driving, tele-medicine, online gaming, or industrial automation, the results of an offloaded task are required by a given deadline, or latency, and this imposes a latency constraint on the communication and computation phases of offloading. There are several factors that contribute to the challenges in meeting low-latency requirements in communication systems:

1. **Propagation Delay**: The time it takes for signals to travel through the communication medium, such as fiber-optic cables or wireless channels, introduces unavoidable propagation delay. As the distance between communicating entities increases, the delay becomes more significant, hindering the achievement of ultra-low latencies.

2. **Processing Delays**: The time taken by devices to process incoming data, including encoding, decoding, encryption, and decryption, adds to the overall latency. The processing capabilities of the devices and the complexity of the algorithms employed influence the extent of these delays.

3. **Rate Limitation**: Low-latency tasks are also limited by the maximum

transmission rate supported by the communication system. The rate limitation can be determined by factors such as latency, the available bandwidth, modulation schemes, and channel conditions. When the required transmission rate exceeds the system's capacity, the communication fails, and any attempts to perform re-transmission of (part of) the message typically results in a significant increase in the time delay .

In this thesis we will focus on the rate limitations. In particular, in our first contribution we will employ a recent approximation of the fundamental rate limit for communication over a finite block length (Polyanskiy et al. 2010), rather than the classical asymptotic-block-length characterization developed by Shannon.

1.7 Contribution of thesis

In this thesis we contribute to the development of computation offloading in mobile edge computing systems by developing offloading algorithms for two related scenarios. In both scenarios each device has a computational task that must be completed by a device-specific deadline. The tasks are modelled as being indivisible, and hence a binary decision must be made as to whether the task is offloaded or is completed locally. This decision is made at the access point, along with the allocation of computation and communication resources among the offloading devices. The offloading devices communicate using the time division multiple access (TDMA) scheme over a single-antenna quasi-static channel that is known to the access point, and each device has its own power constraint. The time that each offloading device has to communicate the description of its task is determined by the deadline by which the device requires the result, the time that it takes to

complete the task at the access point, and the time that it takes to return the result to the device. We will consider problems for which the full description is needed before computation can begin, and for which the results become available once the entire task has been completed. The time that each device has to communicate the description of its task to the access point overlaps with that of the other devices and the access point must determine the appropriate transmission order and allocate transmission times (and powers and rates) to each device. This communication resource allocation is done jointly with the computation resource allocation and the decision as to which devices will offload the tasks. The objective of the joint allocation is to minimize a weighted sum of the energy expended by each device in either offloading its task or completing the task locally. The weight can be interpreted as the (relative) price of the energy at each device.

As will be explained in more detail in the next section, in Chapter2 we will consider a scenario in which each device has (statically) reserved computational resources that guarantee a maximum computation time regardless of the number of offloading devices. As a result, the problem reduces to a problem of joint allocating communication resources and making the offloading decisions. A feature of our approach is that our algorithm explicitly incorporates the fact that the communication block length is finite and hence conventional asymptotic measures of the achievable rate must be modified. Our approach also deals directly with the fact that the communication block length is inherently integer-valued.

In Chapter 3 we consider a scenario in which the computation resources are allocated dynamically, in respect to the requests of the devices, and this allocation is done jointly with the communication resources and the making of the offloading

decisions. This is explained in more detailed in Section 1.7.2.

1.7.1 Chapter 2

This chapter seeks insight into the impact that the requirement to communicate over finite block lengths has on computation offloading schemes, and in particular those with low latencies. The goal is to develop an effective algorithm for finding a good algorithm for selecting the offloading devices and the allocation of communication resources to the offloading devices. Conventionally, communication resource allocation methods have relied on asymptotic limits, which may not be sufficiently accurate when dealing with finite block lengths. Finite block lengths introduce stringent time constraints, requiring a more precise approach to resource allocation. To address this challenge, we consider not only the impact of the low latency constraint, but also the fact that where block lengths have integer values. We propose an efficient algorithm for energy allocation that takes into account the fundamental rate limits for finite block lengths in TDMA-based multi-user computation offloading scenarios.

Our algorithm incorporates new expressions for the achievable rate over finite block lengths that complement Shannon's classical results for the asymptotic case. We also incorporate a relaxation–rounding approach to address the integer nature of the block length. The relaxation–rounding approach is based on a customized incremental rounding scheme specifically designed for the communication model that we employ. Notably, our algorithm is carefully constructed so that it ensures that rounding a feasible solution to the relaxed problem always produces a feasible integer block length. By leveraging a closed-form approximation of the

transmission powers, we simplify the optimization problem into a sequence of convex approximation problems focused solely on the transmission rates.

This chapter is based on a paper that will soon be submitted to the IEEE Transactions on Signal Processing, authored by M. Amin Manouchehrpour, Harvinder Lehal, Mahsa Salmani, and Tim Davidson. Mahsa Salmani, a former graduate McMaster student, developed some initial ideas on binary offloading for the finite block length regime as part of her PhD thesis, but those ideas were somewhat naive and over looked a number of important issues. The concept for this chapter and the paper was inspired by her work. Harvinder Lehal, a former McMaster undergraduate student, performed some preliminary numerical experiments. I contributed to the reformatting of the formulation and completed the numeric analysis. All of us have been supervised by Dr. Davidson throughout this research endeavor.

1.7.2 Chapter 3

In the context of multi-user computation offloading, efficient allocation of computation resources is a crucial factor that significantly impacts system performance. In Chapter 2, the focus is primarily on the allocation of communication resources among the offloading devices for a system with reserved computation resources. However, in addition to communication resources, the allocation of computation resources at the edge server plays a vital role in optimizing offloading efficiency and meeting the computational requirements of the devices.

Computation resource allocation at the edge server entails determining the optimal distribution of computational tasks among the available resources. This

involves considering factors such as task characteristics, computational complexity, and device capabilities. In Chapter 2, the computing resources were reserved for the task of each device. In Chapter 3, the computing resources are allocated jointly with the communication resources.

Our approach to joint computation–communication resource allocation enables flexible allocation of computation resources while ensuring timely execution of tasks. We utilize a simplified modification Shannon's fundamental rate limit to guide our allocation decisions. By leveraging effective heuristics for the computation resource allocation, our design approach simplifies the joint computation–communication resource allocation into a set of convex approximation problems. Firstly, we focus on achieving a good computation resource allocation, and then we proceed to optimize the communication resource allocation. This strategy allows us to efficiently manage the resource allocation and enhance the overall performance of the computation offloading system.

1.7.3 Chapter 4

This chapter provides a summary of the key contributions made in this thesis, along with the conclusions drawn from the conducted research. Additionally, it highlights potential avenues for future exploration and development.

Chapter 2

TDMA-Based Multi-User Binary Computation Offloading in the Finite-Block-Length Regime

Abstract

An effective multi-user computation offloading system is contingent on the selection of the offloading devices and the allocation of communication resources to the selected devices. In previous work, that allocation has been performed using the classical (asymptotic) characterizations of the fundamental rate limits. However, each device will require the results from its task within a certain time limit, and that imposes a bound on the block length. In this chapter, we develop efficient algorithms for offloading decision making and communication resource allocation that incorporate characterizations of the fundamental rate limits on finite-block-length

communication, for a K-device binary computational offloading system that employs time-division multiple access (TDMA). The algorithms involve a relaxation-rounding approach that is constructed around a customized incremental rounding scheme for the transmission block lengths. A feature of our approach is that the relaxation is tightened in such a way that rounding a feasible solution to the relaxed problem is guaranteed to generate a feasible integer block length. Furthermore, by exploiting a closed-form approximation of the transmission powers, our design approach reduces to successively solving convex approximation problems over the transmission rates alone. The proposed algorithms almost completely bridge the performance gaps between a variety of ad-hoc schemes and solutions obtained by exhaustive search.

2.1 Introduction

The computational capabilities of small-scale devices can be greatly enhanced by offering them the opportunity to offload computational tasks to computing infrastructure at the network access point; e.g., Kumar and Lu (2010), Kumar et al. (2013), Khan et al. (2014), Barbarossa et al. (2014), Mao et al. (2017a), and Mach and Becvar (2017). However, providing effective access to this Mobile Edge Computing (MEC) infrastructure requires appropriate allocation of the available resources among the devices. In the case of coordinated access that is managed by the access point and computational tasks that are not divisible, the role of the access point is to jointly decide which devices should offload their tasks (while the others complete their tasks locally), and allocate the available resources among the offloading devices. Often, the objective of that joint decision-allocation problem

is to minimize (a weighted sum of) the energy expended by the mobile devices, while ensuring that they receive the results of their computational tasks by a specified deadline; e.g., Sardellitti et al. (2015), Munoz et al. (2015), Salmani and Davidson (2016), You et al. (2017), Wang et al. (2018), Mao et al. (2017b), Chen et al. (2018), Salmani and Davidson (2018), Salmani and Davidson (2020b), and Salmani and Davidson (2020a).

The presence of that deadline means that the rates at which the devices can communicate with the access point play a critical role in the resource allocation problem; e.g., Mao et al. (2017a). In existing work, the achievable rates have been characterized by assuming that the transmission block lengths are long enough that the guidance from classical (asymptotic) information theoretic results is sufficiently accurate; e.g., Sardellitti et al. (2015), Munoz et al. (2015), Salmani and Davidson (2016), You et al. (2017), Wang et al. (2018), Mao et al. (2017b), Chen et al. (2018), Salmani and Davidson (2018), Salmani and Davidson (2020b), and Salmani and Davidson (2020a). However, the fact that each device has a deadline imposes a natural limit on the block length. In this chapter we will develop communication resource allocation algorithms that explicitly incorporate characterizations Polyanskiy et al. (2010) of the achievable rates in the finite-block-length regime.

The particular problem that we address is the joint offloading decision and uplink communication resource allocation problem in a K-device TDMA-based binary offloading system with reserved computing resources in which the total energy expended by the devices is to be minimized. This problem is a "mixed" optimization problem over the binary offloading decisions, the integer transmission

block lengths, the transmission rates (which will turn out to be rational), and the continuous transmission powers. In order to efficiently obtain good solutions to this problem, we will decompose it into an outer tree search for the offloading decisions and an inner optimization problem over the powers, rates and block lengths. We suggest an efficient algorithm for the outer tree search that is a variant of a tailored pruned greedy search algorithm that was developed in Salmani and Davidson (2020b) for a related problem. The key contribution of this chapter is a set of algorithms for the inner communication resource allocation problem that incorporate different approximations Polyanskiy et al. (2010) of the rate limits in the finite-block-length regime. Our algorithms exploit the structure of the inner problem to reduce its dimension. They are based on computationally-efficient successive convex approximation (SCA, e.g., Hong et al. (2016), Sun et al. (2017), and Scutari and Sun (2018)) methods for the transmission rates, closed-form expressions for the transmission powers, and a randomized Raghavan and Tompson (1987) version of a customized incremental rounding technique for the transmission block lengths. A feature of the proposed approach over our earlier work Salmani and Davidson (2017) and some related work on two-device non-orthogonal multiple access Liu et al. (2022) and Yang et al. (2022) is that when we relax the integer constraint on the transmission block lengths, we also restrict other constraints in such a way that the deterministic version of our customized incremental rounding scheme is guaranteed to generate an integer transmission block length that corresponds to a feasible point of the original problem. This "conservative-relaxation" approach[1]

[1]Although the term "conservative relaxation" is somewhat oxymoronic, it enables us to concisely distinguish our approach from generic relaxation, which does not provide any guarantee on the feasibility of a rounded solution.

greatly simplifies the incorporation of the outcomes of our (inner) resource allocation algorithm into the (outer) tree search that selects the devices that will offload their tasks.

The chapter is arranged as follows: In Sec. 2.2 we establish the system model and provide a generic formulation of our problem; cf. (2.9). In Sec. 2.3 we outline the decomposition of the problem into the outer tree search for the subset of devices that should offload their tasks and the inner "complete-offloading" problem (2.10) that allocates the uplink communication resources to the devices in that subset. In Sec. 2.4 we develop our efficient algorithm for the complete-offloading problem. In Sec. 2.5 we embed our (inner) complete-offloading algorithm within the outer tree search for the offloading subset, and in Sec. 2.6 we provide some results that illustrate the performance of our approach.

2.2 System Model

We consider a centrally-managed computation offloading system with K single-antenna devices, and a single-antenna access point with reserved computing resources. The kth device seeks to complete a computational task within its own specified deadline of L_k seconds. The devices' tasks are modeled as being indivisible, and hence the system must decide whether each device should offload its task to the access point or execute the task locally. The access point employs TDMA (with independent decoding of each message).[2] Its goal is to jointly select the offloading devices, and determine the rate, power, and transmission block length

[2]While TDMA has the inherent advantage of being straightforward to implement, it can also be optimal; notably in the case of two devices with equal channel gains Salmani and Davidson (2018).

for each offloading device, so as to minimize a weighted sum of the energies expended by the devices, under the latency constraints on the tasks and the inherent constraints on the achievable rates.

2.2.1 Offloading decisions

Since each task is modelled as being indivisible, for each device we must make a binary offloading decision. Let $\gamma_k \in \{0, 1\}$ denote the decision that is made for device k, with

$$\gamma_k = \begin{cases} 1, & \text{if device } k \text{ is to offload its task,} \\ 0, & \text{if device } k \text{ is to complete its task locally.} \end{cases} \tag{2.1}$$

If $\gamma_k = 0$, device k expends E_{loc_k} units of energy to compute the task locally, which takes t_{loc_k} seconds. For this to be feasible, we must have $t_{\text{loc}_k} \leq L_k$. If $\gamma_k = 1$, device k invests its energy in offloading the B_k bits that describe its task to the access point. We will consider the class of tasks for which the full description of the task must be received before the access point begins execution, and for which the results are sent back once the task has been completed. Therefore, if t_{del_k} denotes the elapsed time over which the description of device k's task is delivered to the access point, t_{exe_k} denotes the execution time at the access point, and t_{DL_k} denotes the time it takes to return the result, for offloading to be feasible, we must satisfy

$$t_{\text{del}_k} + t_{\text{exe}_k} + t_{\text{DL}_k} \leq L_k. \tag{2.2}$$

Since the result to be communicated back to the device often has a significantly shorter description than the task itself, and since each device's task may be completed at a different time, we will assume that t_{DL_k} is a (different) constant for each device. (Similar models are adopted in most of the existing literature, e.g., Sardellitti et al. (2015), Wang et al. (2018), You et al. (2017), Salmani and Davidson (2018), Salmani and Davidson (2020b), and Salmani and Davidson (2020a).) Furthermore, given our focus on the communication resource allocation, like much of existing work (e.g., Wang et al. (2018), You et al. (2017), Salmani and Davidson (2018), Salmani and Davidson (2020b), and Salmani and Davidson (2020a)), we will consider a scenario in which sufficient computation resources have been reserved at the access point for device k's task to be completed in t_{exe_k} seconds independent of the computational load imposed by other offloading devices. One such scenario is that in which the access point has at least K processors and each arriving task is assigned to its own processor.

2.2.2 Communication model

We adopt a standard communication model in which each device performs narrowband coherent communication over a quasi-static discrete-time-equivalent channel, corresponding to the symbol interval T_s. We characterize each channel using $\alpha_k = |h_k|^2/\sigma^2$, where h_k is the (complex-valued) channel gain, and σ^2 is the variance of the additive white Gaussian noise at the access point. (Since we consider coherent communication, each α_k will be known by the access point.)

Given that we employ TDMA, the devices will transmit in different time slots, and for device k we have that $t_{\mathrm{del}_k} = t_{\mathrm{w}_k} + t_{\mathrm{UL}_k}$, where t_{w_k} denotes the time that

device k has to wait for access to the channel and t_{UL_k} denotes the time that device k is transmitting to the access point. Given our offloading model, we can rewrite the offloading latency constraint in (2.2) as $t_{\mathrm{del}_k}/T_s \leq \tilde{L}_k$, where

$$\tilde{L}_k = (L_k - t_{\mathrm{exe}_k} - t_{\mathrm{DL}_k})/T_s \tag{2.3}$$

is the communication deadline in channel uses. Given the structure of TDMA transmission, an optimal transmission order for the devices is in increasing order of $\tilde{L}_k$, and without loss of generality we will index the devices such that

$$\tilde{L}_1 \leq \tilde{L}_2 \leq \cdots \leq \tilde{L}_K \tag{2.4}$$

and the devices selected for offloading will transmit in that order. That is, offloading device k will transmit in the $\nu(k)$th time slot, where $\nu(k) = \sum_{j=1}^{k} \gamma_j$.

For each offloading device k, the transmission block length, $\tau_{\nu(k)}$, and the data rate, R_k, both of which are expressed in terms of channel uses, must be sufficient to communicate the description of the task to the access point; i.e.,

$$R_k \tau_{\nu(k)} = B_k. \tag{2.5}$$

Offloading device k's transmission time, in seconds, is $t_{\mathrm{UL}_k} = T_s \tau_{\nu(k)}$, and its waiting time is $t_{\mathrm{w}_k} \geq T_s \sum_{i=1}^{\nu(k)-1} \tau_i$. If (offloading) device k uses a transmission power (in units per channel use) of P_k, the energy required to offload the task is $\tau_{\nu(k)} P_k$. The maximum allowable transmission power is $\bar{P}_k$.

While the rate of (offloading) device k must satisfy (2.5), there is also a fundamental limit on the rate at which reliable communication can be achieved with a transmission power P_k. In previous work, the classical asymptotic limit of $R_k \leq \log_2(1 + \alpha_k P_k)$ has been employed. However, that expression ignores the fact that in offloading problems we have latency constraints, and hence the block length $\tau_{\nu(k)}$ is inherently finite. Therefore, we will constrain R_k so that Polyanskiy et al. (2010)

$$R_k \leq \psi_{\epsilon_k}(P_k, \tau_{\nu(k)}), \tag{2.6}$$

where the function $\psi_{\epsilon_k}(P_k, \tau_{\nu(k)})$ characterizes the fundamental limit on the transmission rate (in bits-per-complex-valued-channel-use) for device k, when that device is transmitting with power P_k over $\tau_{\nu(k)}$ uses of the channel, with a prespecified probability of error ϵ_k. (In the finite-block-length regime we must accept a finite probability of error Polyanskiy et al. (2010).)

Determining $\psi_\epsilon(P, \tau)$ explicitly is a difficult problem Polyanskiy et al. (2010). (For notational simplicity, we will temporarily drop the dependence on k.) Therefore, a well-chosen approximation is required. One pragmatic way to approximate $\psi_\epsilon(P, \tau)$ is to use the notion of an SNR-Gap (e.g., Cioffi et al. (1995) and Starr et al. (1999)), and to pick a constant Γ so that the approximation

$$\psi_\epsilon(P, \tau) \approx \hat{\psi}^{(\text{gap})}(P; \Gamma) \ = \log_2\Big(1 + \alpha P/\Gamma\Big) \tag{2.7}$$

is appropriate for the values of P, τ and R that are expected as outcomes of the resource allocation, and the chosen value of ϵ. However, the fact that P, τ and R are design variables in the resource allocation problem (see (2.9) below)

means that it is difficult to make appropriate choices for Γ. More recently, the principles of communication over finite block lengths have been re-examined, and several accurate (and insightful) approximations are now available Polyanskiy et al. (2010); see also Tan and Tomamichel (2015) and Durisi et al. (2016). In particular, the "normal approximation" of $\psi_\epsilon(P,\tau)$ is

$$\hat{\psi}_\epsilon^{(\text{norm})}(P,\tau) = \log_2(1+\alpha P) - \sqrt{\tfrac{V_c(P)}{\tau}}Q^{-1}(\epsilon) + \tfrac{\log_2(2\tau)}{2\tau}, \tag{2.8}$$

where $V_c(P) = \frac{\alpha P(\alpha P+2)}{(\alpha P+1)^2}\Big(\log_2(e)\Big)^2$ is the dispersion of our channel model, and $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}}e^{-t^2/2}dt$. Codes that approach the performance characterized by this approximation of $\psi_\epsilon(P,\tau)$ are beginning to emerge Piao et al. (2020). Our algorithms will be developed for $\hat{\psi}_\epsilon^{(\text{norm})}(P,\tau)$. However, as we will show in Secs 2.4.5 and 2.4.6, the algorithms can be simplified to some degree if we replace $\hat{\psi}_\epsilon^{(\text{norm})}(P,\tau)$ by its first two terms, which we denote by $\hat{\psi}_\epsilon^{(2)}(P,\tau)$, or if we choose an appropriate SNR-Gap approximation. (To facilitate comparisons between the approximations in (2.7) and (2.8), we observe that $\hat{\psi}^{(\text{gap})}(P;\Gamma) = \log(1+\alpha P) - \log\Big(\frac{\Gamma}{1+(\Gamma-1)/(1+\alpha P)}\Big)$.)

2.2.3 General problem formulation

Using the notation $\{\cdot\}$ to denote a set of variables with the index and its range defined implicitly, the design variables in our system are: the decisions as to whether or not each device offloads its task $\{\gamma_k \in \{0,1\}\}$, the (integer-valued) transmission block lengths (in channel uses) $\{\tau_i \in \mathbb{Z}\}$, and the data rates and transmission powers (per channel use) of each offloading device $\{R_k \in \mathbb{R}\}$, $\{P_k \in \mathbb{R}\}$, respectively. With these definitions in place, if we let $\mathcal{S} = \{1,2,\ldots,K\}$ denote the set of all K devices, indexed according to (2.4), and if we let λ_k denote the price per unit of

energy at device k, the problem of minimizing the cost of the energy expended by the devices while completing their tasks within the specified latencies is

$$\min_{\substack{\{\gamma_k\in\{0,1\}\},\{\tau_i\in\mathbb{Z}\},\\ \{R_k\},\{P_k\}}} \quad \sum_k\Big(\gamma_k\lambda_k\tau_{\nu(k)}P_k + (1-\gamma_k)\lambda_k E_{\mathrm{loc}_k}\Big) \tag{2.9a}$$

$$\text{s.t.} \quad \tau_i \geq 0, \tag{2.9b}$$

$$\gamma_k \textstyle\sum_{i=1}^{\nu(k)} \tau_i \leq \tilde{L}_k, \tag{2.9c}$$

$$0 \leq R_k \leq \gamma_k \hat{\psi}_{\epsilon_k}(P_k, \tau_{\nu(k)}), \tag{2.9d}$$

$$0 \leq P_k \leq \gamma_k \bar{P}_k, \tag{2.9e}$$

$$R_k\tau_{\nu(k)} = \gamma_k B_k, \tag{2.9f}$$

$$(1-\gamma_k)t_{\mathrm{loc}_k} \leq L_k, \tag{2.9g}$$

where we have left it implicit that the constraints in (2.9c)–(2.9g) apply for all $k \in \mathcal{S}$ and that the summations in the objective are over all $k \in \mathcal{S}$. The cost of the energy of the devices that compute locally is captured by the second term in the objective, and the latency constraint on these devices is in (2.9g). The first term in the objective and the remaining constraints characterize the offloading devices, with (2.9f) ensuring that the complete description of the task is transmitted, (2.9e) constraining the transmission power of each device, (2.9d) constraining the transmission rate, where $\hat{\psi}_{\epsilon_k}(P_k, \tau_{\nu(k)})$ represents the chosen approximation of the fundamental rate limit, and (2.9c) ensuring that the access point receives the description of the task by the communication deadline $\tilde{L}_k$.

2.3 Architecture of the algorithm

To establish the structural form of the algorithms that we will propose for efficiently generating good solutions to (2.9), we observe that (2.9) is a "mixed" optimization problem with binary, integer and continuous variables. To tackle the combinatorial structure that is imposed by the binary offloading decisions, we will decompose it into an inner problem of finding the uplink communication resource allocation that attains the minimum energy cost for a given set of offloading decisions—a problem that we call "complete offloading"—and the corresponding outer problem of searching over all the possibilities for the set of offloading decisions. As observed in Salmani and Davidson (2020b) for a related problem, the outer problem can be formulated as a tree-search problem, with a complete-offloading problem for a given set of offloading devices being solved at each node of the tree. In particular, in Salmani and Davidson (2020b) a tailored pruned greedy tree-search algorithm was developed for scenarios in which the rate limits are taken from the asymptotic-block-length regime, and there is no constraint on the maximum transmission power. That algorithm was shown to provide better solutions at lower computational cost than a "relaxation-rounding" approach to optimizing over the offloading decisions. Accordingly, in this chapter we will employ a refined version of the pruned greedy tree-search technique in Salmani and Davidson (2020b, Alg. 2) to find good offloading decisions. Therefore, what remains is to develop an efficient algorithm for the (inner) complete-offloading problem that must be solved at each node visited in the tree search.

To formulate the complete-offloading problem, let $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ denote the subset of devices that have been selected to offload their tasks; i.e., $\gamma_k = 1 \, \forall k \in \tilde{\mathcal{S}}$,

$\gamma_k = 0 \, \forall k \in \mathcal{S} \setminus \tilde{\mathcal{S}}$ and $|\tilde{\mathcal{S}}| = \tilde{K}$. Furthermore, let $\tilde{S}'$ denote a re-indexing of $\tilde{S}$ that retains the ordering in (2.4), but indexes the devices from 1 to $\tilde{K}$. (With that re-indexing in place, $\nu(k) = k$.) Substituting the offloading decisions into (2.9), along with the re-indexing of the devices, the complete-offloading problem becomes

$$\min_{\{R_k\},\{P_k\},\{\tau_k \in \mathbb{Z}\}} \quad \sum_{k=1}^{\tilde{K}} \lambda_k \tau_k P_k \tag{2.10a}$$

$$\text{s.t.} \quad \tau_k > 0, \tag{2.10b}$$

$$\sum_{i=1}^{k} \tau_i \leq \tilde{L}_k, \tag{2.10c}$$

$$0 < R_k \leq \hat{\psi}_{\epsilon_k}(P_k, \tau_k), \tag{2.10d}$$

$$0 < P_k \leq \bar{P}_k, \tag{2.10e}$$

$$R_k \tau_k = B_k, \tag{2.10f}$$

where the constraints are imposed for all $k \in \tilde{\mathcal{S}}'$. The complete-offloading problem itself is difficult to solve, due to the integer constraints on $\{\tau_k\}$, and the structure of each $\hat{\psi}_{\epsilon_k}(P_k, \tau_k)$. The key contribution of this chapter is the algorithm developed in Sec. 2.4 for efficiently generating good solutions to (2.10). In the asymptotic-block-length regime, it is possible to reduce (2.10) to an optimization problem over the rates alone; see App. 2.B. One of our goals will be to do the same in the finite-block-length regime.

2.4 Efficient algorithms for (2.10)

The basic structure of our approach to efficiently generating good solutions to the complete-offloading problem in (2.10) consists of four steps: (i) relax the integer

constraints on $\{\tau_k\}$ and, according to (2.10f), set $\tau_k = B_k/R_k$; (ii) efficiently generate a good solution for the remaining optimization problem over the powers and rates; (iii) use a rounding technique on the optimized values of $\{B_k/R_k\}$ to generate good integer values for $\{\tau_k\}$; and (iv) obtain the corresponding rates using (2.10f), and subsequently obtain optimized powers using the upper bound on each R_k in (2.10d). While this basic structure resembles a conventional relaxation-rounding approach, each step is tailored for the particular features of the complete-offloading problem in (2.10). In particular, in order to efficiently generate good integer values for $\{\tau_k\}$, in Sec. 2.4.1 we develop an incremental rounding scheme in which the rounding of the transmission block length for device k exploits the residual from the rounding of the transmission block lengths for devices $1, 2, \ldots, (k-1)$. Furthermore, following the relaxation in step (i) we tighten certain other constraints in the remaining problem over the powers and rates in such a way that the deterministic incremental rounding of $\{B_k/R_k\}$ for any feasible solution to the remaining problem is guaranteed to generate a feasible solution to (2.10); see Sec. 2.4.2.

The resulting conservatively-relaxed problem is not jointly convex in the powers and the rates. Indeed, the constraints are not even continuous. Therefore, in Sec 2.4.3 we develop a further conservative approximation that results in continuous constraints, and then we develop an efficient SCA optimization technique that is based on a sequence of closed-form approximations of the powers and only requires optimization over the rates. Finally, in Sec. 2.4.4 we describe a hybrid deterministic/randomized version of the proposed incremental rounding scheme that uses the solution to the conservatively-relaxed problem to generate good solutions to (2.10).

2.4.1 Relaxation and deterministic incremental rounding

If we relax the integer constraint on the block lengths in (2.10), then we can apply (2.10f) to reduce the complete-offloading problem in (2.10) to an optimization problem over the rates and the powers. We will derive a conservative relaxation in the next section, but for the moment it is enough to let $\{\breve{R}_k\}$ and $\{\breve{P}_k\}$ denote the rates and powers in the relaxed problem. Once we have a solution to the conservatively-relaxed problem, step (iii) of our process constructs values for the block lengths $\{\tau_k\}$ by rounding the values of $\{B_k/\breve{R}_k\}$; cf. (2.10f). The nature of the chosen rounding scheme can have a significant impact on the performance. In order to exploit the fact that the devices transmit sequentially, we will round the values of $\{B_k/\breve{R}_k\}$ incrementally, so that device k can take advantage of the residual arising from the rounding for devices $1, 2, \ldots, (k-1)$. That is, a candidate for τ_1 is constructed by simple rounding, $\tau_1 = \left\lfloor \frac{B_k}{\breve{R}_k} \right\rceil$, and for $k = 2, 3, \ldots, \tilde{K}$, we round the sum of $B_k/\breve{R}_k$ and the residual from the previous roundings,

$$\tau_k = \left\lfloor \frac{B_k}{\breve{R}_k} + \sum_{i=1}^{k-1} \left(\frac{B_i}{\breve{R}_i} - \tau_i \right) \right\rceil. \tag{2.11}$$

We will adopt the convention that an odd multiple of $\frac{1}{2}$ is rounded upwards. While τ_k is clearly a function of $\left\{\breve{R}_1, \breve{R}_2, \ldots, \breve{R}_{k-1}\right\}$, as well as $\breve{R}_k$, we will simplify our notation by keeping that dependence implicit. In particular, we will let ξ_k denote the residual that arises when rounding τ_k. That enables us to re-write (2.11) as $\tau_k = \left\lfloor \frac{B_k}{\breve{R}_k} + \xi_k \right\rceil$, where $\xi_1 = 0$ and for $k = 2, 3, \ldots, \tilde{K}$,

$$\xi_k = \sum_{i=1}^{k-1} \left(\frac{B_i}{\breve{R}_i} - \tau_i \right) = \xi_{k-1} + \frac{B_{k-1}}{\breve{R}_{k-1}} - \tau_{k-1}. \tag{2.12}$$

Note that for $k = 2, 3, \ldots, \tilde{K}$, $-\frac{1}{2} \leq \xi_k < \frac{1}{2}$.

If we round each $B_k/\breve{R}_k$ in the incremental fashion described above, then the rounded value τ_k satisfies

$$\tau_k \in (\tau_{\text{low},k}, \tau_{\text{up},k}] = \left(\tfrac{B_k}{\breve{R}_k} + \xi_k - \tfrac{1}{2},\ \tfrac{B_k}{\breve{R}_k} + \xi_k + \tfrac{1}{2}\right]. \tag{2.13}$$

Once we have rounded the block lengths, the corresponding rates can be computed using (2.10f); i.e., $R_k = B_k/\tau_k$. Those rates satisfy $R_k \in [R_{\text{low},k}, R_{\text{up},k}) = \left[\frac{B_k}{\tau_{\text{up},k}}, \frac{B_k}{\tau_{\text{low},k}}\right)$. That is,

$$R_k \in \left[\frac{\breve{R}_k}{1+(2\xi_k+1)\breve{R}_k/(2B_k)},\ \frac{\breve{R}_k}{1+(2\xi_k-1)\breve{R}_k/(2B_k)}\right). \tag{2.14}$$

The bounds in (2.13) and (2.14) will play a the key role in the formulation of the conservative relaxation in the next section.

2.4.2 A conservative relaxation of (2.10) for $\hat{\psi}_{\epsilon_k}^{(\mathbf{norm})}(P_k, \tau_k)$

In this section we show how the constraints of the relaxation of (2.10) can be tightened in a way that enables us to make feasibility guarantees. In particular, we replace (2.10b) by $\tau_{\text{low},k} > 0$ and, since we are considering the normal approximation in (2.8), we replace (2.10d) by

$$R_{\text{up},k} \leq \log_2(1 + \alpha_k P_k) - \sqrt{\tfrac{V_c(P_k)}{\tau_{\text{low},k}}} Q^{-1}(\epsilon_k) + \tfrac{\log_2(\tau_{\text{low},k})}{\tau_{2\text{up},k}}. \tag{2.15}$$

To ensure that the incrementally-rounded solution satisfies the latency constraint in (2.10c), for each k we impose the constraint $\sum_{i=1}^{k} \left\lfloor \frac{B_i}{\breve{R}_i} + \xi_i \right\rceil \leq \tilde{L}_k$. Since (2.11) can

be written in a recursive form as $\tau_k = \left\lfloor \sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} \right\rceil - \sum_{i=1}^{k-1} \tau_i$, that set of constraints is equivalent to $\left\lfloor \sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} \right\rceil \leq \tilde{L}_k$, and hence is equivalent to $\sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} < \tilde{L}_k + \frac{1}{2}$.

As explained in App. 2.A, the above steps result in an optimization problem in which the constraint on the minimum block length for each device $k \geq 2$ (which is derived from (2.10b)), is a continuous function of $\breve{R}_k$, but a discontinuous function of $\{\breve{R}_i\}_{i=1}^{k-1}$, through ξ_k; see (2.27b) in App. 2.A. Similarly, for $k \geq 2$ the maximum rate constraint on device k that is derived from (2.10d) is a continuous function of $\breve{R}_k$ and $\breve{P}_k$, but a discontinuous function of $\{\breve{R}_i\}_{i=1}^{k-1}$; see (2.27d). The discontinuous (and coupled) nature of these constraints obscures the path towards an efficient algorithm for finding good solutions to the conservatively relaxed problem. In particular, it makes it difficult to manage the behaviour of sequential approximation strategies. Therefore, as explained in App. 2.A, we will use the observation that $\xi_1 = 0$ and that for $k \geq 2$, $\xi_k \in [-1/2, 1/2)$ to construct a further conservative approximation in which all of the constraints are continuous functions of the variables. This further approximation has the additional advantage that it decouples the minimum-block-length constraints from each other, and decouples the maximum-rate constraints from each other, just as these constraints are decoupled in the original formulation in (2.10). The resulting (conservative, relaxed)

optimization problem is

$$\min_{\{\breve{R}_k\},\{\breve{P}_k\}} \sum_{k=1}^{\tilde{K}} \lambda_k B_k \frac{\breve{P}_k}{\breve{R}_k} \tag{2.16a}$$

$$\text{s.t.} \quad \breve{R}_k \leq (1+\delta_{k-1})B_k, \tag{2.16b}$$

$$\sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} < \tilde{L}_k + \tfrac{1}{2}, \tag{2.16c}$$

$$0 < \breve{R}_k \leq \Omega_k(\breve{P}_k, \breve{R}_k), \tag{2.16d}$$

$$0 < \breve{P}_k \leq \bar{P}_k, \tag{2.16e}$$

where δ_i is the Kronecker delta function, the constraints are imposed for all $k \in \tilde{\mathcal{S}}'$, and

$$\Omega_k(\breve{P}_k, \breve{R}_k) = \tfrac{\log_2(1+\alpha_k \breve{P}_k) - b_k(\breve{P}_k, \breve{R}_k)\sqrt{\breve{R}_k}}{a_k(\breve{P}_k, \breve{R}_k)}. \tag{2.17}$$

In (2.17),

$$a_k(\breve{P}_k, \breve{R}_k) = \begin{cases} 1 + y_1(\breve{P}_1, 0) + f_1(\breve{R}_1, 0) & \text{for } k = 1, \\ 1 + y_k(\breve{P}_k, -1/2) + f_k(\breve{R}_k, -1/2) & \text{for } k \geq 2, \end{cases} \tag{2.18}$$

where $y_k(\breve{P}_k, \xi_k)$ and $f_k(\breve{R}_k, \xi_k)$ are defined in (2.29) and (2.30) in App. 2.A, respectively, and

$$b_k(\breve{P}_k, \breve{R}_k) = \begin{cases} g_1(\breve{P}_1, \breve{R}_1, 0) & \text{for } k = 1, \\ g_k(\breve{P}_k, \breve{R}_k, 1/2) & \text{for } k \geq 2, \end{cases} \tag{2.19}$$

where $g_k(\breve{P}_k, \breve{R}_k, \xi_k)$ is defined in (2.31) in App. 2.A. We observe that as the relaxed block length $B_k/\breve{R}_k$ increases, $a_k(\breve{P}_k, \breve{R}_k)$ approaches 1 from above, and $b_k(\breve{P}_k, \breve{R}_k)$ approaches zero, and hence $\Omega(\breve{P}_k, \breve{R}_k)$ in (2.17) approaches the classical fundamental limit on the rate in the asymptotic-block-length case.

2.4.3 Efficiently solving (2.16)

In the asymptotic-block-length regime, the solution of (2.16) can be simplified by finding a closed-form expression for the optimal powers for given rates; cf. (2.32) in App. 2.B. Unfortunately, for the finite-block-length case in (2.16), the presence of $a_k(\breve{P}_k, \breve{R}_k)$ and $b_k(\breve{P}_k, \breve{R}_k)$ in (2.17) has put a closed-form expression for the optimal powers for given rates beyond our reach. However, further analysis of the constraints in (2.16d) will enable us to obtain a simple algorithm to assess the feasibility of (2.16), and an SCA algorithm for its solution.

To begin, we observe that since the constraints in (2.16d) do not couple the powers of the different devices, and since each $\Omega_k(\breve{P}_k, \breve{R}_k)$ is an increasing function of $\breve{P}_k$, if the problem in (2.16) is feasible the upper bounds in (2.16d) are all active at optimality. Therefore, given the rate of device k, $\breve{R}_k$, the optimal value for the power $\breve{P}_k$ is the (unique) value that achieves equality in (2.16d). We will denote that value by

$$\Phi_k(\breve{R}_k) = \mathring{P}_k, \tag{2.20}$$

where $\mathring{P}_k$ is the solution to the one-dimensional, smooth, zero-crossing problem $\breve{R}_k = \Omega_k(\mathring{P}_k, \breve{R}_k)$. By combining (2.16d) and (2.16e), we can show that $\breve{R}_k$ is upper bounded by $\bar{R}_k$, where $\bar{R}_k$ is the solution to the one-dimensional, smooth, zero-crossing problem $\bar{R}_k = \Omega_k(\bar{P}_k, \bar{R}_k)$. Another upper bound on $\breve{R}_k$ arises from

(2.16b). Using these expressions, the problem in (2.16) can be (precisely) transformed into the following optimization problem over the rates alone,

$$\min_{\{\breve{R}_k\}} \quad \sum_{k=1}^{\tilde{K}} \lambda_k \frac{B_k}{\breve{R}k} \Phi_k(\breve{R}_k) \tag{2.21a}$$

$$\text{s.t.} \quad \sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} < \tilde{L}_k + \tfrac{1}{2}, \tag{2.21b}$$

$$0 < \breve{R}_k \leq \acute{R}_k, \tag{2.21c}$$

where $\acute{R}_k = \min\{\bar{R}_k, (1+\delta_{k-1})B_k\}$ and the constraints are imposed for all $k \in \tilde{\mathcal{S}}'$. This problem is analogous to the reduced-dimension problem that arises in the asymptotic-block-length case; see (2.33) in App. 2.B. The feasible set of (2.21) is convex, and the problem is feasible if and only if $\sum_{i=1}^{k} \frac{B_i}{\acute{R}_i} < \tilde{L}_k + \frac{1}{2}$, for all $k \in \tilde{\mathcal{S}}'$. Since the transformation is precise, the same condition is necessary and sufficient for the feasibility of (2.16). Furthermore, if we obtain an optimal solution to (2.21), denoted $\{\breve{R}_k^\star\}$, then the combination of $\{\breve{R}_k^\star\}$ and $\{\breve{P}_k^\star = \Phi_k(\breve{R}_k^\star)\}$ constitutes an optimal solution to (2.16).

While (2.21) generates a simple feasibility test for (2.16), the objective is only implicit, in the sense that we do not have a closed-form expression for the term $\Phi_k(\breve{R}_k)$; cf. (2.20). To develop an efficient algorithm for solving (2.21) we observe that we can rewrite (2.20) to state that $\Phi_k(\breve{R}_k)$ is the value of $\mathring{P}_k$ that satisfies

$$\mathring{P}_k = \frac{2^{\left(a_k(\mathring{P}_k, \breve{R}_k)\breve{R}_k + b_k(\mathring{P}_k, \breve{R}_k)\sqrt{\breve{R}_k}\right)} - 1}{\alpha_k}. \tag{2.22}$$

While this expression remains implicit, it has clear analogies with the corresponding expression in the asymptotic-block-length case; cf. (2.32) in App. 2.B. Furthermore, it suggests the following strategy that generates a closed-form approximation to $\Phi_k(\breve{R}_k)$ and an SCA algorithm.

If we have a previous value for $\breve{R}_k$, say $\breve{R}_k^-$, and a corresponding previous value for $\breve{P}_k$, say $\breve{P}_k^-$, then we can approximate each $a_k(\breve{P}_k, \breve{R}_k)$ in (2.18) by $a_k^- = a_k(\breve{P}_k^-, \breve{R}_k^-)$ and each $b_k(\breve{P}_k, \breve{R}_k)$ in (2.19) by $b_k^- = b_k(\breve{P}_k^-, \breve{R}_k^-)$. If we substitute those approximations into (2.22), we obtain the following closed-form approximation of $\Phi_k(\breve{R}_k)$,

$$\hat{\Phi}_k(\breve{R}_k;\, a_k^-, b_k^-) = \frac{2^{\left(a_k^- \breve{R}_k + b_k^- \sqrt{\breve{R}_k}\right)} - 1}{\alpha_k}. \tag{2.23}$$

Using (2.23) we can approximate the problem in (2.21) by the following problem, which, for later convenience, is expressed in terms of rates $\{\hat{R}_k\}$,

$$\min_{\{\hat{R}_k\}} \quad \textstyle\sum_{k=1}^{\tilde{K}} \lambda_k \frac{B_k}{\hat{R}_k} \hat{\Phi}_k(\hat{R}_k;\, a_k^-, b_k^-) \tag{2.24a}$$

$$\text{s.t.} \quad \text{variants of (2.21b) and (2.21c) with } \breve{R}_k \leftarrow \hat{R}_k. \tag{2.24b}$$

By evaluating the Hessian matrix of the objective, which is diagonal, it can be shown that the objective is strongly convex; see App. 2.D in the Supplementary Material. Furthermore, the feasible set of (2.21) is convex. These observations (and others), enable the problem in (2.24) to be efficiently solved.

Since the problem in (2.24) has the same feasible set as that in (2.21), and that its objective is a convex approximation of the objective in (2.21), we can use an

SCA approach to efficiently generate good solutions to (2.21), and hence to (2.16). In particular, let us presume that we have previous values for the rates, denoted $\{\breve{R}_k^-\}$, that are strictly feasible for (2.21), and previous values for $\{a_k^-\}$ and $\{b_k^-\}$. We then solve the convex problem in (2.24) to obtain the rates $\{\hat{R}_k^\star\}$, with the opportunity to choose $\{\breve{R}_k^-\}$ as a "warm" starting point for solving that problem. Then we update each $\breve{R}_k^-$ by taking a step in the direction of $\hat{R}_k^\star$; e.g., Razaviyayn et al. (2013) and Scutari et al. (2017). That is, $\breve{R}_k^- \leftarrow \breve{R}_k^- + \eta(\hat{R}_k^\star - \breve{R}_k^-)$, where $0 < \eta < 1$ is a step size that decreases with the SCA iterations, but remains persistently exciting. These new rates remain strictly feasible for (2.21), because the previous rates were strictly feasible for (2.21), the problem in (2.24) has the same feasible set as (2.21), and the step size $\eta < 1$. Since $\Phi_k(\breve{R}_k)$ is only defined implicitly (cf. (2.20)), one convenient strategy for updating the step-size is to select a (small) value for $\beta \in (0, 1)$ and an initial value for $\eta \in (0, 1)$ and to update η using $\eta \leftarrow \eta(1 - \beta\eta)$, Scutari et al. (2017). With these new values for the rates, we can compute the corresponding powers, $\{\hat{P}_k^- = \hat{\Phi}_k(\breve{R}_k^-; a_k^-, b_k^-)\}$, and then subsequently update $\{a_k^- = a_k(\hat{P}_k^-, \breve{R}_k^-)\}$ using (2.18), and $\{b_k^- = b_k(\hat{P}_k^-, \breve{R}_k^-)\}$ using (2.19). With the SCA iteration now completed, we test for convergence, and, if needed, commence the next SCA iteration. The resulting algorithm is summarized in Alg. 1. If we denote the output of Alg. 1 by $\{\breve{R}_k^\star\}$, the corresponding powers required to construct a good solution to (2.16) can be computed by solving K one-dimensional smooth zero-crossing problems, namely $\{\breve{P}_k^\star = \Phi_k(\breve{R}_k^\star)\}$; cf. (2.20). However, as we will show in the next subsection, in our (conservative) relaxation-rounding approach, we will only make use of the rates $\{\breve{R}_k^\star\}$.

To initialize the above SCA procedure, we need to determine the feasibility of

(2.21), and to generate a strictly feasible starting point. While we could solve a convex problem that is akin to a "phase one" problem for a primal-barrier interior point method Boyd and Vandenberghe (2004, Sec. 11.4), our simplified feasibility test, described after (2.21) and in Step 1 of Alg. 1, implicitly generates a feasible (but not strictly feasible) point. That enables the application of a simple ad-hoc technique for finding a strictly feasible starting point; see App. 2.E in the Supplementary Material.

Algorithm 1 : An efficient SCA algorithm for (2.21)

Input data: $\tilde{\mathcal{S}}'$, $\{B_k\}$, $\{\tilde{L}_k\}$, $\{\lambda_k\}$, $\{\alpha_k\}$, $\{\bar{P}_k\}$, $\{\epsilon_k\}$, $\beta \in (0,1)$, $\eta^0 \in (0,1)$.
Step 1: Determine feasibility: For each $k \in \tilde{\mathcal{S}}'$, use a one-dimensional line search algorithm to solve $\bar{R}_k = \Omega_k(\bar{P}_k, \bar{R}_k)$; cf. (2.17). Compute all $\acute{R}_k = \min\{\bar{R}_k, (1+\delta_{k-1})B_k\}$. If any component of (2.16c) is violated when each $\breve{R}_k = \acute{R}_k$, then set Flag1 = 1 to denote infeasibility and exit. Otherwise, set Flag1 = 0.
Step 2: Using the method described in App. 2.E in the Supplementary Material, determine a strictly feasible point $\{\breve{R}_k^{(0)}\}$, and set each $\breve{R}_k^- \leftarrow \breve{R}_k^{(0)}$. Set each $\hat{P}_k^- \leftarrow \bar{P}_k$, and $\eta \leftarrow \eta^0$.
Step 3: Calculate each $a_k^- = a_k(\hat{P}_k^-, \breve{R}_k^-)$ using (2.18), and each $b_k^- = b_k(\hat{P}_k^-, \breve{R}_k^-)$ using (2.19).
Step 4: Solve the (strongly) convex problem in (2.24) to find $\{\hat{R}_k^\star\}$.
Step 5: Update each $\breve{R}_k^-$ using $\breve{R}_k^- \leftarrow \breve{R}_k^- + \eta(\hat{R}_k^\star - \breve{R}_k^-)$.
Step 6: Update the step size using $\eta \leftarrow \eta(1-\beta\eta)$.
Step 7: Compute each $\hat{P}_k^- = \hat{\Phi}_k(\breve{R}_k^-; a_k^-, b_k^-)$ using (2.23).
Step 8: Return to Step 3 until convergence
Step 9: Set each $\breve{R}_k^\star = \breve{R}_k^-$.
Output: $\{\breve{R}_k^\star\}$, Flag1.

Although the feasible sets of (2.21) and (2.24) are the same convex set, the fact that $\Phi_k(\breve{R}_k)$ is only defined implicitly means that our problem lies beyond the current class of problems for which convergence guarantees for the SCA approach have been established; e.g.,Razaviyayn et al. (2013) and Scutari et al. (2017). However, in our numerical experience, Alg. 1 has always either converged to a good solution or determined that the problem is infeasible.

2.4.4 Randomized incremental rounding

Although Alg. 1 efficiently generates a good solution $\{\breve{R}_k^\star\}$ to (2.21), or declares that it is infeasible, that solution corresponds to block lengths $B_k/\breve{R}_k^\star$ that are not necessarily integers. Therefore, the values of $B_k/\breve{R}_k^\star$ must be rounded in order to construct a candidate solution for (2.10). By design, when Alg. 1 generates a feasible solution (with Flag1 = 0), the deterministic incremental rounding (see Section 2.4.1) of $\{B_k/\breve{R}_k^\star\}$, followed by recalculation of the rates and the powers, is guaranteed to constitute a feasible solution for (2.10). In particular, as discussed in the opening paragraph of Sec. 2.4, once the incrementally-rounded block lengths, $\{\tau_k\}$, have been obtained, the corresponding rates are $\{R_k^\star = B_k/\tau_k\}$, and the corresponding (optimized) powers are the $\{P_k^\star\}$ that result in the upper bound in (2.10d) holding with equality for these rates. Although that solution is guaranteed to be feasible for (2.10), there may be better solutions that correspond to integer block lengths that are in the neighbourhood of our (deterministically) incrementally-rounded block length. We will search for such solutions by incorporating the principles of randomized rounding Raghavan and Tompson (1987) into the incremental rounding scheme proposed in Sec. 2.4.1.

To round the block length for device $k \in \tilde{\mathcal{S}}'$, we compute the amount by which the quantity to be rounded, namely $B_k/\breve{R}_k^\star + \xi_k$, is above its floor. That is, we compute $p_k = B_k/\breve{R}_k^\star + \xi_k - \lfloor B_k/\breve{R}_k^\star + \xi_k \rfloor$. Then, rather than constructing τ_k by deterministically rounding $B_k/\breve{R}_k^\star + \xi_k$, we instead set τ_k to $\lfloor B_k/\breve{R}_k^\star + \xi_k \rfloor$ with probability $(1 - p_k)$ and set τ_k to $\lceil B_k/\breve{R}_k^\star + \xi_k \rceil$ with probability p_k, where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the rounding down and rounding up operations, respectively. For $k < \tilde{K}$, we then compute ξ_{k+1} using (2.12), and construct the Bernoulli probability

mass function for randomly generating τ_{k+1}. Once all τ_k have been incrementally generated in this way, we assess their feasibility by checking that they satisfy (2.10b) and (2.10c). If so, we calculate the corresponding rates and powers in the same way as in the deterministic case, and then assess their feasibility. If the resulting solution is indeed feasible for (2.10) and has a lower objective value than the best of the previously obtained solutions, the newly generated solution replaces that solution.

We have summarized the proposed hybrid deterministic/randomized incremental rounding scheme in Alg. 2, in which the current best estimate of the optimal value of (2.10) is denoted by $\hat{J}^{\star}_{\text{off}}$ and the solution that achieves that value is $\hat{\mathcal{X}}^{\star}$. Steps 1–5 of Alg. 2 describe the deterministic rounding part, which, by virtue of the conservative relaxed formulation in (2.16), is guaranteed to generate a feasible solution to (2.10) whenever (2.16) is feasible. In Steps 6–10, we search for a better solution to (2.10) by employing the proposed randomized incremental rounding scheme N_{rand} times.

2.4.5 Using $\hat{\psi}^{(2)}_{\epsilon}(P, \tau)$

If we repeat the analysis in Secs 2.4.2 and 2.4.3 for $\hat{\psi}^{(2)}_{\epsilon_k}(P_k, \tau_k)$, which is the first two terms of the normal approximation in (2.8), we obtain algorithms that have the same structure as Algs 1 and 2, but are slightly simpler. The difference is that the functions $f_k(\breve{R}_k, \xi_k)$ that appear in the expressions for $a_k(\breve{P}_k, \breve{R}_k)$ in (2.18) are set to zero.

Algorithm 2 : Incrementally rounding the output of Alg. 1 to obtain a (good) solution to (2.10)

Input data: $\tilde{\mathcal{S}}'$, $\{\breve{R}_k^\star\}$, $\{\lambda_k\}$, $\{B_k\}$, $\{\tilde{L}_k\}$, $\{\alpha_k\}$, $\{\bar{P}_k\}$, $\{\epsilon_k\}$, N_{rand}

Step 1: Calculate all τ_k from $\{B_k/\breve{R}_k^\star\}$ using the deterministic incremental rounding scheme in (2.11).

Step 2: Calculate all $R_k = B_k/\tau_k$.

Step 3: Calculate each P_k using the one-dimensional smooth zero-crossing problem $R_k = \hat{\psi}_{\epsilon_k}(P_k, \tau_k)$; cf. (2.10d).

Step 4: Calc. $\acute{J}_{\text{off}} = \sum_{k\in\tilde{\mathcal{S}}'} \lambda_k \tau_k P_k$ and assemble $\acute{\mathcal{X}} = \{R_k, P_k, \tau_k\}$.

Step 5: Set $\hat{J}_{\text{off}}^\star \leftarrow \acute{J}_{\text{off}}$ and $\hat{\mathcal{X}}^\star \leftarrow \acute{\mathcal{X}}$.

Step 6: Set $n_{\text{rand}} = 0$.

while $n_{\text{rand}} < N_{\text{rand}}$ **do**

  Step 7a: Set $\xi_1 = 0$.

  for $k = 1$ to $\tilde{K} = |\tilde{\mathcal{S}}'|$ **do**

    Step 7b: Calculate $p_k = B_k/\breve{R}_k^\star + \xi_k - \lfloor B_k/\breve{R}_k^\star + \xi_k \rfloor$.

    Step 7c: Randomly generate τ_k using a Bernoulli distribution with $\tau_k = \lfloor B_k/\breve{R}_k^\star + \xi_k \rfloor$ with prob. $(1 - p_k)$ and $\tau_k = \lceil B_k/\breve{R}_k^\star + \xi_k \rceil$ with prob. p_k.

    Step 7d: If $k < \tilde{K}$, calculate $\xi_{k+1} = \xi_k + B_k/\breve{R}_k^\star - \tau_k$.

  end for

  Step 8: Set $n_{\text{rand}} \leftarrow n_{\text{rand}} + 1$.

  if (2.10b) and (2.10c) are satisfied ($\forall k \in \tilde{\mathcal{S}}'$), **then**

    Step 9: Complete Steps 2–5.

    Step 10: If (2.10e) is satisfied ($\forall k \in \tilde{\mathcal{S}}'$), and if $\acute{J}_{\text{off}} < \hat{J}_{\text{off}}^\star$, set $\hat{J}_{\text{off}}^\star \leftarrow \acute{J}_{\text{off}}$ and $\hat{\mathcal{X}}^\star \leftarrow \acute{\mathcal{X}}$.

  end if

end while

Output: $\hat{J}_{\text{off}}^\star$, $\hat{\mathcal{X}}^\star$.

2.4.6 Using $\hat{\psi}^{(\mathbf{gap})}(P;\Gamma)$

That analysis can also be repeated for the SNR-Gap approximation in (2.7). In that case, the problem that is equivalent to (2.16) takes a similar form, with only (2.17) being modified. The modifications are that α_k is replaced by $\grave{\alpha}_k = \alpha_k/\Gamma_k$, $b_k(\breve{P}_k, \breve{R}_k)$ is replaced by $\grave{b}_k(\breve{P}_k, \breve{R}_k) = 0$, and $a_k(\breve{P}_k, \breve{R}_k)$ is replaced by

$$\grave{a}_k(\breve{P}_k, \breve{R}_k) = \begin{cases} 1 + \grave{y}_1(\breve{P}_1, 0) & \text{for } k = 1, \\ 1 + \grave{y}_k(\breve{P}_k, -1/2) & \text{for } k \geq 2, \end{cases} \tag{2.25}$$

where $\grave{y}_k(\breve{P}_k, \xi_k) = \left(\frac{1-2\xi_k}{2B_k}\right) \log_2(1 + \grave{\alpha}_k \breve{P}_k)$.

2.4.7 Independent rounding

Although the incremental rounding scheme in Sec. 2.4.1 was developed specifically for our application, in some settings a simpler scheme in which each block length is rounded independently might be preferred. In that case, (2.11) is replaced by $\tau_k = \left\lfloor \frac{B_k}{\breve{R}_k} \right\rceil$. This rounding scheme corresponds to setting ξ_k in (2.12) to zero for all k. Therefore, analogous "conservative-relaxation" design algorithms for independent rounding can be derived by making that substitution in the analysis in Secs 2.4.1–2.4.6; see App. 2.C. That said, for the independent rounding scheme, the randomized rounding scheme needs to be modified to compensate for the fact that the residuals from rounding the block lengths of devices that transmit earlier are not considered; see App. 2.C.

2.5 An efficient algorithm for binary offloading

In the previous section, we addressed the "inner" complete-offloading problem in which the set of offloading devices is predetermined. In this section, we will complete the algorithm architecture outlined in Sec. 2.3 by tackling the "outer" problem of selecting the set of offloading devices. By applying the "inner–outer" decomposition to original binary offloading problem in (2.9), the outer problem can be written as

$$\min_{\{\gamma_k\}} \quad J^{\star}_{\text{off}}(\{\gamma_k\}) + \textstyle\sum_k (1-\gamma_k)\lambda_k E_{\text{loc}_k} \tag{2.26a}$$

$$\text{s.t.} \quad (1-\gamma_k) t_{\text{loc}_k} \leq L_k, \tag{2.26b}$$

where the range of k is all values in $\mathcal{S}$, and $J^{\star}_{\text{off}}(\{\gamma_k\})$ denotes the optimal solution to the complete-offloading problem in (2.10) for a given set of offloading decisions $\{\gamma_k\}$. The problem in (2.26) is combinatorial, with a search space of 2^K possibilities, but it does admit a tree structure in which the subproblem to be solved at each node is the complete-offloading problem in (2.10) for a given set of offloading devices. That structure provides the opportunity to develop efficient tree-search algorithms that provide good selections for the set of offloading devices, and are of much lower complexity than optimal algorithms, such as branch-and-bound. We propose to use a refined version of the algorithm in Salmani and Davidson (2020b, Alg. 2) that was developed for the asymptotic-block-length case, in the absence of transmission power constraints. That algorithm is a tailored version of a greedy algorithm and enables deterministic pruning of the tree at each step. The

modifications outlined below accommodate the finite-block-length rate characterizations and the fact that the power constraints (cf. (2.9e), (2.10e)) mean that some instances of the complete-offloading problem may be infeasible.

As described in Sec. 2.4, the complete-offloading subproblem that determines $J^{\star}_{\text{off}}(\{\gamma_k\})$ in (2.26) is a difficult mixed-integer non-linear optimization problem. Therefore, in our algorithm we will seek good solutions to the variant of the binary offloading problem in (2.26) in which $J^{\star}_{\text{off}}(\{\gamma_k\})$ is replaced by $\hat{J}^{\star}_{\text{off}}(\{\gamma_k\})$, where $\hat{J}^{\star}_{\text{off}}(\{\gamma_k\})$ is the output of Alg. 2. To describe the algorithm, we retain the use of $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ to denote an unordered set of devices that has been selected for offloading, and let $\tilde{\mathcal{S}}'$ denote the re-indexed version of that set. Within the algorithm, we let $\mathcal{U} \subseteq \mathcal{S}$ denote the set of devices for which an offloading decision has yet to be made. The algorithm is initialized with $\tilde{\mathcal{S}}$ containing all those devices for which the task cannot be completed locally by the specified deadline, and $\mathcal{U}$ containing the remaining devices. If Alg. 1 is unable to find a solution to the resulting complete-offloading problem, the original binary offloading problem in (2.9) is marked as being infeasible. Otherwise, the algorithm proceeds to the tree search for adding devices to the offloading set $\tilde{\mathcal{S}}$. Each layer of the tree search consists of an exploratory step, a deterministic pruning step, and a greedy device selection step that calculates the "best" device to add to the offloading set (if any remain after the pruning step). These steps are outlined in Steps 4, 5 and 7–9 in Alg. 3.

In the exploratory step, for each device in $\mathcal{U}$ we seek to determine the minimum offloading energy cost of the system if that device were to be added to the set of offloading devices, or declare that no feasible solution can be found when it is added, by solving the corresponding instance of (2.10). Since (2.10) is intractable,

we do that efficiently, but slightly conservatively, by applying Algs. 1 and 2. In the deterministic pruning step, we remove from $\mathcal{U}$ all those devices for whom the exploration step (at this node in the tree search) revealed that offloading the task of that device would incur a greater energy cost than local computation.[3] In the greedy device selection step, we select the device for which offloading offers the greatest reduction in the total energy cost of the system.

A feature of Alg. 3 is that we start with the smallest potentially feasible set of offloading devices, and then add devices to the offloading set. That means that in the early stages of the tree search, the problems in the exploratory step have a small number of offloading devices, and hence can be solved relatively quickly. That emphasizes the value of the pruning step, as the branches that are pruned from future exploratory steps correspond to complete-offloading problems that involve a larger number of devices, and hence require more operations to solve. To quantify that value, let $Q^{(i)}$ denote the cardinality of $\mathcal{U}$ at the ith level in the tree search, then the number of instances of Algs 1 and 2 that are performed at that level is $Q^{(i)}$. Alg. 1 is an SCA algorithm and its computational cost can be approximated by the number of successive approximations required for convergence and the cost of solving the convex optimization problem in (2.24). That problem has $(\tilde{K}^{(i)}+1)$ variables, where $\tilde{K}^{(i)}$ is the cardinality of $\tilde{\mathcal{S}}$ at level i in the tree. If that convex optimization problem is tackled using a Newton method, then the computational cost per Newton iteration will be $O\Big((\tilde{K}^{(i)}+1)^3\Big)$. Alg. 2 requires the solution of $(N_{\text{rand}_i}+1)\tilde{K}^{(i)}$ one-dimensional line searches, where N_{rand_i} is the

[3] If we were to solve (2.10) exactly, rather than using Algs. 1 and 2, the removal of these devices would be guaranteed not to degrade the greedy solution; cf. Salmani and Davidson (2020b). While using Algs. 1 and 2 does not provide that theoretical guarantee, the approach works very well in practice; see Sec. 2.6.

Algorithm 3 : An efficient algorithm for a (good) solution to the binary offloading problem in (2.9); see also (2.26).

Input data: $\mathcal{S}$, $\{B_k\}$, $\{L_k\}$, $\{\lambda_k\}$, $\{\alpha_k\}$, $\{\bar{P}_k\}$, $\{\epsilon_k\}$, $\{E_{\mathrm{loc}_k}\}$, $\{t_{\mathrm{loc}_k}\}$.
Step 1: Initialize $\tilde{\mathcal{S}} = \{k \in \mathcal{S} | t_{\mathrm{loc}_k} > L_k\}$.
Step 2:
if $\tilde{\mathcal{S}} \neq \emptyset$ **then** construct the re-indexed set $\tilde{\mathcal{S}}'$, and employ Alg. 1, with outputs $\{\breve{R}_k^\star\}$ and Flag1, to determine if a solution can be found for the corresponding instance of (2.10).

if Flag1 $= 1$ **then** declare the problem in (2.9) to be infeasible by setting Flag3 $= 1$ and $\mathcal{A}^\star = \emptyset$, and then exit.

else Set Flag3 $= 0$ and use Alg. 2, with outputs $\hat{J}^\star_{\mathrm{off}}$ and $\hat{\mathcal{X}}^\star$, to determine a good solution to (2.10). Set $\hat{\mathcal{A}}^\star = \{\tilde{\mathcal{S}}, \hat{\mathcal{X}}^\star\}$.

end if
end if
Step 3: Set $\mathcal{U} = \mathcal{S} \setminus \tilde{\mathcal{S}}$. (Terminate if $\mathcal{U}$ is empty.)
Step 4: Set $\mathcal{V} = \emptyset$.
for each $k \in \mathcal{U}$ **do** Augment $\tilde{\mathcal{S}}$ with $\{k\}$; i.e., $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \{k\}$. Construct the re-indexed set $\tilde{\mathcal{S}}'$, and employ Alg. 1, with outputs $\{\breve{R}_k^\star\}$ and Flag1, to determine if a solution can be found for this instance of (2.10).

if Flag1 $= 1$ **then** add device k to the set of devices to be pruned; i.e., $\mathcal{V} \leftarrow \mathcal{V} \cup \{k\}$

else use Alg. 2, with outputs $\hat{J}^\star_{\mathrm{off},(k)}$ and $\hat{\mathcal{X}}^\star_{(k)}$, to determine a good solution to this instance of (2.10).

if $\hat{J}^\star_{\mathrm{off}} + \lambda_k E_{\mathrm{loc}_k} \leq \hat{J}^\star_{\mathrm{off},(k)}$ **then** add device k to the set of devices to be pruned; i.e., $\mathcal{V} \leftarrow \mathcal{V} \cup \{k\}$

end if

end if
end for
Step 5: Prune the selected devices from the tree; i.e., $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{V}$.
Step 6: If $\mathcal{U} = \emptyset$ **then** terminate the algorithm **end if**
Step 7: Select the "best" device by choosing $k^\star = \arg\max_{k \in \mathcal{U}} (\hat{J}^\star_{\mathrm{off}} + \lambda_k E_{\mathrm{loc}_k} - \hat{J}^\star_{\mathrm{off},(k)})$
Step 8: Update the offloading set and the undecided set; i.e., $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \{k^\star\}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \{k^\star\}$.
Step 9: Update $\hat{J}^\star_{\mathrm{off}} \leftarrow \hat{J}^\star_{\mathrm{off},(k^\star)}$, $\hat{\mathcal{A}}^\star \leftarrow \{\tilde{\mathcal{S}}, \hat{\mathcal{X}}^\star_{(k^\star)}\}$.
Step 10: If $\mathcal{U} = \emptyset$ **then** exit, **else** return to Step 4 **end if**
Output: $\hat{J}^\star = \hat{J}^\star_{\mathrm{off}} + \sum_{k \in \mathcal{S} \setminus \tilde{\mathcal{S}}} \lambda_k E_{\mathrm{loc}_k}$, $\hat{\mathcal{A}}^\star$, Flag3.

number of randomizations employed at the ith level in the tree. Since N_{rand_i} would typically be chosen to be $O(\tilde{K}^{(i)})$ or even $O(1)$, the growth of the computational cost of Alg. 3 will be dominated by the cost of the Newton iterations. By using the fact that $\tilde{K}^{(i)} \leq K - Q^{(i)}$, we can express the computational cost of Alg. 3 as $O\Big(\sum_{i=1}^{K} Q^{(i)}(K - Q^{(i)} + 1)^3\Big)$. By using the Faulhaber formula Conway and Guy (1996) and some rather coarse bounds, it can be shown that a loose upper bound for the argument of this expression is K^5. In practice, the pruning in the algorithm results in a highly efficient implementation.

2.6 Simulation results

We consider a scenario with $K = 4$ devices, each of which communicates over a fading channel model with a path loss exponent of 3.71 and Rayleigh small scale fading at a symbol interval of $T_s = 10^{-6}$s, in the presence of a noise power spectral density of -173 dBm/Hz. The reference gain at a distance of 100m is -20 dB, and, unless otherwise stated, for each realization the devices are positioned at random distances from the access point according to the uniform distribution on [100, 1,000]m. The communication latencies of the devices, $\tilde{L}_k$, are 400, 420, 440, 460 samples respectively, the description lengths of the tasks are $B_k = 1{,}000$ bits, and the power constraints are $\bar{P}_k = 3.3$ µJ/sample, which corresponds to 3.3W. The cost of the energy of each device is normalized to one; i.e., $\lambda_k = 1$.

2.6.1 Complete offloading

For the complete-offloading problem in (2.10) we will consider: (i) Algs 1 and 2 as stated, which employ the normal approximation (NA) $\hat{\psi}_{\epsilon}^{(\text{norm})}(P, \tau)$; (ii) the

variants of that conservative-relaxation approach for the two-term approximation (TTA) $\hat{\psi}_{\epsilon}^{(2)}(P,\tau)$ and the SNR-Gap approximation $\hat{\psi}^{(\mathrm{gap})}(P;\Gamma)$ in Secs 2.4.5 and 2.4.6, respectively; and (iii) the variants for (randomized) independent rounding in Sec. 2.4.7 and App. 2.C. To assess the impact of the proposed conservative-relaxation approach, we will also consider a "baseline" approach that has been heuristically derived from existing results. In the baseline design, the block lengths are simply relaxed, without the additional constraint tightening, and Alg. 1 is replaced by the SNR-Gap variant of the rate allocation problem in the asymptotic-block-length regime; i.e., (2.33) in App. 2.B, but with α replaced by $\dot{\alpha} = \alpha/\Gamma$. We round down each relaxed block length so that the latency constraint is satisfied, and the corresponding power allocation is performed using the SNR-Gap variant of the asymptotic expression in (2.32). Since the baseline design does not take a conservative-relaxation approach, there is no guarantee that that power allocation will satisfy the power constraints in (2.10e).

The performance of the complete-offloading systems will be evaluated in terms of the trade-off between (cost of the) energy expended by the devices, and the probability of error. By probability of error we mean the normal approximation of the probability of information outage, which is (the normal approximation of) a lower bound on the probability of error of any practical code. More specifically, for a given offloading device with block length τ, rate R, and power P, we will determine the probability of error as the value of ϵ that solves $R = \hat{\psi}_{\epsilon}^{(\mathrm{norm})}(P,\tau)$. (This value can be determined using a simple bisection search.) In evaluating the performance of the designs in this way, we observe that the NA designs provide explicit control over the probability of error for each device when the problem

is formulated. In contrast, the SNR-Gap designs only involve a specification of the SNR-Gap at the time of formulation. Although the probability of error can be varied by changing the gap, with a larger gap resulting in lower probability of error (and greater energy usage), the actual probability of error can only be determined after the design (with integer block lengths) is complete. Since the probability of error may be different for each offloading device, we will consider the maximum probability over the devices. For TTA-based designs there is some control over the probability of error, especially if we know that the block lengths will be large. (The third term in the normal approximation in (2.8) is a decreasing function of the block length.) However, we still cannot precisely specify the probability of error in the problem formulation stage.

Rounding schemes

In our first experiment, we consider the impact of the choice of the rounding scheme. In Fig. 2.1 we plot the trade-offs between the probability of error and the total energy for NA-based designs and both deterministic and randomized versions of the incremental and independent rounding schemes in Sec. 2.4.1, and in Sec. 2.4.7 and App. 2.C, respectively. The curves are obtained by averaging over the trade-off curves for each of the 457 out of 1,000 channel realizations for which the baseline scheme generates a feasible solution for all rounding schemes when the target probability of error is 10^{-5}; see Fig. 2.3. In the case of deterministic rounding ($N_{\text{rand}} = 0$), the proposed incremental rounding scheme dominates independent rounding (in the Pareto sense). In fact, the deterministic incremental rounding scheme yields a trade-off that is essentially the same as that achieved by an "extensive-search" approach that examines each combination of block lengths

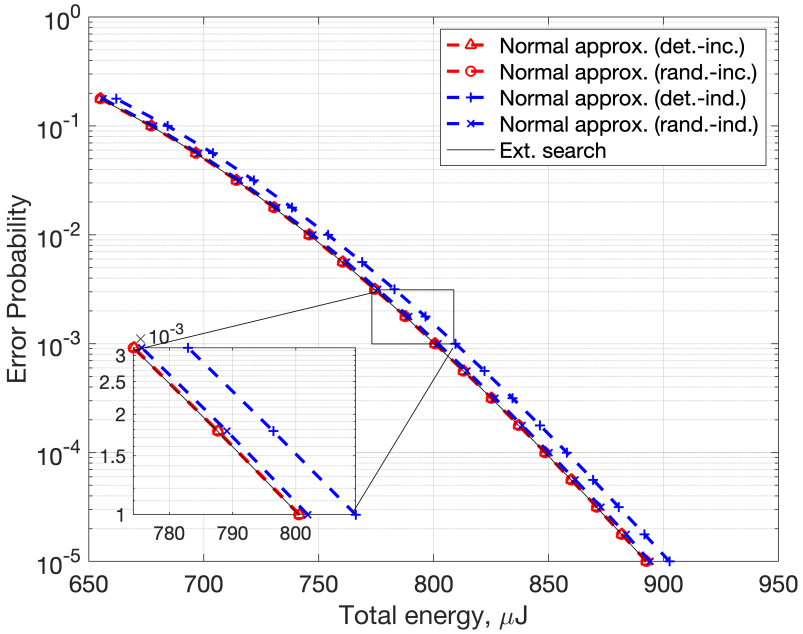


FIGURE 2.1: Error probability versus energy trade-offs achieved by the proposed design method with the normal approximation and different rounding schemes.

within ± 3 integers of the relaxed solution.[4] The randomized scheme in Alg. 2 with $N_{\text{rand}} = 2K$ essentially closes the remaining gap. For the case of independent rounding, $2K$ randomized roundings to the nearest neighbours ($w_k = 0$ in App. D) provides a tangible improvement over deterministic rounding, but the deterministic incremental rounding scheme retains a significant advantage.

Based on these observations, in our subsequent experiments we will focus on the scheme in Alg. 2 with hybrid deterministic randomized incremental rounding with $2K$ random trials.

[4]There are $6^4 = 1,296$ such combinations. For each combination that satisfies the latency constraint, we solve the rate and power allocation problem in Steps 2 and 3 of Alg. 2, and check whether the resulting solution satisfies the power constraints in (2.10e).

Capacity approximations

In our second experiment, we consider the performance of "conservative-relaxation" methods that use different approximations of the finite-block-length capacity. In Fig. 2.2 we plot the trade-off curve between the error probability and energy usage for each method, averaged over the 457 channel realizations used to generate Fig. 2.1. This figure shows that the proposed NA-based method produces a trade-off that dominates those of the other methods, and dominates the baseline design and the proposed SNR-Gap designs by a considerable margin. In this setting, the block lengths are typically in the range of 150-400 channel uses, and hence the conservatively-relaxed TTA-based design achieves a similar trade-off to the NA-based design.[5]

Beyond performance trade-offs on channel realizations for which every design method achieves a feasible solution, we should also evaluate the probability that each method generates a solution that meets the specified probability of error. For the proposed NA-based conservative-relaxation method, this is, by design, simply the probability that the reduced-dimension conservatively relaxed problem in (2.21) has a feasible solution. (That problem has a convex feasible set.) For the other methods, we do not have explicit control over the probability of error during design process, and hence it is determined after the design is completed. As shown in Fig. 2.3, the proposed conservative-relaxation approach is significantly more likely to produce a feasible solution than the baseline design, and that, the NA-based design is more likely to produce a feasible solution than those the SNR-Gap

[5]As suggested above, the curves for the proposed SNR-Gap and baseline designs are obtained by varying the gap and computing the probability of error after the design is completed. Similarly, the curve for the TTA-based design is obtained by varying ϵ in $\hat{\psi}_{\epsilon}^{(2)}(P,\tau)$, and then computing the probability of error after the design is completed.

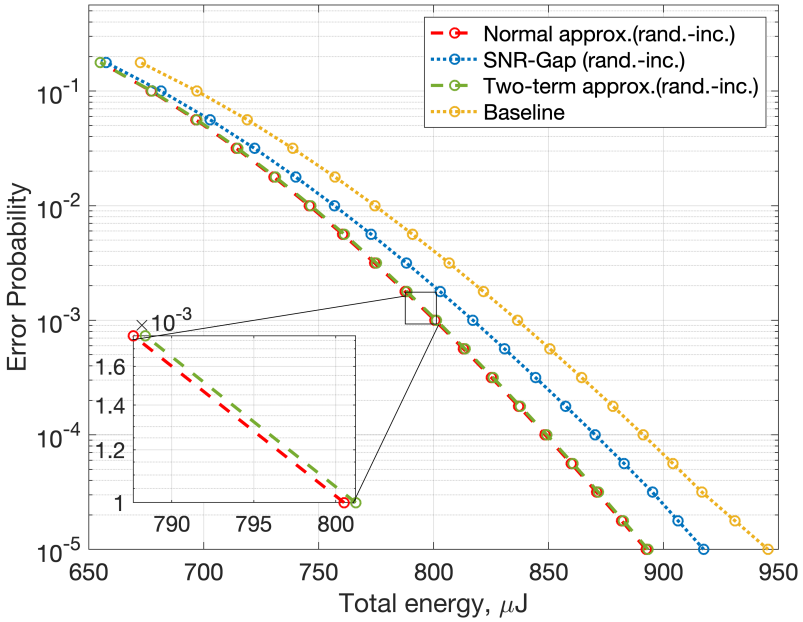


FIGURE 2.2: Error probability versus energy trade-offs achieved by the proposed design method with different capacity approximations.

and TTA-based designs.

2.6.2 Binary offloading

When some or all of the devices have the capability to complete their task locally (i.e., $t_{\text{loc}_k} \leq L_k$), the system has the additional degree of freedom to decide which devices should offload their tasks and which should complete their task locally. In Sec. 2.5 we suggested that these decisions could be made using a pruned greedy tree search over complete-offloading problems. In the following sections, we will demonstrate the performance of that scheme in a case in which all four devices can complete their tasks locally, and $E_{\text{loc}_k} = 640$ µJ for each device.

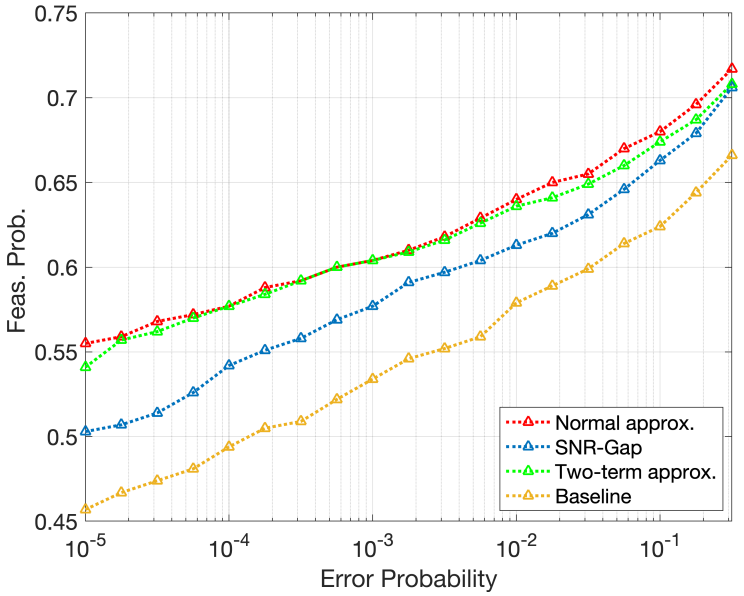


FIGURE 2.3: Probability of feasibility as a function of the probability of error achieved by the proposed design method with different capacity approximations.

Effectiveness of tree search

In Fig. 2.4 we compare the error probability versus total energy trade-offs achieved by various binary and complete-offloading systems in the scenario of Figs 2.1–2.3. Implicit in Fig. 2.4 is the fact that as we reduce the allowable probability of error, the energy required for sufficiently reliable transmission increases and hence fewer devices are selected for offloading in the binary case. Indeed, at a probability of error of 10^{-4}, the binary offloading schemes select an average of $82\%, 81.75\%, 81.5\%$ and 80.75% of the devices for offloading in the NA, TTA, SNR-Gap, and baseline designs, respectively.

To assess the effectiveness of the pruned greedy tree search, we have included the trade-off achieved by a scheme that performs an exhaustive search over all 2^K

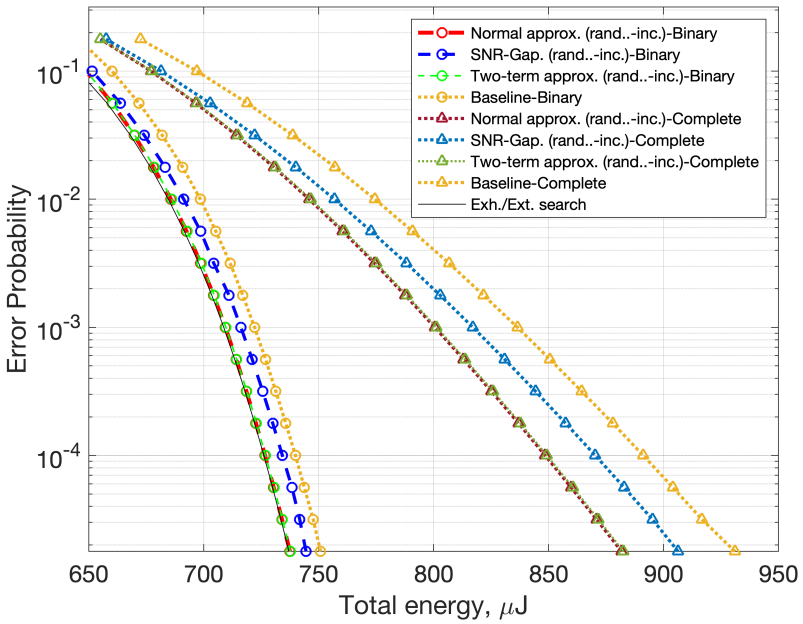


FIGURE 2.4: Error probability versus total error trade-offs achieved by different binary and complete-offloading schemes.

possible offloading decisions, and for each decision performs the extensive-search scheme described in Sec. 2.6.1 for solving the complete-offloading problem. That scheme is denoted by "Exh./Ext. search" in Fig. 2.4. The fact that the trade-offs achieved by the proposed NA and TTA schemes are almost indistinguishable from this benchmark attests to the effectiveness of the proposed pruned greedy tree search.

Variation of performance with position

In this example, we examine the performance of the proposed design as the distance of one of the devices changes. Devices 2 to 4 are placed at distances of 200, 250, and 300m from the access point, respectively, and device 1 is moved, in a quasi-static way, from a distance of 100m to 1,000m. The results are averaged over

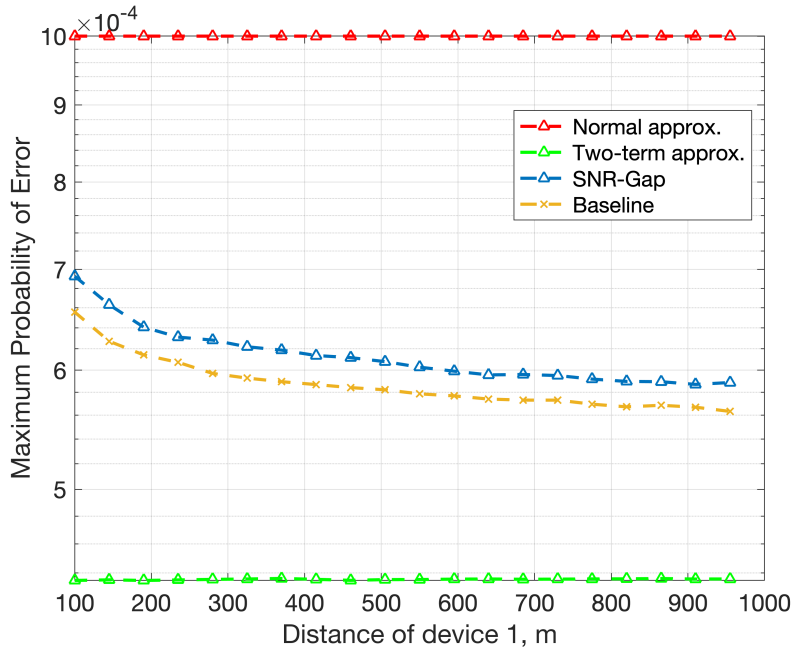


FIGURE 2.5: Probability of error vs distance.

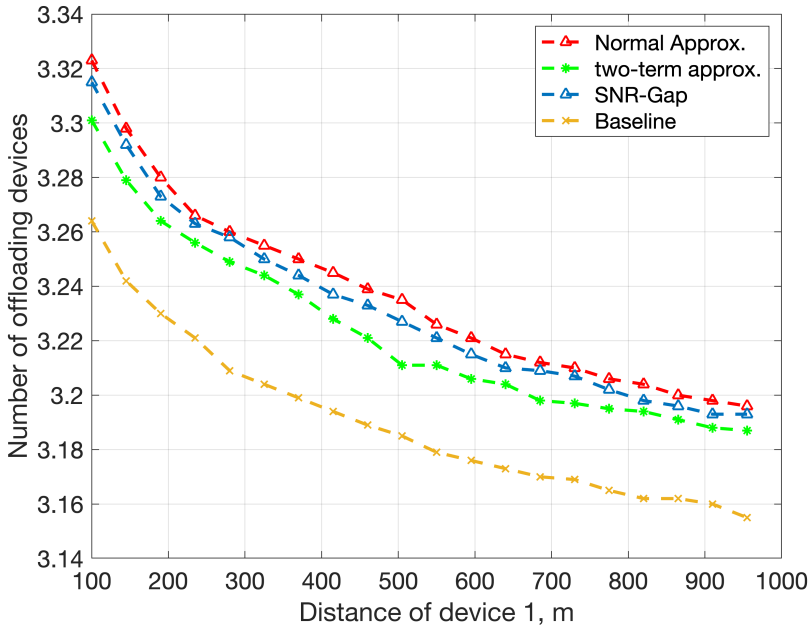


FIGURE 2.6: Number of offloading devices vs distance.

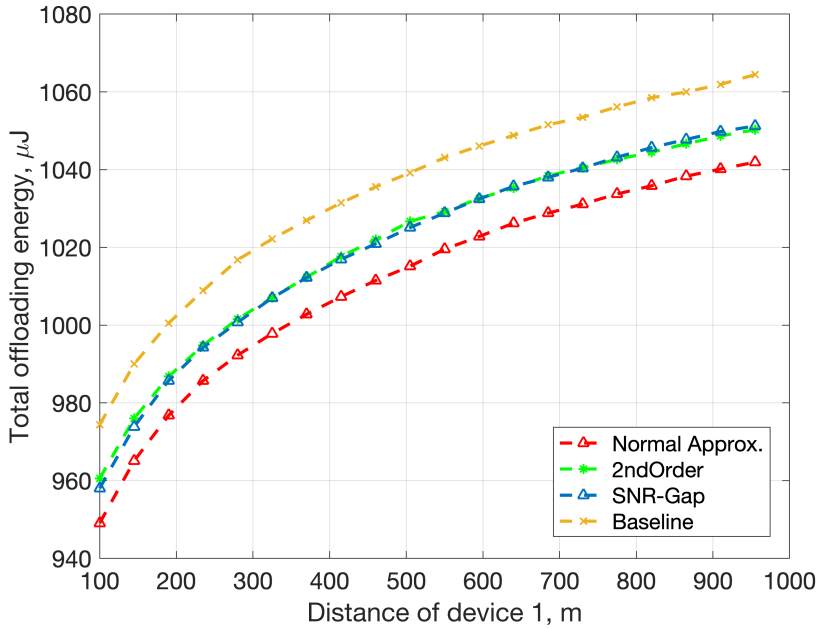


FIGURE 2.7: Total energy vs distance.

1,000 realizations. For the proposed NA-based method, we set the probability of error to be 10^{-3}. As described earlier, the other methods do not provide explicit control over the probability of error. For the TTA-based design we set the target probability of error in that sense to be 10^{-3}. As shown in Fig. 2.5, this results in an offset from the normal approximation due to the influence of the term $\log_2(2\tau)/2\tau$ in (2.8). While that term varies with the block length of each device in each channel realization, it is clear from Fig. 2.5 that the variation in quite small in this setting. For the SNR-Gap and baseline methods we choose the same value for Γ, namely $\Gamma = 1.2$ dB, and it remains constant for all distances. That value was chosen so that the probability of error lies in between those of the normal and two-term approximations.

In Fig. 2.6 we have plotted the average number of devices selected for offloading

as a function of the distance of device 1. As expected, this number decreases as the distance increases, because device 1 will require more communication resources if it offloads. For example, when device 1 is at a distance of 200m, the probability that devices 1 to 4 are offloaded in the NA-based design are 0.86, 0.83, 0.81, 0.78, respectively, whereas at 800m they are 0.75, 0.86, 0.81, 0.78, respectively. (Note that while the overall offloading probability decreases, the reduction in the offloading probability of device 1 at 800m provides the opportunity for device 2 to offload slightly more often.) Fig. 2.6 shows that the NA-based design enables more devices to offload than the SNR-Gap and baseline designs. That figure also shows that the conservative-relaxation approach used in the SNR-Gap design results in more devices offloading than the baseline design, even though they have similar probability of error. Furthermore, the energy expenditure in Fig. 2.7 shows that the conservative-relaxation approach also enables the proposed SNR-Gap design to expend less total energy than the baseline design.

2.7 Conclusion

This chapter has developed efficient algorithms for multi-user binary computation offloading that explicitly incorporate approximations of the fundamental rate limit of communication over finite block lengths. The integer nature of the block length is handled using a relaxation-rounding scheme with a customized incremental rounding scheme and a constraint-tightening technique that guarantees that the deterministically-rounded block lengths will be feasible wherever the relaxed problem has a feasible solution. Using the structure of the design problem, the relaxed problem is reduced to an optimization over only the rates. The proposed

design process yields performance that is close to that of an "extensive-search" technique, and the incorporation of a randomized rounding scheme essentially closes the remaining gap. The proposed algorithm is embedded within a pruned greedy tree search for the set of offloading devices. The numerical results show that the proposed "conservative-relaxation" approach dominates, in the Pareto sense, a variety of competing approaches, including a baseline scheme obtained by simply rounding existing design approaches that are based on insights from the asymptotic-block-length regime.

Acknowledgment

I would like to thank Professor Sorina Dumitrescu of McMaster University for her insightful comments on a draft of this manuscript.

2.A Development of (2.16)

Using the substitutions described in the opening paragraph of Sec. 2.4.2, the initial conservative relaxation of the problem in (2.10) for the incremental rounding scheme can be written as

$$\min_{\{\breve{R}_k\},\{\breve{P}_k\}} \quad \sum_{k=1}^{\tilde{K}} \lambda_k B_k \frac{\breve{P}_k}{\breve{R}_k} \tag{2.27a}$$

$$\text{s.t.} \quad \frac{B_k}{\breve{R}_k} > \frac{1}{2} - \xi_k, \tag{2.27b}$$

$$\sum_{i=1}^{k} \frac{B_i}{\breve{R}_i} < \tilde{L}_k + \frac{1}{2}, \tag{2.27c}$$

$$0 < \breve{R}_k \leq \omega_k(\breve{P}_k, \breve{R}_k, \xi_k), \tag{2.27d}$$

$$0 < \breve{P}_k \leq \bar{P}_k, \tag{2.27e}$$

where the constraints apply for all $k \in \{1, 2, \ldots, \tilde{K}\}$ and

$$\omega_k(\breve{P}_k, \breve{R}_k, \xi_k) = \frac{\log_2(1+\alpha_k \breve{P}_k) - g_k(\breve{P}_k, \breve{R}_k, \xi_k)\sqrt{\breve{R}_k}}{1+y_k(\breve{P}_k, \xi_k)+f_k(\breve{R}_k, \xi_k)}, \tag{2.28}$$

with

$$y_k(\breve{P}_k, \xi_k) = \frac{(1-2\xi_k)\log_2(1+\alpha_k \breve{P}_k)}{2B_k}, \tag{2.29}$$

$$f_k(\breve{R}_k, \xi_k) = -\frac{(2B_k+(2\xi_k-1)\breve{R}_k)}{(2B_k+(2\xi_k+1)\breve{R}_k)} \times \frac{\log_2((2B_k+(2\xi_k-1)\breve{R}_k)/\breve{R}_k)}{2B_k}, \tag{2.30}$$

$$g_k(\breve{P}_k, \breve{R}_k, \xi_k) = \sqrt{\frac{(1+(2\xi_k-1)\breve{R}_k/(2B_k))V(\breve{P}_k)}{B_k}}\, Q^{-1}(\epsilon_k). \tag{2.31}$$

For device 1, $\xi_1 = 0$ and hence its constraints are continuous functions of R_1 and P_1. However, for $2 \leq k \leq \tilde{K}$ the constraints in (2.27b) and (2.27d) are discontinuous functions of $\{\breve{R}_i\}_{i=1}^{k-1}$, through ξ_k. Therefore, we look for inner bounds on these constraints that are continuous functions of the variables. (Inner bounds result in restrictions of the feasible set.) For the positive block length constraint in (2.27b), that simply involves setting $\xi_k = -1/2$, for $k \geq 2$; see (2.16b). For the maximum rate constraint in (2.27d), both terms in the denominator are decreasing functions of ξ_k, and $g_k(\breve{P}_k, \breve{R}_k, \xi_k)$ is an increasing function of ξ_k. Hence, for $k \geq 2$ we will construct the inner bound by setting $\xi_k = -1/2$ in both terms in the denominator, and $\xi_k = 1/2$ in the numerator; see (2.16d).

2.B Solving (2.10) in the asymptotic regime

One approach to solving (2.10) in the asymptotic-block-length regime (cf. Salmani and Davidson (2020b))[6] begins by relaxing the integer constraint on the block length and using (2.10f) to obtain $\tau_k = B_k/R_k$. Then, since $\hat{\psi}_{\epsilon_k}(P_k, \tau_k) = \log_2(1 + \alpha_k P_k)$, we can obtain a closed-form solution for the optimal powers for given rates,

$$P_k(R_k) = \tfrac{2^{R_k}-1}{\alpha_k}. \tag{2.32}$$

That enables us to reduce the overall allocation problem to the following rate allocation problem:

$$\min_{\{R_k\}} \quad \textstyle\sum_{k=1}^{\tilde{K}} \lambda_k \frac{B_k}{\alpha_k}\left(\frac{2^{R_k}-1}{R_k}\right} \tag{2.33a}$$

$$\text{s.t.} \quad \textstyle\sum_{i=1}^{k} \frac{B_i}{R_i} \leq \tilde{L}_k, \tag{2.33b}$$

$$0 < R_k \leq \log_2(1 + \alpha_k \bar{P}_k), \tag{2.33c}$$

where the constraints are applied to all $k \in \tilde{\mathcal{S}}'$. This problem is feasible if and only if (2.33b) is satisfied when each $R_k = \log_2(1 + \alpha_k \bar{P}_k)$. Furthermore, it is convex and can be efficiently solved. The corresponding powers and (relaxed) transmission block lengths can be generated using the closed-form expressions above.

2.C Independent rounding of relaxed block lengths

In (2.16) we provided a conservative relaxed formulation for the incremental rounding case. By setting $\xi_k = 0$ for all offloading devices we obtain the corresponding

[6]Note that in Salmani and Davidson (2020b) there was no explicit bound on the transmission power.

problem for the independent rounding case:

$$\min_{\{\check{R}_k\},\{\check{P}_k\}} \quad \textstyle\sum_{k=1}^{\tilde{K}} \lambda_k B_k \frac{\check{P}_k}{\check{R}_k} \tag{2.34a}$$

$$\text{s.t.} \quad \check{R}_k < 2B_k, \tag{2.34b}$$

$$\textstyle\sum_{i=1}^{k} \frac{B_i}{\check{R}_i} < \tilde{L}_k + \frac{1}{2} - \frac{(k-1)}{2}, \tag{2.34c}$$

$$0 < \check{R}_k \leq \check{\Omega}_k(\check{P}_k, \check{R}_k), \tag{2.34d}$$

$$0 < \check{P}_k \leq \bar{P}_k, \tag{2.34e}$$

where the constraints are applied for all $k \in \tilde{\mathcal{S}}'$, $\check{\Omega}_k(\check{P}_k, \check{R}_k) = \Big(\log_2(1+\alpha_k\check{P}_k) - \check{b}_k(\check{P}_k, \check{R}_k)\sqrt{\check{R}_k}\Big)/\check{a}_k(\check{P}_k, \check{R}_k)$, with $\check{a}_k(\check{P}_k, \check{R}_k) = 1+y_k(\check{P}_k, 0)+f_k(\check{R}_k, 0)$ and $\check{b}_k(\check{P}_k, \check{R}_k) = g_k(\check{P}_k, \check{R}_k, 0)$, where $y_k(\check{P}_k, \xi_k)$, $f_k(\check{R}_k, \xi_k)$ and $g_k(\check{P}_k, \check{R}_k, \xi_k)$ are defined in App. 2.A. A key difference between this formulation and that in (2.16) is the communication deadline constraint in (2.34c). Since the independent rounding scheme is (quite naturally) unable to take advantage of the residuals from rounding the block lengths of the previous devices, this constraint is more conservative than (2.16c). Furthermore, the constraint in (2.34c) becomes increasingly conservative as k increases.

Using a similar derivation to that in Sec. 2.4.3, but with $\xi_k = 0$ for all k, the problem that is analogous to (2.24) has a similar structural form, but with (i) the right hand side of (2.21b) replaced by $\tilde{L}_k + \frac{1}{2} - \frac{(k-1)}{2}$; (ii) the term $\hat{\Phi}_k(\hat{R}_k; a_k^-, b_k^-)$ replaced by $\hat{\Upsilon}_k(\hat{R}_k; \check{a}_k^-, \check{b}_k^-) = \frac{2^{\left(\check{a}_k^- \hat{R}_k + \check{b}_k^- \sqrt{\hat{R}_k}\right)} - 1}{\alpha_k}$, where $\check{a}_k^-$ and $\check{b}_k^-$ are defined by analogy with the incremental case, using the expressions for $\check{a}_k(\check{P}_k, \check{R}_k)$ and $\check{b}_k(\check{P}_k, \check{R}_k)$ above; and (iii) the term $\acute{R}_k$ is replaced by $\min\{\bar{\check{R}}_k, 2B_k\}$, where $\bar{\check{R}}_k$ satisfies $\bar{\check{R}}_k = \check{\Omega}_k(\bar{P}_k, \bar{\check{R}}_k)$.

This approach will yield an algorithm that is analogous to Alg. 1 in the sense that if that algorithm generates a feasible solution, the deterministically-independently-rounded block lengths, along with the corresponding updated rates and powers are guaranteed to constitute a feasible solution for (2.10). However, since the independent rounding scheme does not take advantage of the residuals from rounding the block lengths of the previous devices, the randomized search for better solutions than the deterministically-rounded solution ought to be expanded beyond the integer neighbours of the relaxed block length that would be generated by the direct analogy of Alg. 2. Therefore, we suggest the following modifications be made to Alg. 2 for the case of independent rounding.

- **Input data:** The input data rates are $\{\check{R}_k^\star\}$, the rates provided by the variant of Alg. 1 for the independent rounding case. The input data are augmented to include a set of $\tilde{K}$ integers $\{w_k\}$ that represent the half-width of the randomization procedure.

- **Step 0:** An initialization step is added in which for each device k we define $\mathcal{T}_k$ to be the set of integers in the interval $\left[\lfloor B_k/\check{R}_k^\star \rfloor - w_k,\ \lceil B_k/\check{R}_k^\star \rceil + w_k\right]$, and we compute the width of that interval, namely $W_k = 2w_k + 1$.

- **Step 1:** The block lengths τ_k are computed by independent rounding; i.e., $\tau_k = \lfloor B_k/\check{R}_k^\star \rceil$.

- **Step 6:** This step is replaced by: Set $n_{\text{rand}} = 0$, and for each k, and each integer $q_i \in \mathcal{T}_k$, compute

$$p_{k,i} = \frac{W_k - |B_k/\check{R}_k^\star - q_i|}{\sum_{q_i \in \mathcal{T}_k}(W_k - |B_k/\check{R}_k^\star - q_i|)}. \tag{2.35}$$

- **Step 7a:** This step is removed
- **Steps 7b–7d:** Randomly generate τ_k, with each τ_k being set, independently, to $q_i \in \mathcal{T}_k$ with probability $p_{k,i}$.

There are two key parameters in this modified version of Alg. 2. The first is w_k, the half-width of the randomized search for integer block lengths for device k. If we set $w_k = 0$, then the randomized search is only over the nearest integer neighbours to $B_k/\check{R}_k^\star$. If we can afford a significant number of randomised trials, a choice that is arguably better tailored to the structure of the conservatism in (2.34c) is to choose $w_k = \lceil k/2 \rceil$. The second parameter is the choice of the probability mass function in Step 6. There are many possible choices, but most reasonable choices will have the bulk of the probability mass concentrated on integers that are close to $B_k/\check{R}_k^\star$.

2.D Strong Convexity of (2.23)

To establish the strong convexity of (2.23) it suffices to establish the strong convexity of $f(R;\, a, b) = \left(2^{(aR+b\sqrt{R})} - 1\right)/R$ for $a \geq 1$ and $b \geq 0$. If we define $g(R) = \log(2)(aR + b\sqrt{R})$, then the second derivative of $f(R)$ with respect to R can be written as $f''(R;\, a, b) = 2^{(aR+b\sqrt{R})} z(R;\, a, b)/R^3$, where

$$z(R;\, a, b) = \left(Rg'(R) - 1\right)^2 + 1 + R^2 g''(R) - 2^{(1-(aR+b\sqrt{R}))}, \tag{2.36}$$

with $g'(R) = \log(2)(a + bR^{-1/2}/2)$ and $g''(R) = -\log(2)bR^{-3/2}/4$. To establish strong convexity, we will first show that $f''(R;\, a, 0) > 0$ for all $R > 0$ and $a \geq 1$.

Then we will show that for $b \geq 0$, $f''(R;\, a, b)$ is an increasing function of b for all $R > 0$ and $a \geq 1$.

For the first step, we will observe that since $h(R) = 2^{aR} > 0$, it is sufficient to show that $z(R;\, a, 0) > 0$. Since, $z(0;\, a, 0) = 0$, to show that $z(R;\, a, 0) > 0$ for $R > 0$ it sufficient to show that its derivative with respect to R is positive; i.e.,

$$z'(R;\, a, 0) = 2\log(2)a\Big(\log(h(R)) - 1 + 1/h(R)\Big) > 0. \tag{2.37}$$

Since the function $\log(x) - 1 + 1/x > 0$ for all $x > 1$ and since $h(R) > 1$ for all $R > 0$, $z'(R;\, a, 0) > 0$ for all $R > 0$.

For the second step, we observe that $\frac{\partial}{\partial b} f''(R;\, a, b) = \frac{\log(2)2^{(aR+b\sqrt{R})}}{4R^{5/2}} T(R)$, where

$$T(R) = A_1 R^2 - B_1 R + A_2 R^{3/2} - B_2 R^{1/2} + 3, \tag{2.38}$$

with $A_1 = 4a^2\log^2(2)$, $A_2 = 4ab\log^2(2)$, $B_1 = 4a\log(2) - b^2\log^2(2)$, and $B_2 = 3b\log(2)$. Therefore, to complete the proof if suffices to show that $T(R) > 0$. To assist in doing so, we define $T_1(R) = A_1 R^2 - B_1 R$, and $T_2(R) = A_2 R^{3/2} - B_2 R^{1/2}$.

To prove that $T(R) > 0$ for all $R > 0$, we will first consider the case where $B_1 > 0$. By taking the derivative and setting it to zero, we can show that $T_1(R)$ achieves its minimum value when $R = R_1^\star = B_1^2/(4A_1)$. That minimum value is $T_{1,\min} = -(b^2\log(2)/(4a) - 1)^2$. Since $B_1 > 0$, $b^2\log(2)/(4a) < 1$ and hence $T_{1,\min} > -1$. In a similar manner, we can show that $T_2(R)$ achieves its minimum value when $R = R_2^\star = B_2/(3A_2) = 1/(4a\log(2))$, and that minimum value is $T_{2,\min} = -\Big(b^2\log(2)/a\Big)^{1/2}$. Since $B_1 > 0$, $b^2\log(2)/a < 4$ and hence $T_{2,\min} > -2$.

For all $R > 0$, $T(R) \geq T_{1,\min} + T_{2,\min} + 3$, and hence when $B_1 > 0$, $T(R) > 0$ for all $R > 0$.

When $B_1 = 0$, $\inf_{R>0} T_1(R) = 0$, and a similar analysis reveals that $T(R) > 0$ for all $R > 0$.

In the case where $B_1 < 0$, $T_3(R) = A_1R^2 - B_1R + A_2R^{3/2}$ is a positive increasing function for all $R > 0$. For $R \leq 1/(b^2 \log^2(2))$, $T_4(R) = 3 - B_2R^{1/2} \geq 0$ and hence $T(R) = T_3(R) + T_4(R) > 0$ for $R \leq 1/(b^2 \log^2(2))$. In addition, when $B_1 < 0$, $T_1(R)$ is a positive increasing function for all $R > 0$, and $T_2(R)$ is an increasing function for all $R > R_2^\star$. If we let $x = b\sqrt{\log(2)/a}$, then we can write $T(R_2^\star) = T_1(R_2^\star) + T_2(R_2^\star) + 3$ as $x^2/4 - x + 9/4$. That quadratic function is positive for all (positive) x, and hence $T(R_2^\star) > 0$. That means that $T(R_2^\star) = T_1(R_2^\star) + T_2(R_2^\star) + 3 > 0$, and since $T(R)$ is an increasing function for $R \geq R_2^\star$, we have that $T(R) > 0$ for all $R \geq R_2^\star$. What remains to complete the proof for the case of $B_1 < 0$ is to show that $T(R) > 0$ for all $R \in \Big(1/(b^2 \log^2(2)), 1/(4a \log(2))\Big)$. For this case, we write $T(R) = A_1R^2 + T_5(R)$, where $T_5(R) = -B_1(R) + A_2R^{3/2} - B_2R^{1/2} + 3$, and observe that since $A_1R^2 > 0$, it is sufficient to show that for this range of values of R, $T_5(R) > 0$. If we define $y = b \log(2) R^{1/2}$ and $\beta = 4a/(b^2 \log(2)$ we can rewrite $T_5(R)$ as $(1 - \beta)y^2 + 2a \log(2) Ry - 3y + 3$. Furthermore, for $R \in \Big(1/(b^2 \log^2(2)), 1/(4a \log(2))\Big)$, $0 < y < 1$ and $0 < \beta < 1$. Therefore, $(1 - \beta)y^2 > 0$, $2a \log(2) Ry > 0$ and $-3y + 3 > 0$, and hence $T_5(R) > 0$.

Since we have now shown that $T(R) > 0$ for all $R > 0$ when $B_1 > 0$, $B_1 = 0$ and $B_1 < 0$, $f''(R; a, b)$ is an increasing function of b for all $R > 0$ for all values of $a \geq 1$, $b \geq 0$. Since $f''(R; a, 0) > 0$ for all $R > 0$ and $a \geq 1$, that means that we

have shown that $f''(R;\, a, b) > 0$ for all $R > 0$, $a \geq 1$ and $b \geq 0$, and hence that $f(R;\, a, b)$ is a strongly convex function of R.

2.E Initialization of Alg. 1

Since feasible set of (2.24) is the same as that of (2.21), we can determine whether (2.24) is feasible by determining whether (2.21b) holds when each $\breve{R}_k = \acute{R}_k$; see the discussion after (2.21). If that test shows that the problem is feasible, then we can generate a strictly feasible point for (2.21), and hence for (2.24), by determining a set of rates $\{\breve{R}_k\}$ such that each $\breve{R}_k < \acute{R}_k$ while (2.21b) still holds. A simple way to generate such a set is to is to pick a (large) value for $\rho \in (0, 1)$, set an integer $m = 1$, construct $\tilde{R}_k = (1 - \rho^m)\acute{R}_k$ for all k, and then test the feasibility of (2.21b) for the set of rates $\{\breve{R}_k = \tilde{R}_k\}$. If that constraint is not satisfied, then increment m and repeat until a strictly feasible set $\{\breve{R}_k\}$ is found.

Chapter 3

Joint Computing and Communication Resource Allocation for Binary Computation Offloading

Abstract

In order to effectively utilize shared computing resources at an access point, careful selection of the offloading devices and joint allocation of the computing and communication resources are essential. The joint allocation problem is combinatoric, and this chapter addresses that challenge by employing a decomposition strategy. For a scenario in which the set of offloading devices has been selected, we introduce effective heuristics for computation resource allocation and propose an optimal communication resource allocation algorithm for a given computation

resource allocation under the time division multiple access (TDMA) scheme. We then extend that approach to tackle the problem of selecting the offloading devices, using a methodology similar to that used in the previous chapter. Our numerical results demonstrate that the proposed efficient algorithms provide performance that is close to that of the jointly optimal solution.

3.1 Introduction

Mobile edge computing can enhance the computational capacity of wireless devices by enabling the offloading of computational tasks to their associated access point. However, the edge server at the access point only has limited computation resources. Consequently, in order to effectively enhance the devices' computational capabilities using the MEC approach, the limitations of both the edge server's computation resources and the (multi-user) communication system between the devices and the access point must be considered. These perspectives lead us to a problem of jointly selecting the devices that will offload their tasks and allocating the computing and communication resources to those offloading devices. As in the previous chapter, the objective of our joint selection–allocation problem will be to minimize the cost of the energy expended by the mobile devices while ensuring that they receive the results of their computational tasks within a specified deadline.

In this chapter, we address the joint selection–allocation problem in a scenario with a single fixed-frequency CPU at the access point, and with the time division multiple access (TDMA) scheme. The methodology for selecting the offloading

devices remains consistent with the previous chapter. While the joint selection–allocation problem is ultimate goal of this chapter, our key contributions lie in addressing the case of a fixed-set of offloading devices (i.e., the complete offloading problem), and jointly allocating the shared computation and communication resources in that case.

To develop the proposed algorithm, we show that for an optimal computing allocation, it is sufficient to allocate all the shared computing resources to one device at a time. That results in a formulation for the allocation that is a large scale combinational problem. To tackle that challenge we introduce heuristic algorithms for computation resource allocation, enabling us to develop a simple algorithm for the determining a good transmission order for the offloading devices. These outcomes allow us to formulate an efficiently solvable communication resource allocation problem that minimizes the energy cost of offloading. This involves optimizing the transmission block lengths, rates, powers, subject to latency constraints determined by the computation resource allocation.

Computation resource allocation has been studied in MEC systems, as demonstrated by Dinh et al. (2017), where the objective is to reduce both the overall execution time of tasks and the energy consumption of the devices by considering both fixed CPU frequency and variable CPU frequency in the offloading process. In this chapter, we assume that the CPU frequency is fixed. In the previous chapter, we focused on communication resource allocation for a fixed computation resource allocation. In this chapter, we aim to jointly allocate computing and communication resources for the offloading devices by decomposing the problem

into two layers. First, the computation resource allocation is addressed by proposing two good heuristics, one of which successively maximizes the communication deadline starting with the largest. The other one has the same overall goal, but constrains the allocation so that each task is computed in a contiguous time block. We show that after finding a good solution for the computation resource allocation problem, we can easily determine the communication deadline for each offloading device. Then, similar to Chapter 2, we allocate the communication resources based on the communication deadlines.

This chapter is organized as follows: In Sec. 3.2, we establish the computation and communication models for both the complete and binary offloading problems. In Sec. 3.3, we outline the decomposition of the problem into computation and communication resource allocation problems. In Sec. 3.4 we establish two efficient heuristic schemes for allocating the computing resources, and show how this reveals the corresponding user transmission ordering. In Sec. 3.5 we develop our efficient algorithm for the remaining uplink communication resource allocation problem, and in Sec. 3.6, we provide some results that illustrate the performance of our approaches.

3.2 System model

As in Chapter 2, we consider a computation offloading system with K single-antenna devices and a single-antenna access point. Each device seeks to complete a computational task within its own specified deadline. The tasks are modeled as being indivisible, and hence the system must make a binary decision as to whether each device should offload its task to the access point or fully execute the task

locally. The contribution of this chapter is to develop a system that makes that decision jointly with the allocation of computation resources at the access point and communication resources on the uplink to the access point. The goal is to minimize a weighted sum of the energies expended by the devices, subject to the deadlines by which each device must receive its results, the computational capacity of the access point, and the communication capacity of the multiple access channel.

3.2.1 Offloading model

We consider a scenario in which the k^{th} device's task can be described in B_k bits and requires W_k operations to complete. The device requires the solution to the task within L_k seconds. The access point is able to compute F operations per second and the device can compute F_k operations per second. Since each device's task is modelled as being indivisible, the system must make a binary decision $\gamma_k \in \{0, 1\}$ as to whether the task is to be offloaded or computed locally. If $\gamma_k = 0$, the task is computed locally, which takes $t_{\text{loc},k} = W_k/F_k$ seconds and consumes $E_{\text{loc},k}$ units of energy. For this to be feasible, we must have $t_{\text{loc},k} \leq L_k$.

If $\gamma_k = 1$, the description of the task is transmitted to the access point, the task is completed there, and then the solution is returned to the device. We will consider the class of tasks for which the full description of the task must be received before computation can begin, and for which the results become available once the task has been completed. Therefore, for an offloading device, if $t_{\text{del},k}$ denotes the elapsed time over which the description of the task is delivered to the access point, $t_{\text{exe},k}$ denotes the elapsed time over which the computation is performed, and $t_{\text{DL},k}$ denotes the time that it takes to return the result to the device, then as illustrated

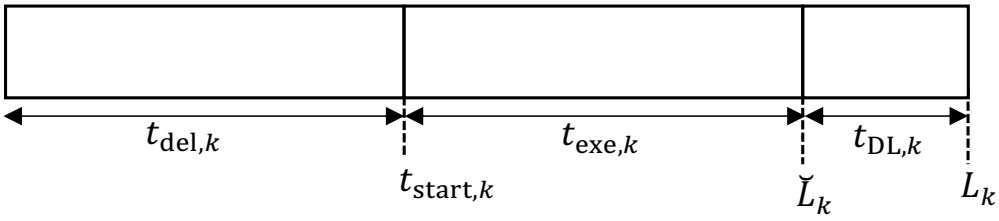


FIGURE 3.1: General offloading time slot model for an individual device.

in Fig. 3.1 for the case of a single device, for offloading to be feasible we must satisfy

$$t_{\text{del},k} + t_{\text{exe},k} + t_{\text{DL},k} \leq L_k. \tag{3.1}$$

Since the result to be communicated back to the device often has a significantly shorter description than the task itself, and since each device's task may be completed at a different time, we will assume that t_{DL_k} is a (different) constant for each device. (Similar models are adopted in most of the existing literature; e.g., Sardellitti et al. 2015; Wang et al. 2018; You et al. 2017; Salmani and Davidson 2020b; Salmani and Davidson 2020a.) As a result, we will rewrite (3.1) as

$$t_{\text{del},k} + t_{\text{exe},k} \leq \breve{L}_k \tag{3.2}$$

where $\breve{L}_k = L_k - t_{\text{DL},k}$, is the computing deadline for device k; see also Fig. 3.1. That is, the time by which the access point must complete the task for device k.

3.2.2 Computation model for offloading devices

To simplify the development of our models, let us first consider a scenario in which $\breve{K}$ devices have been selected for offloading. Without loss of generality, we will

re-index those devices starting from one and will re-order them such that

$$\breve{L}_1 \leq \breve{L}_2 \leq \cdots \leq \breve{L}_{\breve{k}} \leq \cdots \leq \breve{L}_{\breve{K}}. \tag{3.3}$$

We will use the index $\breve{k}$ to denote that re-indexing. With this ordering, we know that by $\breve{L}_{\breve{k}}$ the access point must have completed the computational tasks of devices $1, 2, \ldots, \breve{k}$. That leads naturally to the notion of a computing time slot. For $j = 2, 3, \ldots, \breve{K}$ we define the j^{th} computing time slot to be the interval $(\breve{L}_{j-1}, \breve{L}_j]$, which is of duration

$$\Delta_j = \breve{L}_j - \breve{L}_{j-1} \tag{3.4}$$

in seconds. In the $\breve{K}^{\text{th}}$ computing time slot we can only allocate computing resources to device $\breve{K}$'s task, as all the other tasks need to be completed by $\breve{L}_{\breve{K}-1}$. Similarly, in computing time slot $\breve{K}-1$ we can only allocate computing resources to the tasks of devices $\breve{K}-1$ and $\breve{K}$, and in the j^{th} computing time slot, we can allocate computing resources to devices $j, j+1, \ldots, \breve{K}$. In order to define the first computing time slot in a way that enables consistent notation, we define $\breve{L}_0$ to be the time at which the first computation begins, and we define the first computing time slot to be $(\breve{L}_0, \breve{L}_1]$ which is duration $\Delta_1 = \breve{L}_1 - \breve{L}_0$. Although it may appear to be a design variable, as we show in App. 3.A the optimal value of $\breve{L}_0$ can be determined prior to the computing resource allocation, and can be achieved by a sizable class of computing resource allocation scheme.

Given our definition for the computing time slots in terms of the computing deadlines, in the j^{th} computing time slot we will have allocate the computing resources amongst the tasks of (at most) $\breve{K}+1-j$ devices, namely devices $j, j+$

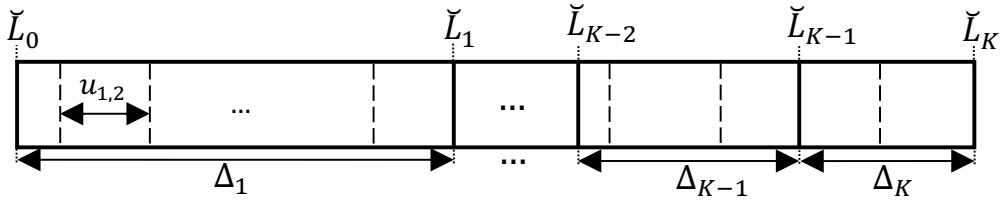


FIGURE 3.2: Computation model at the access point.

$1, \ldots, \breve{K}$ according to the indexing in (3.3). For simplicity, we will adopt a model in which the computing resources can be shared in an arbitrary manner, and that this sharing can be changed at any time, without any "context switching" penalty.

As we show in App. 3.B, under this computing model an optimal computing resource allocation can be obtained by allocating all the computing resources to the task of one device at a time, and by allocation at most one computing subslot to each user withing a given computing time slot. Since there are up to $\breve{K}+1-j$ tasks to consider in the j^{th} computing slot and since for slots 2 to $\breve{K}$ we need to consider the possibility of an idle computing slot, for $j = 2, \ldots, \breve{K}$ we will divide the j^{th} computing time slot into $\breve{K}+2-j$ subslots of duration $u_{jm}\Delta_j$, with $m = 0, 1, \ldots, \breve{K}+1-j$ and $u_{jm} \in [0,1]$, and we will use $f_{jm\breve{k}} \in \{0,1\}$ to indicate whether or not the computing resources are allocated to the task of device $\breve{k}$ in computing subslot (j,m); see Fig. 3.2 for an illustration. In the first computing time slot there is no need to consider an idle time slot so in that slot we need only consider (at most) $\breve{K}$ computing subslots. However, for notational simplicity we will notionally allocate $\breve{K}+1$ subslots that case. (At optimality, the redundant subslot in the first computing subslot will have duration of zero.) Since device $\breve{k}$'s task must be completed by the end of the $\breve{k}^{\text{th}}$ computing time slot, the set of computing allocation variables is $\mathcal{F} = \left\{ f_{jm\breve{k}} \in \{0,1\} \right\}_{j=1,m=0,\breve{k}=j}^{\breve{K},\breve{K}+1-j,\breve{K}}$. Since at

most one task can be worked on in each subslot, we have $\sum_{\breve{k}=j}^{\breve{K}} f_{jm\breve{k}} \leq 1$ for all j and m, and since it is sufficient to allocate at most one subslot to each task, we have $\sum_{m=0}^{\breve{K}+1-j} f_{jm\breve{k}} \leq 1$ for all j and $\breve{k}$. The fractional durations of each computing subslot satisfy

$$u_{jm} \geq 0, \quad \text{and} \quad \sum_{m=0}^{\breve{K}+1-j} u_{jm} = 1. \tag{3.5}$$

With this computing model in place, the number of operations of device $\breve{k}$'s task that are completed in the (j,m)th computing subslot is $(f_{jm\breve{k}}F)(u_{jm}\Delta_j)$, where, as earlier, F is the number of operations per second that the access point can complete. In order to ensure that we complete the task of device $\breve{k}$, we must ensure that

$$\sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} (f_{jm\breve{k}}F)(u_{jm}\Delta_j) \geq W_{\breve{k}}. \tag{3.6}$$

Note that the first summation only goes to $\breve{k}$ and hence this constant implicitly ensures that we complete device $\breve{k}$'s task by the end of the $\breve{k}^{\text{th}}$ computing time slot. That is, by its computing deadline, $\breve{L}_{\breve{k}}$.

To complete the model for the computing resource allocation we need to determine $t_{\text{exe},\breve{k}}$, the length of the interval over which device $\breve{k}$'s task is completed. This can be written as $\breve{L}_{\breve{k}} - t_{\text{start},\breve{k}}$, where $t_{\text{start},\breve{k}}$ is the time at which the access point starts computing device $\breve{k}$'s task; see Fig. 3.1 for an illustration. If we define $\mathbb{I}(\cdot)$ to be the indicator function, with $\mathbb{I}(x) = 0$ if $x \leq 0$ and $\mathbb{I}(x) = 1$ if $x > 0$, then we can write

$$t_{\text{start},\breve{k}} = \breve{L}_{\breve{k}} - \sum_{j=1}^{\breve{k}} \sum_{m=0}^{K+1-j} u_{jm}\Delta_j \mathbb{I}\Big(\sum_{p=1}^{j}\sum_{q=0}^{m} f_{pq\breve{k}}\Big). \tag{3.7}$$

3.2.3 Communication model for offloading devices

We will model the narrowband communication channel from each (single-antenna) device to the (single-antenna) access point using a conventional discrete-time baseband equivalent channel model with symbol interval T_s. The channels are modelled as being quasi-static and flat in frequency. We will assume coherent communication in which the access point has knowledge of the baseband equivalent channel.

The access point employs the time domain multiple access (TDMA) scheme, and hence time is divided into $\breve{K}$ communication time slots, and only one of the $\breve{K}$ devices selected for offloading will transmit in each time slot. Given the indexing of the offloading devices in (3.3), we let $\pi(\breve{k})$ denote the index of the communication time slot in which device $\breve{k}$ transmits. Given an arbitrary channel use within the i^{th} communication time slot, device $\breve{k} = \pi^{-1}(i)$ will be transmitting, and the received signal takes the form

$$y = h_{\breve{k}}\sqrt{P_{\breve{k}}}s_{\breve{k}} + v, \tag{3.8}$$

where $s_{\breve{k}}$ in the symbol transmitted by the $\breve{k}^{\text{th}}$ device, which is normalized such that $E\{|s_{\breve{k}}|^2\} = 1$, $P_{\breve{k}}$ is the transmission power (per channel use) of the $\breve{k}^{\text{th}}$ device, $h_{\breve{k}}$ is the (complex-valued) channel gain of the $\breve{k}^{\text{th}}$ device, and v is the circular zero-mean additive white Gaussian noise at the access point, which has a variance of σ^2. For convenience, we will define the channel-to-noise ratio for device $\breve{k}$ to be $\alpha_{\breve{k}} = |h_{\breve{k}}^2|/\sigma^2$. The maximum allowable transmission power of device $\breve{k}$ is $\bar{P}_{\breve{k}}$.

If we denote the length of the i^{th} communication time slot, in channel uses, as τ_i, then in order to ensure that we can communicate the description of the $\breve{k}^{\text{th}}$

device's computation task to the access point we must ensure that

$$R_{\breve{k}}\tau_{\pi(\breve{k})} \geq B_{\breve{k}}, \tag{3.9}$$

where $R_{\breve{k}}$ is the data rate (per channel use) employed by the $\breve{k}^{th}$ device. There is a fundamental limit on the rate at which reliable communication can be achieved over the channel in (3.8); e.g., Polyanskiy et al. (2010). In this chapter we will employ the SNR-gap approximation that limit (e.g., Cioffi et al. (1995) and Starr et al. (1999)) in order to incorporate some of the effects of finite-block-length communication and the use of practical codes. That is, given an SNR-gap $\Gamma_{\breve{k}} \geq 1$ we will constrain $R_{\breve{k}}$ so that

$$0 \leq R_{\breve{k}} \leq \log_2\Big(1 + \frac{\alpha_{\breve{k}}P_{\breve{k}}}{\Gamma_{\breve{k}}}\Big). \tag{3.10}$$

The amount of energy that device $\breve{k}$ expands in offloading its task is $P_{\breve{k}}\tau_{\pi(\breve{k})}$. The amount of time it takes for device $\breve{k}$ to deliver its task to the access point is the sum of the time it needs to wait for access to the channel, $T_s\sum_{i=1}^{\pi(\breve{k})-1}\tau_i$, and the time it takes to transmit, $\tau_{\pi(\breve{k})}T_s$. That is,

$$t_{\text{del},k} = T_s\sum_{i=1}^{\pi(\breve{k})}\tau_i. \tag{3.11}$$

The remaining constraint on our offloading system is that computing cannot start until the full description of the task is delivered to the access point. That is, for each $\breve{k}$ we require $t_{\text{start},\breve{k}} \geq t_{\text{del},\breve{k}}$. With this model in place, an example of a joint computing–communication resource allocation for a case of $\breve{K} = 3$ devices is

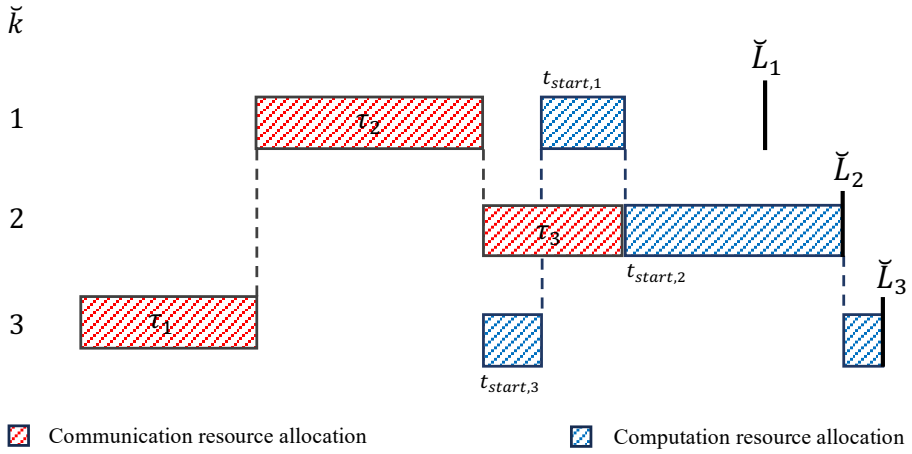


FIGURE 3.3: An example of a joint computation–communication resource allocation in a three-device case. In this case $\pi(1) = 2, \pi(2) = 3$, and $\pi(3) = 1$

illustrated in Fig. 3.3.

3.2.4 The "complete offloading" problem

Having established the computing and communication models for the offloading devices, we can now formulate the joint computing–communication resource allocation problem for a set of devices that have been selected for offloading. We will call this problem the complete offloading problem. In particular, given a set of $\breve{K}$ devices that have been re-indexed according to (3.3), and given the price per unit of energy at each device, $\lambda_{\breve{k}}$, we seek to jointly optimize the computing resource allocations in each computing subslot, $\left\{f_{jm\breve{k}} \in \{0,1\}\right\}_{j=1,m=0,\breve{k}=j}^{\breve{K},\breve{K}+1-j,\breve{K}}$, the fractional length of each computing subslot $\left\{u_{jm}\right\}_{j=1,m=0}^{\breve{K},\breve{K}+1-j}$, the time at which the access point begins computing, $\breve{L}_0$, which determines the duration of the first computing slot

$\Delta_1 = \breve{L}_1 - \breve{L}_0$, the order in which the devices transmit to the access point $\pi(\cdot)$, and the rates, $\left\{R_{\breve{k}}\right\}_{\breve{k}=1}^{\breve{K}}$, transmission power, $\left\{P_{\breve{k}}\right\}_{\breve{k}=1}^{\breve{K}}$, and communication time slot lengths, $\left\{\tau_{\breve{k}}\right\}_{\breve{k}=1}^{\breve{K}}$, so as to minimize the cost of the energy expended by the offloading devices, subject to the computational task of each device being completed by its computing deadline $\breve{L}_{\breve{k}}$. That problem can be written as

$$\min \quad \sum_{\breve{k}=1}^{\breve{K}} \lambda_{\breve{k}} \tau_{\pi(\breve{k})} P_{\breve{k}} \tag{3.12a}$$

$$\text{s.t.} \quad \tau_{\breve{k}} > 0, \tag{3.12b}$$

$$R_{\breve{k}} \tau_{\pi(\breve{k})} \geq B_{\breve{k}}, \tag{3.12c}$$

$$0 \leq P_{\breve{k}} \leq \bar{P}_{\breve{k}}, \tag{3.12d}$$

$$0 \leq R_{\breve{k}} \leq \log_2\Big(1 + \alpha_{\breve{k}} P_{\breve{k}} / \Gamma_{\breve{k}}\Big), \tag{3.12e}$$

$$T_s \sum_{i=1}^{\pi(\breve{k})} \tau_i \leq t_{\mathrm{del},\breve{k}}, \tag{3.12f}$$

$$t_{\mathrm{start},\breve{k}} \geq t_{\mathrm{del},\breve{k}}, \tag{3.12g}$$

$$t_{\mathrm{start},\breve{k}} = \breve{L}_{\breve{k}} - \sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} u_{jm} \Delta_j \mathbb{I}\Big(\textstyle\sum_{p=1}^{j} \sum_{q=0}^{m} f_{pq\breve{k}}\Big), \tag{3.12h}$$

$$\sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} (u_{jm} \Delta_j)(f_{jm\breve{k}} F) \geq W_{\breve{k}}, \tag{3.12i}$$

$$\Delta_1 = \breve{L}_1 - \breve{L}_0, \tag{3.12j}$$

$$f_{jm\breve{k}} \in \{0,1\},\ \textstyle\sum_{\breve{k}} f_{jm\breve{k}} \leq 1,\ \sum_m f_{jm\breve{k}} \leq 1, \tag{3.12k}$$

$$u_{jm} \geq 0,\ \textstyle\sum_{m=0} u_{jm} = 1, \tag{3.12l}$$

where we have left it implicit in the constraints that the free indices range over all possible values, and for notational simplicity we have not explicitly listed the design variables in the formulation.

In the formulation in (3.12) we have used the redundant variables $\{t_{\mathrm{del},\breve{k}}\}$ and $\{t_{\mathrm{start},\breve{k}}\}$ and the constraint in (3.12g) to highlight how the computation resource allocation, which is characterized by the constants in (3.12i)–(3.12l), is connected to the communication resource allocation, which is characterized by the constraints in (3.12b)–(3.12g) and by the objective in (3.12a). The constraints in (3.12i) implicitly ensure that device $\breve{k}$'s task is assigned sufficient resources that it will be completed by $\breve{L}_{\breve{k}}$, and the constraint in (3.12h) characterizes when the access point must start working on device $\breve{k}$'s task in order to complete that task in time. The fact that the description of that task must arrive at the access point before that time is captured by (3.12g). The delivery time is captured by (3.12f), and the fact that the whole description must be received is captured by (3.12c). The constraint in (3.12e) ensures that sufficient power is allocated to support reliable communication at the chosen rates, and the constraint in (3.12d) captures the constraint on that power allocation. The objective is the sum of the cost of the energy used by each device.

3.2.5 The binary offloading problem

Now that we have formulated the problem of joint computing–communication resource allocation problem for a given set offloading devices, we can succinctly formulate the binary offloading problem that jointly selects the offloading devices with the joint computing–communication resource allocation. To do so, we let $J^{\star}_{\mathrm{off}}(\{\gamma_k\})$ denote the optimal value for the complete offloading problem in (3.12) for a given set of offloading decisions $\{\gamma_k\}$, and we notionally set $J^{*}_{\mathrm{off}}(\{\gamma_k\}) = +\infty$ wherever that problem is infeasible. (Recall that the formulation in (3.12) involves a re-indexing of the offloading devices according to (3.3).) The binary offloading

problem can then be formulated as

$$\min_{\gamma_k \in \{0,1\}_{k=1}^K} \quad J^*_{\text{off}}\Big(\{\gamma_k\}\Big) + \sum_{k=1}^{K} (1-\gamma_k)\lambda_k E_{\text{loc},k} \tag{3.13a}$$

$$\text{s.t.} \quad (1-\gamma_k) t_{\text{loc},k} \leq L_k. \tag{3.13b}$$

Since $E_{\text{loc},k}$ and $t_{\text{loc},k}$ are known constants, this formulation suggests a natural description of the binary offloading problem into a binary tree search over complete offloading problems. We will employ that decomposition, and will exploit the fact that the binary search admits a tree structure. As such, the main contribution of this chapter will be development of efficient algorithm for the complete offloading problem in (3.12).

3.3 Solution strategy for the complete offloading problem in (3.12)

The complete offloading problem in (3.12) is difficult to solve directly, due to the $\sum_{j=1}^{\breve{K}} (\breve{K}+2-j)(\breve{K}-j) = \frac{1}{6}\breve{K}(2\breve{K}^2+3\breve{K}-5)$ binary variables $f_{jm\breve{k}}$ describing the allocation of computing resources, and the $\breve{K}$-item permutation $\pi(\cdot)$ that determines the order in which the devices transmit. As we outline in App. 3.C, one approach to solving that problem is to perform an exhaustive search over the binary variables and the permutation, and for each instance solve a convex optimization problem over $\{R_{\breve{k}}\}$ and $\{u_{jm}\}$. However, the number of instances grows rapidly with $\breve{K}$. Therefore, in the following sections we will develop an efficient algorithm that yields good solutions in practice, as we will demonstrate in Sec. 3.6.

The algorithm is based on the observation that once we have selected a computation resource allocation, the optimal transmission order (for that computation resource allocation) can be easily determined and the communication resource allocation problem can be written as a convex optimization problem over the rates; see (3.20) in Sec. 3.5. Therefore, one possible gateway to an efficient algorithm for the complete offloading problem is to find an efficient algorithm for a good computing resource allocation.

The objective of the complete offloading problem is a weighted sum of the energy that each device expends in transmitting the description of its task to the access point. This energy is related to the computation resource allocation through the constraint in (3.12g), which states that the description of device k's task must be delivered to the access point before its computation is scheduled to start; i.e., $t_{\text{start},\breve{k}} \geq t_{\text{del},\breve{k}}$. For an individual device, the energy required to communicate a fixed number of bits is a decreasing function of the transmission block length. Therefore, in the development of a computing resource allocation heuristic, one appropriate strategy is seek to make each $t_{\text{start},\breve{k}}$ large. In App. 3.A, we show that the largest possible value for the smallest $t_{\text{start},k}$, that is, the largest possible value of $\breve{L}_0$, is fixed for a large class of good computing resource allocations that includes optimal allocations. Therefore, in our proposed heuristic we will construct a computing resource allocation that successively maximizes each $t_{\text{start},\breve{k}}$, starting from the largest. That is, if we define $\breve{S} = \{1, 2, \ldots, \breve{K}\}$ to be the set of re-indexed offloading devices, we will first allocate the computing resources in a way that

seeks to

$$\max \quad \max_{\breve{k}\in\breve{S}} \quad t_{\text{start},\breve{k}} \tag{3.14a}$$

$$\text{s.t.} \quad \sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} (u_{jm}\Delta_j)(f_{jm\breve{k}}F) \geq W_{\breve{k}}, \tag{3.14b}$$

$$f_{jm\breve{k}} \in \{0,1\},\ \textstyle\sum_{\breve{k}} f_{jm\breve{k}} \leq 1,\ \sum_m f_{jm\breve{k}} \leq 1, \tag{3.14c}$$

$$u_{jm} \geq 0,\ \textstyle\sum_{m=0} u_{jm} = 1, \tag{3.14d}$$

where the optimization variables are all $f_{jm\breve{k}}$ and all u_{jm}. As we will show in the next section, an optimal computing resource allocation for this problem can be efficiently found. If we let $\breve{k}^\star$ denote the index of the device whose $t_{\text{start},\breve{k}}$ is maximized this way, then the next step in our strategy is to seek to assign the remaining computational resources so as to

$$\max \quad \max_{\breve{k}\in\breve{S}\setminus\breve{k}^\star} t_{\text{start},\breve{k}} \tag{3.15a}$$

$$\text{s.t.} \quad \text{(3.14b)–(3.14d)}, \tag{3.15b}$$

where design variables are those $f_{jm\breve{k}}$ and u_{jm} that were not determined by the solution to (3.14). That process is repeated until all the computing resources are allocated.

As we will show in Sec. 3.6, this "sequentially maximized maximum delivery time" (SMMDT) heuristic produces computing resource allocations that enable the devices to offload their tasks with low energy. However, this computing resource allocation may require switching between the tasks of different devices. For scenarios in which the cost of such context switching cannot be neglected, in

Sec. 3.4.3 we develop an alternative heuristic that constrains the computing resource allocation so that each task is completed in one contiguous block, without the need for context switching. We will call that heuristic the contiguous SMMDT (C–SMMDT) heuristic.

3.4 Efficient heuristics for computation resource allocation

To develop our SMMDT heuristic, we will first develop an effective heuristic for the case in which all the computational deadlines $\breve{L}_{\breve{k}}$ are the same. In Sec. 3.4.2 we will use insight from that case to extend our approach for the case of different latencies. In Sec. 3.4.3 we offer a variation of our approach that requires contiguous computation resources and hence avoids context switching.

3.4.1 The equal computation deadline case

In the case where all the computation deadlines are the same (i.e., $\breve{L}_{\breve{k}} = \breve{L}$ for all $\breve{k}$), there is only one computation time slot. To determine the length of that time slot, and hence $\breve{L}_0$, the time at which computation must begin, we observe that the total number of operations that must be completed is

$$W_T = \sum_{\breve{k}=1}^{\breve{K}} W_{\breve{k}}. \tag{3.16}$$

Since the access point processes F operations per second, the largest possible value for $\breve{L}_0$, which leaves the largest possible amount of communication time for the device that starts its computation first, is obtained when there are no idle

computing slots. In that case,

$$\breve{L}_0 = \breve{L}_{0,\max} = \breve{L} - W_T/F. \tag{3.17}$$

Note that we have not yet determined which device must start its computation first. Indeed, we note that this optimized value of $\breve{L}_0$ is independent of the order in which the devices compute their tasks.

Having established the optimal value of $\breve{L}_0$ and the fact that there will be no idle computing time slot, the task that remains is to divide the interval $[\breve{L}_0, \breve{L}]$ into $\breve{K}$ computing subslots and to assign each device to a subslot. That is, we assign $\{f_{1m\breve{k}}\}$. The duration of the subslot assigned to device $\breve{k}$ is the minimum required to complete its computation, namely $\frac{W_{\breve{k}}}{F}$. Therefore, if $f_{1m\breve{k}} = 1$ then $u_{1m} = \frac{W_{\breve{k}}}{F\Delta_1}$, where $\Delta_1 = \breve{L} - \breve{L}_0 = W_T/F$. As described at the end of Sec. 3.3, the heuristic that will guide our allocation is to sequentially maximize the time that each user has to deliver the descriptions of its task to the access point, starting with the largest.

In order to follow the heuristic, the device assigned to the last computing subslot is that with the smallest number of operations to complete; i.e., if $\breve{k}_{\text{last}} = \arg\min_{\breve{k}\in\breve{S}} W_{\breve{k}}$, then $f_{1\breve{K}\breve{k}_{\text{last}}} = 1$ and $f_{1\breve{K}\breve{k}} = 0$ for all $\breve{k} \neq \breve{k}_{\text{last}}$. As a result of this assignment, the delivery time for the description of this task must satisfy $t_{\text{del},\breve{k}_{\text{last}}} \leq \breve{L} - \frac{W_{\breve{k}_{\text{last}}}}{F}$.

The device assigned to the second last computing time slot is that with the smallest number of operations to complete among the remaining devices; i.e., if $\breve{k}_{2^{\text{nd}}\text{last}} = \arg\min_{\breve{k}\in\breve{S}\setminus\{\breve{k}_{\text{last}}\}} W_{\breve{k}}$, then $f_{1(\breve{K}-1)\breve{k}_{2^{\text{nd}}\text{last}}} = 1$ and $f_{1(\breve{K}-1)\breve{k}} = 0$ for all

$\breve{k} \neq \breve{k}_{2^{\text{nd}}\text{last}}$. As a result, the delivery time for that device must satisfy $t_{\text{del},\breve{k}_{2^{\text{nd}}\text{last}}} \leq \breve{L} - \frac{W_{\breve{k}_{\text{last}}}}{F} - \frac{W_{\breve{k}_{2^{\text{nd}}\text{last}}}}{F}$. This process is then used recursively to assign computational resources to the tasks of the remaining devices, and in doing so, assign the upper bounds on the delivery times for the description of these tasks.

3.4.2 The case of different computational deadlines

In this section we extend the principles of the SMMDT heuristic to the case of different computing deadlines, $\breve{L}_1 \leq \breve{L}_2 \leq \cdots \leq \breve{L}_{\breve{K}}$. As outlined above, the broad philosophy can be described as follows: In order to allow as much time as possible for the description of each task to be delivered to the access point, we assign the computing resources as late as possible. When there are devices competing for these resources, the task with the least computing requirement is computed last.

To describe the details of the proposed heuristic, let us first consider the $\breve{K}^{\text{th}}$ computing time slot, which is of duration $\Delta_{\breve{K}} = \breve{L}_{\breve{K}} - \breve{L}_{\breve{K}-1}$. In this slot we can only assign computing resources to device $\breve{K}$, because the tasks of the other devices should be completed before $\breve{L}_{\breve{K}-1}$. If $W_{\breve{K}} \leq F\Delta_{\breve{K}}$ then the task of device $\breve{K}$ can be completed within the $\breve{K}^{\text{th}}$ computing time slot. In that case, the computing allocation within this time slot consists of an idle subslot of duration $\Delta_{\breve{K}} - \frac{W_{\breve{K}}}{F}$ followed by the computation of the task of device $\breve{K}$ in a subslot of duration $\frac{W_{\breve{K}}}{F}$. That is, $f_{\breve{K},1,\breve{K}} = 1$ and $u_{\breve{K},1} = \frac{W_{\breve{K}}}{\Delta_{\breve{K}} F}, f_{\breve{K},0,\breve{K}} = 0$, and $u_{\breve{K},0} = (1 - u_{\breve{K},1})$. If $W_{\breve{K}} > F\Delta_{\breve{K}}$, then we assign the whole of the last computing time slot to device $\breve{K}$; i.e., $f_{\breve{K},1,\breve{K}} = 1$, $u_{\breve{K},1} = 1$ and $u_{\breve{K},0} = 0$. With this assignment in place, the

number of operations that must be completed on device $\breve{K}$'s task prior to $\breve{L}_{\breve{K}-1}$ is

$$V_{\breve{K},\breve{K}-1} = \max\Big\{0, W_{\breve{K}} - F\Delta_{\breve{K}}\Big\}. \tag{3.18}$$

In the $(\breve{K}-1)^{\text{th}}$ computing time slot we have up to three subslots, an idle subslot, and two computing subslots, one for each of devices $\breve{K}-1$ and $\breve{K}$. The number of as yet unassigned operations for the devices' tasks are $V_{\breve{K},\breve{K}-1}$ in (3.18) and $V_{\breve{K}-1,\breve{K}-1} = W_{\breve{K}-1}$. The last subslot, subslot $(\breve{K}-1,2)$, is assigned to the task of the device with the smaller value of $V_{\breve{k},\breve{K}-1}$, with a tie going to device $\breve{K}$ in order to have contiguous computation. Let $\breve{k}_2$ denote the index of the selected device; i.e., $\breve{k}_2 = \arg\max_{\breve{k}\in\{\breve{K}-1,\breve{K}\}} V_{\breve{k},\breve{K}-1}$. If $V_{\breve{k}_2,\breve{K}-1} \geq \Delta_{\breve{K}-1}F$ then the whole computing slot is assigned to device $\breve{k}_2$; i.e., $f_{\breve{K}-1,2,\breve{k}_2} = 1$, $u_{\breve{K}-1,2} = 1$ and all other $f_{\breve{K}-1,m,\breve{k}}$ and $u_{\breve{K}-1,m}$ are set to zero. If $V_{\breve{k}_2,\breve{K}-1} < \Delta_{\breve{K}-1}F$, then $f_{\breve{K}-1,2,\breve{k}_2} = 1$, $u_{\breve{K}-1,2} = \frac{V_{\breve{k}_2,\breve{K}-1}}{F\Delta_{\breve{K}-1}}$, and we assign subslot 1 to the other device, indexed $\breve{k}_1$. If $V_{\breve{k}_1,\breve{K}-1} \geq (1-u_{\breve{K}-1,2})\Delta_{\breve{K}-1}F$, then the whole of the remainder of computing subslot 1 is assigned to device $\breve{k}_1$. Otherwise, device $\breve{k}_1$ completes its task within computing slot $\breve{K}-1$ and the first subslot of that computing time slot, namely subslot $(\breve{K}-1,0)$, will be idle. Following this allocation, the number of operations that must be completed prior to $\breve{L}_{\breve{K}-2}$ for each task are $V_{\breve{k},\breve{K}-2} = \max\{0, V_{\breve{k},\breve{K}-1} - \Delta_{\breve{K}-1}F\sum_{m=1}^{2} f_{\breve{K}-1,m,\breve{k}}u_{\breve{K}-1,m}\}$, for $\breve{k} = \breve{K}-1, \breve{K}$.

This allocation process is repeated recursively for computing time slots $\breve{K}-2 \geq j \geq 2$. In particular, in computing slot j, we must assign the computing resources amongst at most $\breve{K}+1-j$ devices, with as yet unassigned computing demands

$V_{\breve{k},j}$, $\breve{k} = j, j+1, \breve{K}$, where $V_{j,j} = W_j$ and for $\breve{k} > j$

$$V_{\breve{k},j} = \max\Big\{0, V_{\breve{k},j+1} - \Delta_{j+1} F \sum_{m=1}^{\breve{K}-j} f_{j+1,m,\breve{k}} u_{j+1,m}\Big\}. \tag{3.19}$$

Following our heuristic, computing subslot $(j, \breve{K}+1-j)$ is assigned to the device with the smallest positive value of $V_{\breve{k},j}$, which we will denote as device $\breve{k}_j^\star$. That is, if we let $\mathcal{V}_j = \{V_{\breve{k},j} | V_{\breve{k},j} > 0\}$ and $\mathcal{S}_j = \{\breve{k} | V_{\breve{k},j} \in \mathcal{V}_j\}$, then $\breve{k}_j^\star = \arg\min_{\breve{k}\in\mathcal{S}_j} V_{\breve{k},j}$. If $V_{\breve{k}_j^\star,j} \geq \Delta_j F$, then the whole computing slot is assigned to device $\breve{k}_j^\star$; i.e. $f_{j,\breve{K}+1-j,\breve{k}_j^\star} = 1$, $u_{j,\breve{K}+1-j} = 1$ and all other $f_{j,m,\breve{k}}$ and $u_{j,m}$ are set to zero. If $V_{\breve{k}_j^\star,j} < \Delta_j F$, then we set $f_{j,\breve{K}+1-j,\breve{k}_j^\star} = 1$ and $u_{j,\breve{K}+1-j} = \frac{V_{\breve{k}_j^\star,j}}{F\Delta_j}$. In that case, the remaining time in computing slot j is $\tilde{\Delta}_j = (1 - u_{j,\breve{K}+1-j})\Delta_j$. To assign the remaining computing resources, we recursively repeat this process with $\mathcal{S}_j \leftarrow \mathcal{S}_j \setminus \{\breve{k}_j^\star\}$ and Δ_j replaced by $\tilde{\Delta}_j$, until all the computing resources in the slot are allocated (i.e., $\tilde{\Delta}_j = 0$) or there are no more tasks to which we can assign resources (i.e., $\mathcal{S}_j = \emptyset$).

Once the computing resources in computing time slots $\breve{K}$ down to 2 have been allocated, we can allocate the computing resources in the first time slot by computing $V_{\breve{k},1}$ using (3.19), and then applying the procedure developed in Sec. 3.4.1, using each $V_{\breve{k},1}$ in place of $W_{\breve{k}}$, and setting $\breve{L} = \breve{L}_1$. Note that in App. 3.A we obtain an expression for the optimal value of $\breve{L}_0$. Moreover, we show that this starting point is not only optimal for SMMDT, but is also globally optimal. Furthermore, for the case of equal computing deadlines, this optimal value for $\breve{L}_0$ is achieved by all computing resource allocations that complete the tasks by $\breve{L}$ and have no idle time slots.

To enhance the clarify of the exposition, the above description of the sequentially maximized maximum delivery time (SMMDT) algorithm for computation resource allocation employs new variables $V_{j,\breve{k}}$, $\mathcal{V}_j$ and $\tilde{\Delta}_j$. A more memory efficient version of that algorithm that uses recursive updates is formally stated in Alg. 4.

Algorithm 4 : SMMDT computation resource allocation

Input data: $\breve{\mathcal{S}}$, $\breve{L}_0$, $\{\breve{L}_k\}$, $\{\Delta_k\}$, $\{W_k\}$, F
Step 1: Set $\mathcal{V} = \breve{\mathcal{S}}$, $\breve{K} = |\breve{\mathcal{S}}|$
Step 2: Set each element of $\{f_{j,m,k}\}_{j=1,m=1,k=1}^{\breve{K},\breve{K}-j+1,\breve{K}}$ and $\{u_{j,m}\}_{j=1,m=1}^{\breve{K},\breve{K}-j+1}$ to zero.
Step 3: Set $\delta = \breve{L}_{\breve{K}}$
for $j = \breve{K}, \breve{K}-1, \ldots, 1$, **do**
  Step 4: Set the counter, $m = j$
  Step 5: Determine eligible devices for the j th computing slot $\mathcal{C} = \mathcal{V} \backslash \{1, 2, \ldots, j-1\}$
  while $\breve{L}_j - \delta < \breve{L}_{j-1}$ **do**
    Step 6: Determine the device with the minimum remaining computing cost $\breve{d} = \arg\min_{\breve{k} \in \mathcal{C}} \{W_{\breve{k}}\}$
    if $\delta - W_{\breve{d}}/F > \breve{L}_{j-1}$ **then**
      Step 7: Set $f_{j,m,\breve{d}} = 1$ and $u_{j,m} = W_{\breve{d}}/(F\Delta_j)$
      Step 8: Set $t_{\text{start},\breve{d}} = \delta - W_{\breve{d}}/F$
      Step 9: Update $\mathcal{C}$, $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\breve{d}\}$
      Step 10: Update δ, $\delta \leftarrow \delta - W_{\breve{d}}/F$
      Step 11: Update the counter m, $m \leftarrow m - 1$
    else
      Step 12: Set $f_{j,m,\breve{d}} = 1$ and $u_{j,m} = 1$
      Step 13: Update $W_{\breve{d}}$, $W_{\breve{d}} \leftarrow W_{\breve{d}} - (\delta - \breve{L}_{j-1})F$
      Step 14: Update δ, $\delta \leftarrow \breve{L}_j - \Delta_j$
    end if
  end while
end for
Output: $\{t_{\text{start},\breve{k}}\}$, $\{f_{j,m,k}\}$, and $\{u_{j,m}\}$, for $j = 1, 2, \ldots, \breve{K}$, $m = 0, \ldots, j$, $\breve{k} = 1, \ldots, \breve{K}$

3.4.3 Contiguous computation resource allocation

In Sec. 3.6 we will show that the SMMDT heuristic generates computation resource allocations that yield low energy consumption amongst the offloading devices. However, the SMMDT heuristic may require switching between the tasks of different devices. For scenarios in which the impact of that context switching is significant, in this section we develop a variant of the SMMDT heuristic that provides contiguous computing resource allocation. We will call this heuristic the contiguous SMMDT heuristic (C–SMMDT). There are several ways in which this heuristic can be described. We will provide a description that emphasizes the connection to the SMMDT heuristic.

Like the SMMDT heuristic, we start from last computing time slot, where only device $\breve{K}$ can compute, and we employ the same computing resource allocation as the SMMDT heuristic. That is, if $W_{\breve{K}} \leq F\Delta_{\breve{K}}$ then the task of device $\breve{K}$ can be completed the $\breve{K}^{\text{th}}$ computing time slot, with the remaining time in that computing slot being idle. Hence, $f_{\breve{K},1,\breve{K}} = 1$, $u_{\breve{K},1} = \frac{W_{\breve{K}}}{\Delta_{\breve{K}} F}$, $f_{\breve{K},0,\breve{K}} = 0$, and $u_{\breve{K},0} = 1 - u_{\breve{K},1}$. If $W_{\breve{K}} > F\Delta_{\breve{K}}$ then we assign the whole of the last computing time slot to device $\breve{K}$.

For the $(\breve{K}-1)^{\text{th}}$ slot, as in the SMMDT heuristic we define $V_{\breve{k},\breve{K}-1}$ to be the number of operations on device $\breve{k}$'s task that must be completed before $\breve{L}_{\breve{K}-1}$. That is, $V_{\breve{K},\breve{K}-1}$ is given in (3.18) and $V_{\breve{K}-1,\breve{K}-1} = W_{\breve{K}-1}$. The difference between the heuristics lies in the choice of $\breve{k}_2$, the device whose task will be worked on during the last subslot of this computing slot, which is subslot $(\breve{K}-1, 2)$. If device $\breve{K}$'s task requires computation resources prior to $\breve{L}_{\breve{K}-1}$, then it is assigned in the last subslot to ensure contiguity. That is, if $V_{\breve{K},\breve{K}-1} > 0$, then we assign $\breve{k}_2 = \breve{K}$.

Otherwise, we set $\breve{k}_2 = \breve{K} - 1$. Once $\breve{k}_2$ has been assigned, the resources in the $(\breve{K}-1)^{\text{th}}$ computing slot are allocated in the same way as the SMMDT heuristic.

For computing slot j, $\breve{K}-2 \geq j \geq 2$, we determine the number of operations of device $\breve{k}$'s task that must be computed before $\breve{L}_j$ using $V_{j,j} = W_j$ and (3.19). As in the case of the $(\breve{K}-1)^{\text{th}}$ computing slot, the difference between the C–SMMDT heuristic and the original SMMDT heuristic lies in the choice of the device whose task will be worked on in the last subslot. In the contiguous case, if a device's task is to be worked on at the beginning of the $(j+1)^{\text{th}}$ computing slot, it will also be worked on at the end of the j^{th} computing slot. To state that more formally, let $m^{\star}_{j+1}$ denote the lowest indexed subslot in slot $j+1$ in which the access point is active. That is let $m^{\star}_{j+1} = \arg\min_m u_{j+1,m}$ subject to $u_{j+1,m} > 0$. Let $q^{\star}_{j+1}$ denote the index of device whose task is being computed in that slot. That is, let $q^{\star}_{j+1}$ be such that $f_{j+1,m^{\star}_{j+1},q^{\star}_{j+1}} = 1$. If device $q^{\star}_{j+1}$ has computation to complete prior to $\breve{L}_j$, then we assign the last computing subslot of computing slot j to the task of that device, and then proceed with the SMMDT heuristic. If device $q^{\star}_{j+1}$'s task is completed within the $(j+1)^{\text{th}}$ computing slot, then we proceed directly with the SMMDT heuristic. That is, if $V_{q^{\star}_{j+1},j} > 0$, then we set the first $k^{\star}_j = q^{\star}_{j+1}$ and proceed with the SMMDT heuristic. If $V_{q^{\star}_{j+1},j} = 0$, we proceed directly with the SMMDT heuristic. In the first computing slot we employ the analogous modification of the SMMDT heuristic.

A memory-efficient version of the C–SMMDT heuristic can be constructed by making the following modifications to Alg. 4:

- **Step 6:** Replace by "If $q > 0$ set $\breve{d} \leftarrow q$ else set $\breve{d} = \arg\min_{\breve{k}\in\mathcal{C}}\{W_{\breve{k}}\}$"

- **Step 10:** Add "Set $q \leftarrow 0$"
- **Step 14:** Add " Set $q \leftarrow \breve{d}$"

3.5 Optimal communication resource allocation

Once the computing resource allocation has been performed, we know the times by which the description of the tasks must be delivered to the access point; i.e., we know $t_{\text{stat},\breve{k}}$ in the constraint $t_{\text{del},\breve{k}} \leq t_{\text{start},\breve{k}}$. If we let $D_{\breve{k}} = t_{\text{start},\breve{k}}/T_s$ denote the communication deadline in channel uses, then the remaining optimization problem over the communication resources takes the form

$$\min_{\{R_{\breve{k}}\},\{P_{\breve{k}}\},\{\tau_{\breve{k}}\},\pi(\cdot)} \quad \sum_{\breve{k}=1}^{\breve{K}} \lambda_{\breve{k}} P_{\breve{k}} \tau_{\pi(\breve{k})} \tag{3.20a}$$

$$\text{s.t.} \quad \tau_{\breve{k}} > 0 \tag{3.20b}$$

$$R_{\breve{k}} \tau_{\pi(\breve{k})} \geq B_{\breve{k}} \tag{3.20c}$$

$$0 \leq P_{\breve{k}} \leq \bar{P}_{\breve{k}} \tag{3.20d}$$

$$0 \leq R_{\breve{k}} \leq \log_2\Big(1 + \alpha_{\breve{k}} \bar{P}_{\breve{k}}/\Gamma_{\breve{k}}\Big) \tag{3.20e}$$

$$\sum_{i=1}^{\pi(\breve{k})} \tau_i \leq D_{\breve{k}} \tag{3.20f}$$

This problem takes a similar to form those in Salmani and Davidson (2018), Liu et al. (2023), and Chapter 2, and from that work we know that the optimal transmission order is in the order of increasing communication deadlines. That is, $\pi_{\text{opt}}(\cdot)$ is such that

$$D_{\pi_{\text{opt}}^{-1}(1)} \leq D_{\pi_{\text{opt}}^{-1}(2)} \leq \cdots \leq D_{\pi_{\text{opt}}^{-1}(\breve{K})}. \tag{3.21}$$

To simplify our notation we will let $\tilde{k} = \pi_{\text{opt}}(\breve{k})$ denote a re-indexing of the offloading devices so that they are in increasing order of their communication deadlines.

The problem in (3.20) can be simplified by observing that at optimality the constraints in (3.20c) and the constraints on the right hand side of (3.20e) hold with equality. These observations, enable us to rewrite (3.20) as an optimization problem over either $\{R_{\breve{k}}\}, \{P_{\breve{k}}\}$, or $\{\tau_{\breve{k}}\}$ alone. The problem over $\{R_{\breve{k}}\}$ takes a similar form to that in (2.33), namely.

$$\min_{\{R_{\tilde{k}}\}} \quad \sum_{\tilde{k}=1}^{\tilde{K}} \lambda_{\tilde{k}} \frac{B_{\tilde{k}}\Gamma_{\tilde{k}}}{\alpha_{\tilde{k}}} \left(\frac{2^{R_{\tilde{k}}} - 1}{R_{\tilde{k}}} \right) \tag{3.22a}$$

$$\text{s.t.} \quad \sum_{\tilde{\ell}=1}^{\pi(\tilde{k})} \left(\frac{B_{\tilde{\ell}}}{R_{\tilde{\ell}}} \right) \leq D_{\tilde{k}} \tag{3.22b}$$

$$0 \leq R_{\tilde{k}} \leq \log_2 \left(1 + \alpha_{\tilde{k}} \bar{P}_{\tilde{k}} / \Gamma_{\tilde{k}} \right) \tag{3.22c}$$

where the index $\tilde{\ell}$ in (3.22b) indicates that the devices are ordered in increasing order of their communication deadlines. By analytically evaluating the second derivatives of the objective in (3.22a) and the left hand side of the constraint in (3.22c) it can be shown that this problem is convex. Furthermore, it is feasible if and only if (3.22b) is satisfied when each $R_{\tilde{k}}$ achieves its upper bound in (3.22c). Once the problem in (3.22b) has been solved for $\{R^{\star}_{\tilde{k}}\}$ the optimal time slot lengths for (3.20) are $\tau^{\star}_{\tilde{k}} = B_{\tilde{k}}/R^{\star}_{\tilde{k}}$, and the optimal powers are $P^{\star}_{\tilde{k}} = \frac{\Gamma_{\tilde{k}} \left(2^{R^{\star}_{\tilde{k}}} - 1 \right)}{\alpha_{\tilde{k}}}$.

3.6 Simulation results

In this section, we examine the performance of the proposed joint computing–communication resource allocation algorithm in scenarios involving $K = 3$ and

4 devices, in which each device communicates over a quasi-static fading channel model. The channel model includes a path loss exponent of 3.71 and Rayleigh small scale fading. The reference gain at a distance of 100m is -20 dB, and unless otherwise specified, the users are randomly positioned at distances between 100m and 1,000m from the access point, following a uniform distribution. The symbol interval is $T_s = 10^{-6}$s, the noise power spectral density is -173 dBm/Hz, and each device has a transmission power constraint of $\bar{P}_k = 4\mu$J per sample. The overall latencies of the devices, denoted as L_k, are 480, 500, and 510 ms, respectively. We set the downlink communication time, $t_{\mathrm{DL},k}$, to 10 ms for all devices and hence deadlines are $\breve{L}_{\breve{k}} = 470, 490$ and 500 ms, respectively. The description lengths of the tasks are set to $B_k = 1{,}000$ bits for all devices. The number of operations required to complete each task is $W_k = 3{,}000$, 3,500, 4,000, for $k = 1, 2, 3$ respectively, and we will examine the system performance as F, the number of such operations that the access point can process per second, changes. (We have left the definition of what constitutes an "operation" implicit.) For the purpose of this section, we normalize the price of the energy cost at each device to one, represented by $\lambda_k = 1$.

3.6.1 Complete offloading

We will first assess the performance of the proposed approaches in a three-device complete offloading scenario; i.e., $\breve{K} = K = 3$. Since the devices are already ordered in increasing order of their computing deadlines, $\breve{L}_{\breve{k}}$, we do not need to re-index them in order to satisfy (3.3).

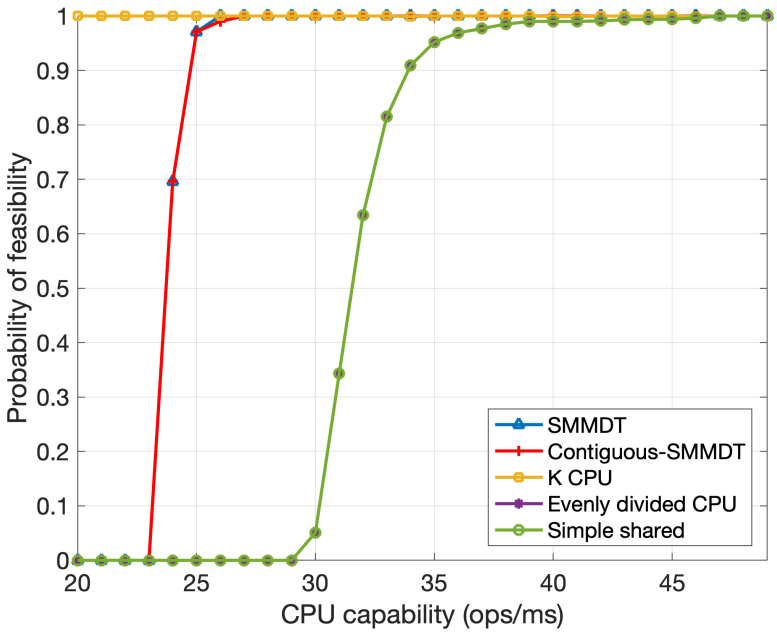


FIGURE 3.4: Probability of feasibility versus CPU capability.

In Fig. 3.4 we show how the fraction of feasible solutions varies with the computing capabilities of the access point (AP). These curves were obtained from 1,000 realizations of the fading channels. As expected, as the computing capabilities improve, the system can allocate more time for communication. That increases the chance that the problem descriptions can be reliably communicated over a given channel realization under the imposed power constraints. Fig. 3.4 shows that the greater flexibility of the SMMDT heuristic enables it to provide feasible solutions for a slightly larger fraction of the fading channels than the contiguous SMMDT when the computational capability is around 25 ops/ms. However both provide a much greater fraction of feasible solutions than two simpler benchmark heuristics that we have selected. The first benchmark involves reserving once third of the

computing capability for each device at all times; i.e., equally dividing the computing resources in a static manner. The second benchmark involves reserving equal fractions of the computing resources for the devices that may need them in each computing time slot; i.e., between $\breve{L}_2$ and $\breve{L}_3$ all the computing resources are allocated to device 3, between $\breve{L}_1$ and $\breve{L}_2$ they are equally divided between devices 2 and 3, and prior to $\breve{L}_1$ they are equally divided among all three devices. These benchmarks will be referred to as the evenly divided and simple shared benchmarks, respectively. We have also included the feasibility rate of a system in which each device has been allocated to its own dedicated processor capable of processing F operations per second. This benchmark will be referred to as the K CPU benchmark. While the performance of the K CPU system is not necessarily achievable by the system that we consider (which has only one processor capable of F operations per second), this benchmark does provide a useful bound on the achievable performance.

While Fig. 3.4 plots the fraction of channels for which a feasible solution is generated, Fig. 3.5 plots the average offloading energy. Since the offloading energy of an infeasible case is notionally infinite, we have only plotted the curves for values of the computational capability F for which all 1,000 realizations of the fading channel generated a feasible solution. Fig. 3.5 shows that the SMMDT heuristic is able to offload the tasks using an energy that is lower than that of the contiguous-SMMDT heuristic. Indeed, the SMMDT scheme bridges a significant fraction of the gap between the energy of the C–SMMDT scheme and that of the not-necessarily-achievable lower bound provided by the scheme with K processors. In fact, once the processor capability is above 42 ops/ms, the SMMDT scheme

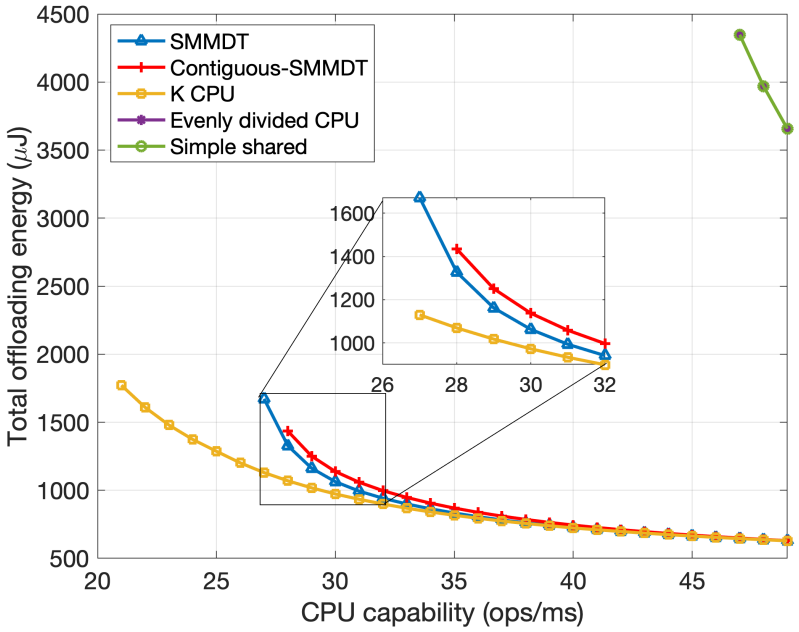


FIGURE 3.5: Average overall energy over all 1,000 channel realizations versus CPU capability

actually achieves the not-necessarily-achievable lower bound. That said, the K-CPU system is able to provide a feasible solution for a broader range of processor capabilities; see also Fig. 3.4. As F increases further, the performance of the C–SMMDT heuristic also approaches the lower bound.

To provide more context for these results, in Fig. 3.6 we provide a variation on the results in Fig. 3.5 in which the average is taken over the 900 channels for which the "evenly divided CPU" scheme is able to provide a feasible solution for a processor capability of $F \geq 34$ ops/ms; see Fig. 3.4. Figs. 3.4–3.6 demonstrate that the utilization of the proposed heuristics allows us to achieve two main advantages. First, they enhance the feasibility of the problem for lower values of the computational capability, F. Second, they notably reduce the cost of offloading

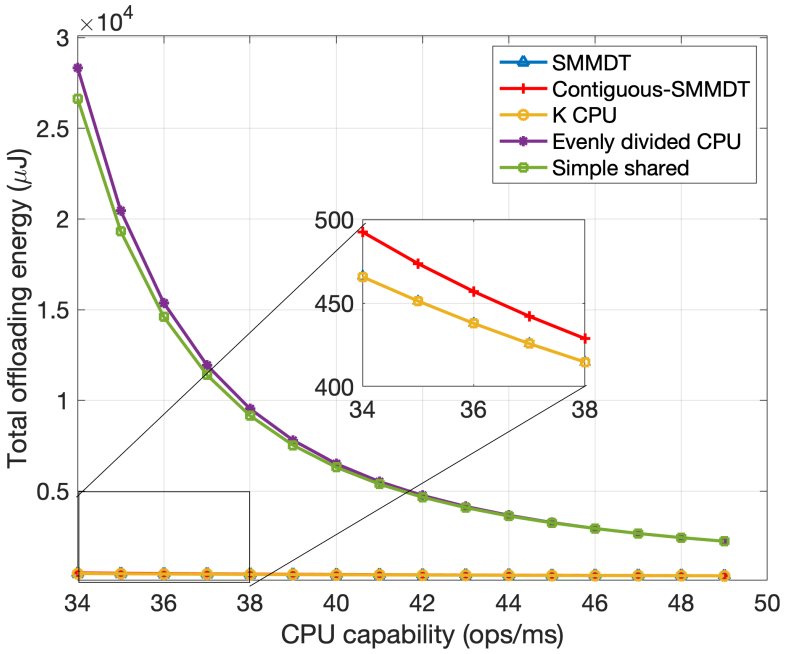


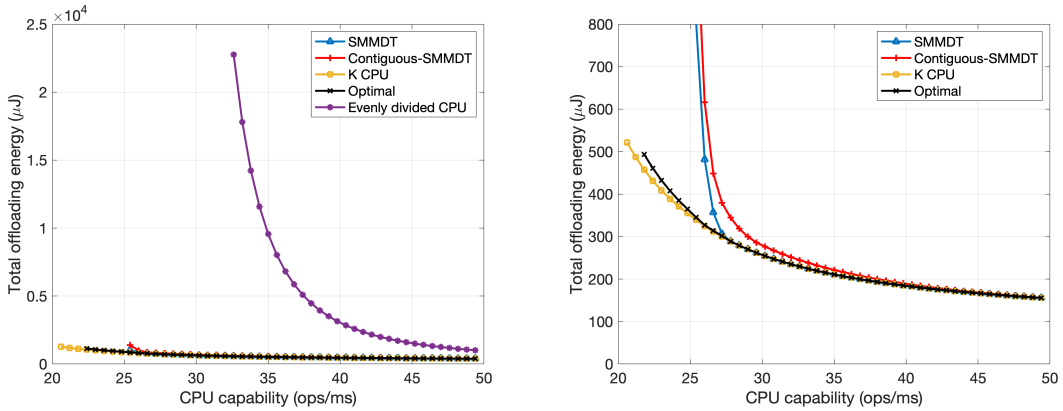
FIGURE 3.6: Average overall energy over a set of 900 of the 1,000 channel realizations versus CPU capability

energy in comparison to the benchmarks. Through a comparison of the performance between the proposed heuristics and the K-CPU lower bound, it becomes evident that the proposed heuristics can achieve performance levels quite close to those of the lower bound while only requiring one CPU at the access point.

3.6.2 Optimal allocation in the complete offloading case

Given the good performance of the proposed resource allocation schemes in the above example, it is appropriate to compare the performance of those schemes to that of the optimal joint computing–communication resource allocation. As we outlined in App. 3.C, the optimal allocation can be found by performing an exhaustive search over all possible choices of the computing subslot allocations, $f_{jm\check{k}}$,

and the communication ordering, $\pi(\cdot)$. For each choice, a reduced-dimension convex optimization problem is solved to determine the remaining parameters. Since finding the optimal solution for a given channel realization requires considerable computational effort (see App. 3.C), averaging over a reasonable number of channel realization is not possible in a reasonable amount of time. Instead, in this section we consider two different channel realizations in which we have fixed the channel gains for the first two devices to $\alpha_1 = |h_1|^2/\sigma_e^2 = 300$ and $\alpha_2 = |h_2|^2/\sigma_e^2 = 400$, respectively. For the first scenario, we set the channel gain of the third device to be large, namely $\alpha_3 = |h_3|^2/\sigma_e^2 = 500$, and for the second experiment we decrease α_3 by 10 dB; i.e., $\alpha_3 = 50$. As Fig. 3.5 has shown that the simple shared and evenly divided CPU bookmarks provide similar performance in the experiments, in this section we will only employ the evenly divided CPU scheme as our benchmark.



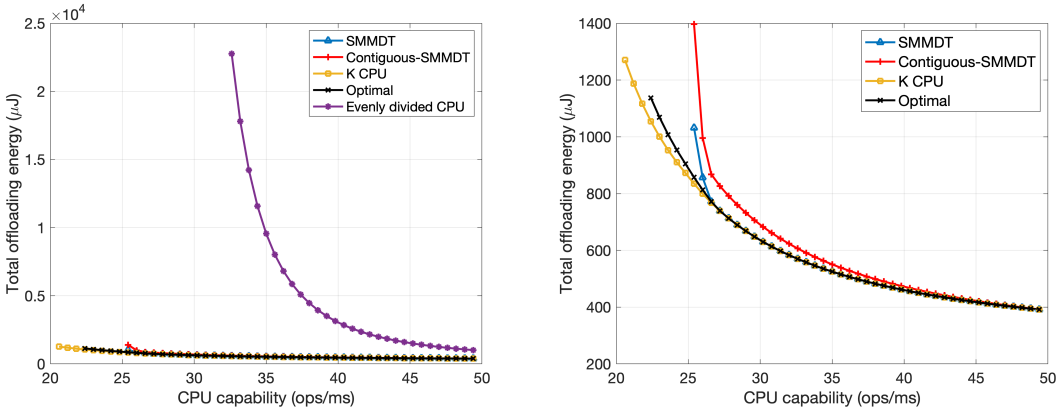
(a) Including the "evenly divided CPU" curve.

(b) Excluding the "evenly divided CPU" curve.

FIGURE 3.7: Offloading energy vs CPU capability for the case that $\alpha_1 = 300, \alpha_2 = 400$, and $\alpha_3 = 500$.

Fig. 3.7(a) shows that the SMMDT and C–SMMDT heuristics provide a feasible solution to the complete offloading problem for computational capabilities as low as

24.5 ops/ms. That is quite close to the feasibility threshold of the optimal scheme, which is 21.8 ops/ms and is significantly lower than that of the evenly divided benchmark, which is 30.8 ops/ms. In terms of the offloading energy, the SMMDT scheme achieves the same energy as the optimal scheme for all computational capabilities at or above 27.8 ops/ms, and they both achieve the K-CPU lower bound. In contrast, the C–SMMDT does not attain the performance of the optimal scheme for any value of F in the figure. Indeed, it is noteworthy that the SMMDT heuristic consistently outperforms the C–SMMDT heuristic.



(a) Including the "evenly divided CPU" curve.

(b) Excluding the "evenly divided CPU" curve.

FIGURE 3.8: Offloading energy vs CPU capability for the case that $\alpha_1 = 300, \alpha_2 = 400$, and $\alpha_3 = 50$.

For the second scenario, the α_3 is decreased 10 dB, and the computational capability thresholds above which each scheme provides a feasible solution are degraded a little. As shown in Fig. 3.8(b), the optimal solution is able to offload the tasks when $F \geq 22.4$ ops/ms, while the SMMDT and C–SMMDT heuristics are able to offload the tasks when $F \geq 25.2$ ops/ms, and benchmark does that when

32.6 ops/ms. Furthermore, as this scenario has a lower gain for device 3, offloading requires more energy than in the scenario in Fig. 3.7. In this more difficult scenario, the performance advantage of the proposed SMMDT heuristic over its contiguous counter part is larger than in Fig. 3.7. In particular, as in Fig. 3.7 the SMMDT scheme achieves the same performance as the K-CPU lower bound when $F \geq 27.8$ ops/ms. This is because it generates the same communication deadlines as the K-CPU scheme.

3.6.3 Binary offloading

When some or all of the devices have the capability to complete their task locally (i.e., $t_{\text{loc},k} \leq L_k$), the system has the additional degree of freedom to decide which devices should offload their tasks. In this section, we will demonstrate the performance of binary offloading under the same channel model as the previous section, for $K = 4$ devices, with latencies of $L_k = 410, 440, 450, 460$ms for $k = 1, 2, 3, 4$, downlink communication times $t_{\text{DL},k} = 10$ms, energy consumption of local computation $E_{\text{loc},k} = 200\mu$J for each user, computation requirements of $W_k =$ 2,500, 3,000, 3,500, 4,000 operations for $k = 1, 2, 3, 4$, and with local computation being feasible for all devices.

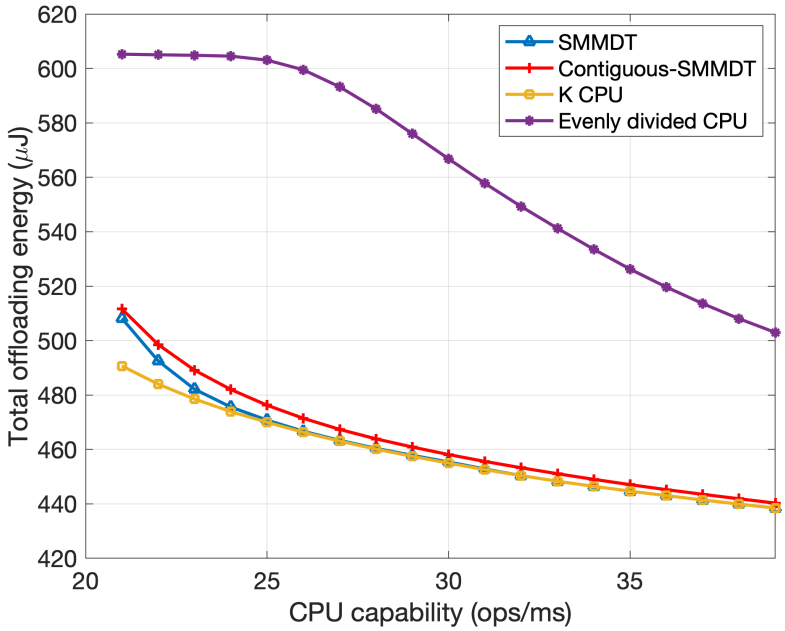
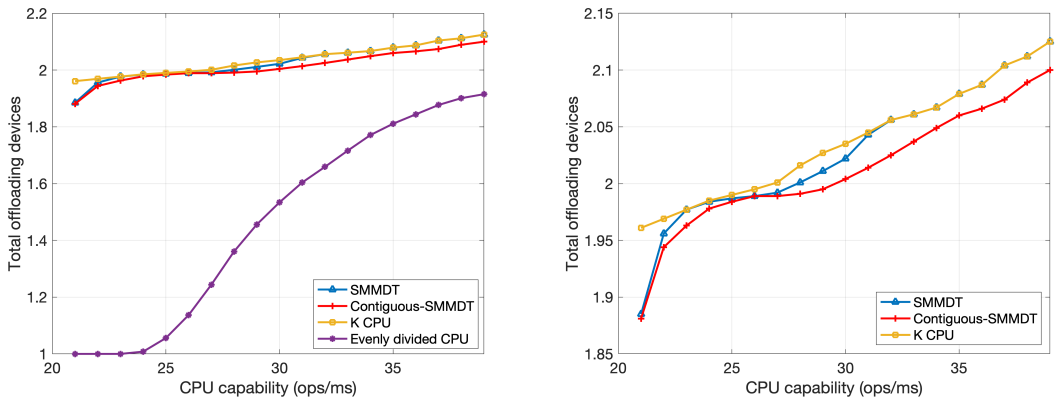


FIGURE 3.9: Overall energy versus CPU capability for binary offloading problem

In Fig. 3.9, we present the variation of the average energy expended by the devices with the computational capability of the access point. As local computation remains possible, binary offloading is feasible across all channel realizations, resulting in the curves being averaged over all 1,000 channel realizations. In Fig. 3.9, the curve for the SMMDT heuristic consistently outperforms the C–SMMDT heuristic. For lower CPU capabilities, the proposed heuristics expend more energy compared to the not-necessarily-achievable K-CPU lower bound. This is because the K-CPU lower bound enables more devices to offload their tasks. However, for higher CPU capabilities this energy gap is substantially reduced. For the case of $F \geq 33$ ops/ms, the SMMDT heuristic and the lower bound provide identical performance, while the C–SMMDT heuristic closely follows these two curves. This suggests that

the offloading sets for SMMDT heuristic and K-CPU lower bound are the same, even though the proposed scheme requires only one CPU at the access point; see also Fig. 3.10, below. Furthermore, the energy consumption associated with the evenly divided CPU benchmark is significantly larger than that of the proposed heuristics. As the computational capacity of the access point increases, the energy gap between the evenly-divided benchmark and the heuristics slowly decreases, but it remains large.



(a) Including the "evenly divided CPU" curve.

(b) Excluding the "evenly divided CPU" curve.

FIGURE 3.10: Number of offloading devices versus CPU capability for binary offloading problem.

To gain further insight into the different offloading energies among the considered schemes, Fig. 3.10 shows the variation of the average number of offloading devices as a function of the computational capability of the access point. Fig. 3.10(b) demonstrates that for $F \geq 33$ ops/ms the SMMDT heuristic offloads the same number of devices (on average) as the K-CPU upper bound, and that for $21 \leq F \leq 33$ it offloads a similar number of devices. The C–SMMDT heuristic

has the additional constraint of contiguity of computation and offloads a slightly smaller number of devices (on average) across all CPU capabilities. Indeed, the C–SMMDT heuristic never reaches the same number of offloading devices as the SMMDT heuristic nor the K-CPU upper bound, even at higher CPU capabilities. Fig. 3.10(a) illustrates that the evenly-divided-CPU benchmark only offloads a single device to the access point for $F < 22$ ops/ms, whereas the proposed heuristic offloads more than 1.9 devices on average at $F = 22$ ops/ms. Additionally, the benchmark's performance is significantly inferior in comparison to the proposed heuristics and the K-CPU upper bound. This trend true even when the access point employs a CPU that is capable of more than 40 ops/ms.

3.7 Conclusion

In conclusion, this chapter emphasizes the importance of careful modelling of the temporal structure of computation offloading systems and of the development of effective algorithms for joint allocation of computing and communication resources To address the challenges of joint resource allocation, we first considered the complete offloading problem, for which the set of offloading devices has already been selected. For that problem, we adopt a decomposition approach, introducing effective heuristics for computation resource allocation and optimal communication resource allocation within the TDMA scheme. Our approach is versatile, extending naturally to handle the binary offloading problem. Through numerical evaluations, we demonstrate the efficacy of our proposed algorithms. The results show that our decomposition and resource allocation strategies can rival the performance of joint optimization, especially as the computational capability of the access point

increases, and that it can even approach the performance of the lower bound obtained by giving the access point one independent processor per device.

3.A Optimal starting point for computation

In this appendix we derive an expression for the largest possible value for $\breve{L}_0$, the time at which the access point must begin computation of the one of the tasks in order to be able to complete all of their tasks by their computational deadlines, $\breve{L}_k$. We will show that this value can be computed from the parameters of the scenario, prior to any resource allocation.

To begin, let us consider the case of $\breve{K} = 1$ offloading device. In this case, we simply allocate all the available computing resources to the task of that device, and the largest possible value for $\breve{L}_0$ is

$$\breve{L}_{0,\max} = \breve{L}_1 - W_1/F. \tag{3.23}$$

In the case of the $\breve{K} = 2$ offloading devices, two possible scenarios arise. If device 2's task is completed within computing time slot 2, then computing time slot 2 will consist of an idle subslot followed by the completion of the task of device 2. In that case, the value of $\breve{L}_{0,\max}$ is determined by the computational requirements of device 1 and takes the form in (3.23). If some portion of device 2's task needs to be completed prior to $\breve{L}_1$, then there will be no idle computing subslots and $\breve{L}_{0,\max} = \breve{L}_2 - (W_1 + W_2)/F$. Therefore in the case of two offloading devices we have

$$\breve{L}_{0,\max} = \min\Big\{\breve{L}_1 - W_1/F, \breve{L}_2 - (W_1 + W_2)/F\Big\}. \tag{3.24}$$

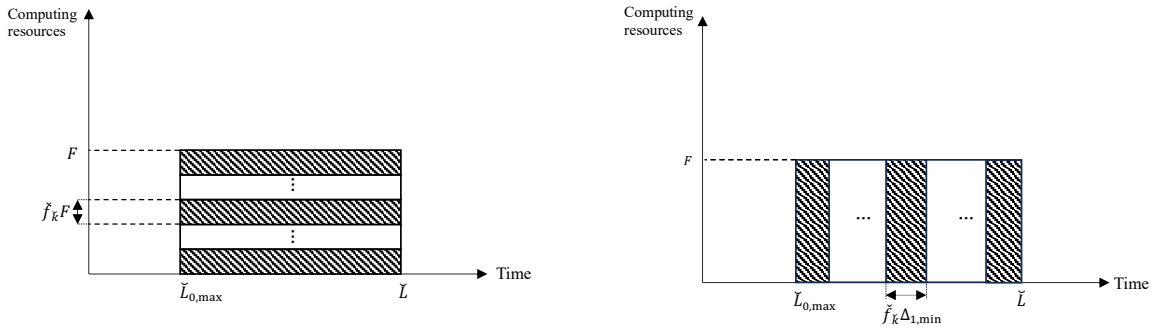
In the case of $\breve{K} = 3$ offloading devices, if the tasks of devices 2 and 3 can be completed after $\breve{L}_1$, then $\breve{L}_{0,\max}$ takes the form in (3.23). If that is not the case, but device 3's task can be completed after $\breve{L}_2$, then $\breve{L}_{0,\max}$ takes the form in (3.24). Otherwise, $\breve{L}_{0,\max} = \breve{L}_3 - (W_1 + W_2 + W_3)/F$. Hence, in the case of three offloading devices,

$$\breve{L}_{0,\max} = \min\Big\{\breve{L}_1 - W_1/F, \breve{L}_2 - (W_1 + W_2)/F, \breve{L}_3 - (W_1 + W_2 + W_3)/F\Big\}. \quad (3.25)$$

Using analogous arguments, for the general case of $\breve{K}$ offloading devices we have

$$\breve{L}_{0,\max} = \min_{\breve{k}\in\{1,2,\ldots,\breve{K}\}} \Big\{\breve{L}_{\breve{k}} - \frac{1}{F}\sum_{j=1}^{\breve{k}} W_j\Big\}. \quad (3.26)$$

3.B Optimality of allocating computation resources to one task at a time



(a) Sharing computation fractions.

(b) Sharing time fractions.

FIGURE 3.11: Two computing resource allocation schemes for a scenario with common computing deadlines.

In the most general form of a computing resource allocation in our setting, the computing resources can be shared between the tasks of multiple devices. In this appendix, we will show that adopting a simpler resource allocation strategy in which all the computing resources are allocated on the task of one device at a time is sufficient to obtain an optimal solution to the complete offloading problem. That is, we will show that the computing architecture described in Sec. 3.2.2 and illustrated in Fig. 3.2 is sufficient to obtain an optimal allocation.

We will first consider the case in which all the computing deadlines are the same; i.e., $\breve{L}_k = \breve{L}$. In this case there is only one computing time slot, and for optimal schemes that time slot is $[\breve{L}_{0,\max}, \breve{L}]$, where, as in App. 3.A, $\breve{L}_{0,\max} = \breve{L} - \frac{1}{F}\sum_{\breve{k}=1}^{\breve{K}} W_{\breve{k}}$. Let $\Delta_{1,\min} = \breve{L} - \breve{L}_{0,\max}$. One possible computing allocation that achieves $\breve{L}_{0,\max}$ is to allocate a fraction $\check{f}_{\check{k}} = \frac{W_{\check{k}}}{\sum_{\check{k}}^{\breve{K}} W_{\check{k}}}$ of the computing resources to device $\breve{k}$'s task for the whole of the computing time slot; see Fig. 3.11(a). However, doing so results in the access point having to start computing all of the devices' tasks at $\breve{L}_{0,\max}$. That means that the descriptions of the tasks of all devices must be delivered by $\breve{L}_{0,\max}$. An alternative allocation that also achieves $\breve{L}_{0,\max}$ is to allocate all the computing resources to device $\breve{k}$'s task for a fraction $\check{f}_{\check{k}}$ of the time interval $[\breve{L}_{0,\max}, \breve{L}]$; see Fig. 3.11(b). Both schemes enable all the devices' tasks to be completed by $\breve{L}$. In Fig. 3.11, this is apparent in the fact that the rectangles representing the number of operations assigned to device $\breve{k}$ have the same area. In the case in Fig. 3.11(b), since the computing resources are allocated to only one device at a time, only one device's task needs to be delivered by $\breve{L}_{0,\max}$. The communication deadlines for the other task descriptions are later. Since the energy required to reliably communicate a message over a given time interval is a decreasing function of the

length of that interval, the system can use that additional time to reduce the energy that is needed to offload those tasks.

To expand that discussion to the general case, let us consider an arbitrary time interval $[t_a, t_b]$ in which a subset $\mathcal{D} \subseteq \{1, 2, \ldots, \breve{K}\}$ of the tasks share the computing resource. Let $\Theta_{\breve{k}}$ denote the number of operations performed on device $\breve{k}$'s task during this interval. Note that the full description of all the tasks must be delivered prior to t_a. As suggested by the previous argument, there is a computationally equivalent system consisting of $|\mathcal{D}|$ subslots in which in each subslot one device's task is allocated all the computing resources. The width of the subslot for device $\breve{k}$ is $\frac{\Theta_{\breve{k}}}{F}(t_b - t_a)$. By t_b, this system achieves the same amount of progress in computing the devices' tasks as the shared case and hence it is sufficient for optimality. In the general case, the subslot model will allow more time for the task descriptions to be delivered, and hence it will facilitate lower energy expenditure by the devices.

With this argument in place we know that for any time interval in which the computational resource is shared, there is a time divided system in which only one device's task being worked on at a time, for which the optimal (cost of the) energy is no greater than that of the allocation with shared resources. Hence, it is sufficient to allocate the computing resources to one device at a time.

Having established that result, we now need to establish how many subslots we need to consider in each computing time slot.

In the last computing time slot $(\breve{L}_{\breve{K}-1}, \breve{L}_{\breve{K}}]$ there is at most one task to work on. Hence we need only consider two subslots: one in which the access point works on the task of device $\breve{K}$ and, if that task can be completed within the time slot, the

other subslot will be idle. Since the description of the task must be delivered to the access point prior to computing beginning, if there is a need for an idle time slot it will be the first time slot. In the $(\breve{K}-1)^{\text{th}}$ computing time slot we have at most two tasks to work on, namely device $(\breve{K}-1)$'s task and the component of device $\breve{K}$'s task that must be completed before $\breve{L}_{\breve{K}-1}$. Although we could assign many subslots in this computing time slot, and in doing so interleave the computation of the two tasks, the communication of the description of one of the tasks will have a greater impact on the total (cost of the) offloading energy. Therefore, that task should be worked on during the latter part of the computing slot, without interleaving, in order to make the delivery time for the description of that device's task as long as possible. Hence, we need at most three subslots in the $(\breve{K}-1)^{\text{th}}$ computing time slot, one for the task of device $(\breve{K}-1)$, one for the task of device $\breve{K}$, and one idle slot.

A similar argument enables us to show that in the j^{th} computing time slot we need only consider $(j+1)$ subslots, one for the task of each device and one as a possible idle time slot. Note that the duration of the subslots is a design variable, and with the exception of the idle subslot, the assignment of tasks to subslots is also a design variable. If an idle subslot arises, it arises because the tasks of devices $j, j+1, \ldots, \breve{K}$ can be completed after $\breve{L}_{j-1}$. That idle subslot will be the first subslot of the j^{th} computing time slot.

3.C Optimal solution for (3.12)

One strategy for finding the optimal solution to (3.12), is to determine the jointly optimal computing subslot lengths and communication resource allocation, for all

possible choices of the task to be worked on in each computing subslot, $\{f_{jm\breve{k}} \in \{0,1\}\}$ and all possible communication transmission orders $\pi(\cdot)$, and then to pick the best solution. For a given allocation of tasks to computing subslots and a given transmission order, the remaining optimization problem is over $\{u_{jm}\}$, $\breve{L}_0$, $\{P_k\}, \{R_k\}$, and $\{\tau_k\}$. Since we derived the optimal value of $\breve{L}_0$ in App. 3.A we can restrict attention to candidate solutions that achieve this value. Therefore, the remaining optimization problem can be reduced to the following problem over $\{u_{jm}\}$, $\{P_{\breve{k}}\}$, $\{R_{\breve{k}}\}$ and $\{\tau_{\breve{k}}\}$,

$$\min \quad \sum_{\breve{k}=1}^{\breve{K}} \lambda_{\breve{k}} \tau_{\pi(\breve{k})} P_{\breve{k}} \tag{3.27a}$$

$$\text{s.t.} \quad \tau_{\breve{k}} > 0, \tag{3.27b}$$

$$R_{\breve{k}} \tau_{\pi(\breve{k})} \geq B_{\breve{k}}, \tag{3.27c}$$

$$0 \leq P_{\breve{k}} \leq \bar{P}_{\breve{k}}, \tag{3.27d}$$

$$0 \leq R_{\breve{k}} \leq \log_2\Big(1 + \alpha_{\breve{k}} P_{\breve{k}} / \Gamma_{\breve{k}}\Big), \tag{3.27e}$$

$$T_s \sum_{i=1}^{\pi(\breve{k})} \tau_i \leq t_{\text{del},\breve{k}}, \tag{3.27f}$$

$$t_{\text{start},\breve{k}} \geq t_{\text{del},\breve{k}}, \tag{3.27g}$$

$$t_{\text{start},\breve{k}} = \breve{L}_{\breve{k}} - \sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} u_{jm} \Delta_j \Phi_{jm\breve{k}}, \tag{3.27h}$$

$$\sum_{j=1}^{\breve{k}} \sum_{m=0}^{\breve{K}+1-j} (u_{jm}\Delta_j)(f_{jm\breve{k}} F) \geq W_{\breve{k}}, \tag{3.27i}$$

$$u_{jm} \geq 0, \ \textstyle\sum_{m=0} u_{jm} = 1, \tag{3.27j}$$

where $\Phi_{jm\breve{k}} = \mathbb{I}\Big(\sum_{p=1}^{j}\sum_{q=0}^{m} f_{pq\breve{k}}\Big)$, is known since the $f_{jm\breve{k}}$ are fixed. Therefore, constraints (3.27h)–(3.27j) are linear in u_{jm} for all j and m.

We observe that $\pi(\cdot)$ is given, and that the computation and communication resource allocations only interact through (3.27g). Therefore, we can use the techniques that were used in App. 2.B to reduce the communication resource allocation part to an optimization over $\{R_k\}$ alone. That is, the problem in (3.27) can be simplified to

$$\min_{\{R_k\},\{u_{jm}\}} \quad \sum_{\breve{k}=1}^{\breve{K}} \lambda_{\breve{k}} \frac{B_{\breve{k}}}{\alpha_{\breve{k}}}\Big(\frac{2^{R_{\breve{k}}}-1}{R_{\breve{k}}}\Big) \tag{3.28a}$$

$$\text{s.t.} \quad 0 \le R_{\breve{k}} \le \log_2\Big(1+\alpha_{\breve{k}}\bar{P}_{\breve{k}}/\Gamma_{\breve{k}}\Big), \tag{3.28b}$$

$$\sum_{i=1}^{\breve{k}} \frac{B_i}{R_i} \le \Big(\breve{L}_{\breve{k}} - \sum_{j=1}^{\breve{k}}\sum_{m=0}^{\breve{K}+1-j} u_{jm}\Delta_j \Phi_{jm\breve{k}}\Big)/T_s, \tag{3.28c}$$

$$\sum_{j=1}^{\breve{k}}\sum_{m=0}^{\breve{K}+1-j} (u_{jm}\Delta_j)(f_{jm\breve{k}}F)) \ge W_{\breve{k}}, \tag{3.28d}$$

$$u_{jm} \ge 0,\ \textstyle\sum_{m=0} u_{jm} = 1, \tag{3.28e}$$

This problem is jointly convex in $\{R_{\breve{k}}\}$ and $\{u_{jm}\}$ and can be efficiently solved.

In this appendix, we have shown that one way to solve (3.27) optimally is to solve the problem in (3.28) for each computing resource allocation and each permutation. There are $\Pi_{i=1}^{\breve{K}}(\breve{K}+1-i)!$ possible choices for the set $\{f_{jm\breve{k}}\}$ and there are $\breve{K}!$ possible permutation of the transmission orders. As a result, the exhaustive search approach requires the solution of $\breve{K}!\Pi_{i=1}^{\breve{K}}(\breve{K}+1-i)!$ different instances of the convex problem in (3.28).

Chapter 4

Conclusion and Future Work

4.1 Conclusion

The effective utilization of shared computing resources at an access point relies on two key factors: the selection of offloading devices and the joint allocation of computing and communication resources. Most previous approaches to these problems employed classical asymptotic-block-length characterizations of fundamental rate limits to perform communication resource allocation. However, doing so overlooked the fact that the offloading devices' latency constraints inherently limited the available block length. Furthermore, many previous approaches employed rather simple models for the computing resource allocation, and this could impact the effectiveness of computational offloading. In this thesis, we developed resource allocation algorithms that addressed both of these issues.

In Chapter 2, we developed efficient algorithms for offloading decision-making and communication resource allocation in a K-device binary computational offloading system that used time-division multiple access (TDMA). The approach

incorporated fundamental rate limits on finite-block-length communication, and the algorithms utilized a relaxation-rounding technique based on a customized incremental rounding scheme for the transmission block lengths. This approach ensured that the process of rounding a feasible solution to the relaxed problem always generated a feasible integer block length. Additionally, we leveraged a closed-form approximation of transmission powers to simplify the optimization process and bridge the performance gaps between ad-hoc schemes and extensive search solutions.

In Chapter 3, we tackled the challenge of effectively utilizing shared computing resources at an access point. The combinational nature of the computation resource allocation made this a computationally daunting problem. Our approach was based on decomposing the joint computation and communication resource allocation problem. We first examined the "completed" offloading problem, in which the offloading decisions had already been made. For that problem, we proposed effective heuristics for computation resource allocation and showed how the optimal communication resource allocation for TDMA, for a given computation-resource allocation, could be obtained by solving a convex optimization problem over the rates alone. Our approach to the complete offloading problem was then incorporated into an efficient tree search algorithm for the binary offloading decisions. The numerical results demonstrated that the proposed decomposition and algorithmic approaches for computation resource allocation could achieve performance levels comparable to joint optimization, at much lower computational cost.

In conclusion, the thesis emphasized the importance of optimal offloading device

selection and joint resource allocation for effective multi-user computation offloading systems. We proposed novel algorithms and heuristic methods that provided effective sharing of the finite-block-length communication resources of the channel and the finite computing resources of the access point, and the proposed methods demonstrated significant improvements in performance compared to the existing approaches.

4.2 Future work

In this thesis, computationally-efficient algorithms have been proposed to achieve energy optimizing solutions for (i) the communication resource allocation problem for integer finite block length, and (ii) the joint communication and computation resource allocation in both the binary offloading and complete offloading cases of a K-user offloading system. The success of these algorithms suggest several additional directions that would be worthy of further investigation.

4.2.1 Communication model and resource allocation

In this thesis, in order to focus on the core principles of computation offloading, we focused on time division multiple access (TDMA) systems with a single antenna at the access point and single antenna devices. Although TDMA is simple to implement, it is well known that there are other multiple access schemes that have larger achievable rate regions; e.g., Cover and Thomas (2012). Superposition coding with successive decoding, which is also known as Non-Orthogonal Multiple Access; e.g., Dai et al. (2018) is one popular example. Therefore, a natural extension would be to consider such schemes. These schemes have been considered in the case

of asymptotically long block lengths (Salmani and Davidson (2018), Ding et al. (2022), and Liu et al. (2023)) under the simplified computation model in Chapter 2, and hence can be incorporated into the joint computation-communication resource allocation problem in Chapter 3 in a natural way. However, the case of finite block lengths is quite involved, as the capacity region has not yet been well approximated even in the two-user case (MolavianJazi and Laneman (2015)). That said, some early work on two-user NOMA schemes with fixed order sequential decoding (Liu et al. (2022) and Yang et al. (2022)) suggests methodologies that could be employed to develop pragmatic schemes.

Beyond the consideration of different multiple access schemes, it is well known that the provision of multiple antennas and the access point and, potentially at the devices, offers the opportunity for larger achievable rate regions than single antenna systems; e.g., Goldsmith et al. (2003). These larger rate regions offer the opportunity to reduce the energy consumed by each offloading device. While there has been some work on computation offloading using NOMA signalling with fixed-order sequential decoding and multiple antennas of the access point (Wang et al. (2019)), and NOMA signalling with independent decoding and multiple antennas at both the access point and the devices (Sardellitti et al. (2015)), these systems have been based on rate characterizations for asymptotically long block lengths. They have also been based on fixed computation resource allocation (Wang et al. (2019)) or a simplified scheme for allocating computing resources (Sardellitti et al. (2015)). Those observations suggest that the ideas developed in this thesis could be applied in these more general settings.

4.2.2 Computation resource allocation

In Chapter 3 of this thesis, we have addressed the problem of joint computation and communication resource allocation. with a goal of controlling the computational cost of solving the joint problem. We took a decomposition approach in which we developed two heuristic approaches for computation resource allocation. The emphasis was on finding an optimal solution independent of the communication constraints, such as channel gains, limits the transmission power of each device, the description length of each device's task, and the related price s of each device's energy.

Although the proposed decomposition based allocations perform well in practice, often close to the optimal allocation, incorporating information regarding the communication environment into the heuristic for computation resource allocation offers the potential for further enhancements. The additional information can lead to more informed decisions regarding computing resource allocation, and hence more appropriate communication deadlines, potentially improving the overall system performance.

4.2.3 Utilizing machine learning approaches

In computation offloading, there are numerous opportunities to leverage machine learning techniques, such as neural networks and reinforcement learning, as demonstrated by Li et al. (2018). In the context of this thesis, there are two possible machine learning approaches that warrant particular attention.

The first approach involves utilizing machine learning for decision-making regarding offloading devices. The resource allocation problem in the binary offloading case combines binary and continuous variables. Our current approach decomposes the problem into an inner energy minimization problem, focusing on continuous variables when binary offloading decisions are given, and an outer problem aimed at finding good binary offloading decisions. While our customized pruned greedy search algorithm achieves close-to-optimal binary offloading decisions with reasonable computational cost, it is advantageous to further reduce the computational overhead for binary decision making. One potential approach is to employ deep neural networks or reinforcement learning, trained to provide binary offloading decisions based on the system parameters, as demonstrated by Salmani et al. (2019).

Furthermore, for the computation resource allocation, we can also leverage the neural network approach. In the formulation of the joint computation–communication resource allocation problem, after decomposing the joint problem into computation and communication resource allocation, the ordering of the computation sub-slots is known. This ordering determines a binary set, and the constraints on the computation resource allocation problem become linear over the duration of each sub-slot. By training an appropriate machine learning model, we can enhance the computation resource allocation and, consequently, improve the overall performance of the system. This integration of neural networks in computation resource allocation holds promise for optimizing resource utilization and enhancing the efficiency of computation offloading systems.

Bibliography

Abbas, N., Zhang, Y., Taherkordi, A., and Skeie, T. (2018). Mobile Edge Computing: A survey. *IEEE Internet of Things J.* 5(1), 450–465.

Barbarossa, S., Sardellitti, S., and Di Lorenzo, P. (2014). Communicating while Computing: Distributed Mobile Cloud Computing over 5G Heterogeneous Networks. *IEEE Signal Process. Mag.* 31(6), 45–55.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Chen, M., Dong, M., and Liang, B. (2018). Resource Sharing of a Computing Access Point for Multi-User Mobile Cloud Offloading with Delay Constraints. *IEEE Trans. Mobile Comput.* 17(12), 2868–2881.

Cioffi, J. M., Dudevoir, G. P., Eyuboglu, M. V., and Forney, G. D. (1995). MMSE Decision-Feedback Equalizers and Coding—Part II: Coding Results. *IEEE Trans. Commun.* 43(10), 2582–2604.

Conway, J. H. and Guy, R. K. (1996). *The Book of Numbers*. Copernicus.

Cover, T. M. and Thomas, J. A. (2012). *Elements of Information Theory*. 2nd ed. John Wiley & Sons.

Dai, L., Wang, B., Ding, Z., Wang, Z., Chen, S., and Hanzo, L. (2018). A Survey of Non-Orthogonal Multiple Access for 5G. *IEEE Commun. Surveys & Tuts.* 20(3), 2294–2323.

Ding, Z., Xu, D., Schober, R., and Poor, H. V. (2022). Hybrid NOMA Offloading in Multi-User MEC Networks. *IEEE Trans. Wireless Commun.* 21(7), 5377–5391.

Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. *Wireless Commun. & Mob. Comput.* 13(18), 1587–1611.

Dinh, T. Q., Tang, J., La, Q. D., and Quek, T. Q. S. (2017). Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling. *IEEE Trans. Commun.* 65(8), 3571–3584.

Durisi, G., Koch, T., and Popovski, P. (2016). Toward Massive, Ultra-reliable, and Low-Latency Wireless Communication With Short Packets. *Proc. IEEE* 104(9), 1711–1726.

Fernando, N., Loke, S. W., and Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems* 29(1), 84–106.

Goldsmith, A., Jafar, S., Jindal, N., and Vishwanath, S. (2003). Capacity limits of MIMO channels. *IEEE J. Sel. Areas Commun.* 21(5), 684–702.

Hong, M., Razaviyayn, M., Luo, Z.-Q., and Pang, J.-S. (2016). A Unified Algorithmic Framework for Block-Structured Optimization Involving Big Data. *IEEE Signal Process. Mag.* 33(1), 57–77.

Khan, A. R., Othman, M., Madani, S. A., and Khan, S. U. (2014). A Survey of Mobile Cloud Computing Application Models. *IEEE Commun. Surveys & Tut.* 16(1), 393–413.

Kumar, K., Liu, J., Lu, Y.-H., and Bhargava, B. (2013). A Survey of Computation Offloading for Mobile Systems. *Mob. Netw. Appl.* 18(1), 129–140.

Kumar, K. and Lu, Y.-H. (2010). Cloud Computing for Mobile Users: Can Offloading Computation save Energy? *IEEE Computer* 43(4), 51–56.

Li, J., Gao, H., Lv, T., and Lu, Y. (2018). Deep Reinforcement Learning-Based Computation Offloading and Resource Allocation for Mobile Edge Computing (MEC). In: *Proc. IEEE Wireless Commun. & Netw. Conf.*

Liu, X., Schaible, C., and Davidson, T. N. (2023). Multiple Access Computation Offloading for the K-User Case. In: *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*

Liu, Z., Zhu, Y., Hu, Y., Sun, P., and Schmeink, A. (2022). Reliability-Oriented Design Framework in NOMA-Assisted Mobile Edge Computing. *IEEE Access* 10, 103598–103609.

Mach, P. and Becvar, Z. (2017). Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surveys & Tuts.* 19(3), 1628–1656.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017a). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surveys & Tuts.* 19(4), 2322–2358.

Mao, Y., Zhang, J., Song, S. H., and Letaief, K. B. (2017b). Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems. *IEEE Trans. Wireless Commun.* 16(9), 5994–6009.

MolavianJazi, E. and Laneman, J. N. (2015). A Second-Order Achievable Rate Region for Gaussian Multi-Access Channels via a Central Limit Theorem for Functions. *IEEE Trans. Inf. Theory* 61(12), 6719–6733.

Munoz, O., Pascual-Iserte, A., and Vidal, J. (2015). Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading. *IEEE Trans. Veh. Technol.* 64(10), 4738–4755.

Patel, M., Naughton, B., Chan, C.-L., Sprecher, N., Abeta, S., Neal, A., and et al. (2014). Mobile-edge computing: Introductory technical. *ETSI White Paper*, 1–36.

Pham, Q.-V., Fang, F., Ha, V. N., Piran, M. J., Le, M., Le, L. B., Hwang, W.-J., and Ding, Z. (2020). A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art. *IEEE Access* 8, 116974–117017.

Piao, J., Niu, K., Dai, J., and Dong, C. (2020). Approaching the Normal Approximation of the Finite Blocklength Capacity Within 0.025 dB by Short Polar Codes. *IEEE Wireless Commun. Lett* 9(7), 1089–1092.

Polyanskiy, Y., Poor, H. V., and Verdú, S. (2010). Channel Coding Rate in the Finite Blocklength Regime. *IEEE Trans. Inf. Theory* 56(5), 2307–2359.

Raghavan, R. and Tompson, C. D. (1987). Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica* 7(4), 365–374.

Rahimi, M. R., Ren, J., Liu, C.-H., Vasilakos, A. V., and Venkatasubramanian, N. (2014). Mobile cloud computing: A survey, state of art and future directions. *Mob. Net. & Applics* 19(2), 133–143.

Razaviyayn, M., Hong, M., and Luo, Z.-Q. (2013). A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization. *SIAM J. Optim.* 23(2), 1126–1153.

Salmani, M. and Davidson, T. N. (Oct. 2017). Energy-Optimal Computational Offloading for Simplified Multiple Access Schemes. In: *Conf. Rec. Asilomar Conf. Signals, Syst., Comput.* Pacific Grove, CA, 1847–1851.

Salmani, M. and Davidson, T. N. (2018). *Multiple Access Computational Offloading: Communication Resource Allocation in the Two-User Case (Extended Version)*. Available ta `https://arxiv.org/abs/1805.04981`.

Salmani, M. and Davidson, T. N. (2020a). Energy-Optimal Multiple Access Computation Offloading: Signalling Structure and Efficient Communication Resource Allocation. *IEEE Trans. Signal Process.* 68, 1646–1661.

Salmani, M. and Davidson, T. N. (2020b). Uplink Resource Allocation for Multiple Access Computational Offloading. *Signal Process.* 168(107322), 1–12.

Salmani, M. and Davidson, T. N. (July 2016). Multiple Access Computational Offloading. In: *Proc. IEEE Wrkshp Signal Process. Adv. Wireless Commun.* Edinburgh, Scotland.

Salmani, M., Sohrabi, F., Davidson, T. N., and Yu, W. (2019). Multiple Access Binary Computation Offloading via Reinforcement Learning. In: *Proc. 16th Canadian Wrkshp Inf. Theory.*

Sardellitti, S., Scutari, G., and Barbarossa, S. (2015). Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Trans. Signal Inf. Process. Netw.* 1(2), 89–103.

Scutari, G., Facchinei, F., and Lampariell, L. (2017). Parallel and Distributed Methods for Constrained Nonconvex Optimization — Part I: Theory. *IEEE Trans. Signal Process* 65(8), 1929–1944.

Scutari, G. and Sun, Y. (2018). Parallel and Distributed Successive Convex Approximation Methods for Big-Data Optimization. In: *Multi-agent Optimization*. Ed. by F. Facchinei and J.-S. Pang. C.I.M.E. Lectures in Mathematics. Springer, 141–308.

Starr, T., Cioffi, J. M., and Silverman, P. J. (1999). *Understanding Digital Subscriber Line Technology*. Prentice Hall.

Sun, Y., Babu, P., and Palomar, D. P. (2017). Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning. *IEEE Trans. Signal Process.* 65(3), 794–816.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surveys & Tuts.* 19(3), 1657–1681.

Tan, V. Y. F. and Tomamichel, M. (2015). The Third-Order Term in the Normal Approximation for the AWGN Channel. *IEEE Trans. Inf. Theory* 61(5), 2430–2438.

Wang, F., Xu, J., and Ding, Z. (2019). Multi-Antenna NOMA for Computation Offloading in Multiuser Mobile Edge Computing Systems. *IEEE Trans. Commun.* 67(3), 2450–2463.

Wang, F., Xu, J., Wang, X., and Cui, S. (2018). Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems. *IEEE Trans. Wireless Commun.* 17(3), 1784–1797.

Yang, Y., Hu, Y., and Gursoy, C. (2022). Energy Efficiency Analysis in RIS-aided MEC Networks with Finite Blocklength Codes. In: *Proc. IEEE Wireless Commun. Netw. Conf.* Austin, TX, 423–428.

You, C., Huang, K., Chae, H., and Kim, B.-H. (2017). Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Trans. Wireless Commun.* 16(3), 1397–1411.