RESOURCE MANAGEMENT FOR MOBILE COMPUTATION OFFLOADING

RESOURCE MANAGEMENT FOR MOBILE COMPUTATION OFFLOADING

BY

HONG CHEN, M.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ECE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2023) McMaster University

(ECE) Hamilton, Ontario, Canada

TITLE:	Resource Management for Mobile Computation Offloading
AUTHOR:	Hong Chen M.Sc. (Information & Communication Engineering), Chongqing University of Posts and Telecommunications, Chongqing, China
SUPERVISOR:	Prof. Dongmei Zhao Prof. George Karakostas
NUMBER OF PAGES:	xiii, 152

Lay Abstract

Mobile devices (MDs) such as smartphones are currently used to run a wide variety of application tasks. An alternative to local task execution is to arrange for some MD tasks to be run on a remote non-mobile edge server (ES). This is referred to as mobile computation offloading (MCO). The work in this thesis studies two important facets of the MCO problem.

1. The first considers the joint effects of communication and computational resource assignment on task completion times. This work optimizes task offloading decisions, subject to task completion time requirements and the cost that one is willing to incur when designing the network. Procedures are proposed whose objective is to minimize average mobile device power consumption, subject to these cost constraints.

2. The second considers the use of digital twins (DTs) as a way of implementing mobile computation offloading. A DT implements features that describe its physical system (PS) using models that are hosted at the ES. A model selection problem is studied, where multiple DTs share the execution services at a common ES. The objective is to optimize the feature accuracy obtained by DTs subject to the communication and computation resource availability. The thesis proposes different approximation and decomposition methods that solve these problems efficiently.

Abstract

Mobile computation offloading (MCO) is a way of improving mobile device (MD) performance by offloading certain task executions to a more resourceful edge server (ES), rather than running them locally on the MD. This thesis first considers the problem of assigning the wireless communication bandwidth and the ES capacity needed for this remote task execution, so that task completion time constraints are satisfied. The objective is to minimize the average power consumption of the MDs, subject to a cost budget constraint on communication and computation resources. The thesis includes contributions for both soft and hard task completion deadline constraints. The soft deadline case aims to create assignments so that the probability of task completion time deadline violation does not exceed a given violation threshold. In the hard deadline case, it creates resource assignments where task completion time deadlines are always satisfied. The problems are first formulated as mixed integer nonlinear programs. Approximate solutions are then obtained by decomposing the problems into a collection of convex subproblems that can be efficiently solved. Results are presented that demonstrate the quality of the proposed solutions, which can achieve near optimum performance over a wide range of system parameters.

The thesis then introduces algorithms for static task class partitioning in MCO. The objective is to partition a given set of task classes into two sets that are either executed locally or those classes that are permitted to contend for remote ES execution. The goal

is to find the task class partition that gives the minimum mean MD power consumption subject to task completion deadlines. The thesis generates these partitions for both soft and hard task completion deadlines. Two variations of the problem are considered. The first assumes that the wireless and computational capacities are given and the second generates both capacity assignments subject to an additional resource cost budget constraint. Two class ordering methods are introduced, one based on a task latency criterion, and another that first sorts and groups classes based on a mean power consumption criterion and then orders the task classes within each group based on a task completion time criterion. A variety of simulation results are presented that demonstrate the excellent performance of the proposed solutions.

The thesis then considers the use of digital twins (DTs) to offload physical system (PS) activity. Each DT periodically communicates with its PS, and uses these updates to implement *features* that reflect the real behaviour of the device. A given feature can be implemented using different *models* that create the feature with differing levels of system accuracy. The objective is to maximize the minimum feature accuracy for the requested features by making appropriate model selections subject to wireless channel and ES resource availability. The model selection problem is first formulated as an NP-complete integer program. It is then decomposed into multiple subproblems, each consisting of a modified Knapsack problem. A polynomial-time approximation algorithm is proposed using dynamic programming to solve it efficiently, by violating its constraints by at most a given factor. A generalization of the model selection problem is then given and the thesis proposes an approximation algorithm using dependent rounding to solve it efficiently with guaranteed constraint violations. A variety of simulation results are presented that demonstrate the excellent performance of the proposed solutions.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Prof. Dongmei Zhao, Prof. George Karakostas, and Prof. Terence D. Todd for their continuous support, motivation, and invaluable guidance throughout my Ph.D. study and research. Their unwavering belief in my potential and their encouragement during moments of doubt have inspired me to push beyond my limits and achieve more than I thought possible. I am thankful for the countless hours they invested in reviewing my work and providing valuable feedback that helped refine my research and successfully complete this thesis.

I extend my warm thanks to Prof. Emil Sekerinski for his insightful suggestions. I would also like to thank the members of the examining committee for reading my thesis and providing valuable feedback.

I would like to thank my kind and wise father, who always supports my passions without hesitation and always provides valuable and insightful advice when I am indecisive. I am extremely grateful to my warm mother for her endless love, unconditional support, and thoughtful care. I would also like to thank my sweet fiance, Jie Chen, for his unwavering companionship and unconditional belief in me.

Many thanks to my fellow colleagues, Peyvand Teymoori, Chunhui Guo, Kiana Noroozi, Mehrad Vaezi, Amirhosein Aghaei, and Subaha Mahmuda, in the Wireless Networking Laboratory for their help and valuable discussions we had.

Abbreviations

5G	Fifth Generation
BS	Base Station
BSTCP	Basic Static Task Class Partitioning
CPU	Central Processing Unit
CLE	Concurrent Local Execution
DT	Digital Twin
DP	Dynamic Programming
ES	Edge Server
ETSI ISG	European Telecommunications Standards Institute Industry Specification Group
FPTAS	Fully Polynomial Time Approximation Scheme
HD	Hard Deadlines
HP	Hierarchical partitioning
IP	Integer Programming
JSTCP	Joint Static Task Class Partitioning and Network Resource Allocation
LOGA	Level of Geometric Accuracy
MCO	Mobile Computation Offloading
MCC	Mobile Cloud Computing

MEC	Mobile Edge Computing
MD	Mobile Device
MINLP	Mixed Integer Nonlinear Programming Problem
NO	Network Owner
NL	Network Leaseholder
PS	Physical System
PDC	Partitioning based on Delay Constraints
QoS	Quality of Service
QoE	Quality of Experience
RAN	Radio Access Network
SD	Soft Deadline
STCP	Static Task Class Partitioning
TCP	Task Class Partitioning
VEC	Vehicular Edge Computing

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Mobile devices (MDs) such as smartphones are increasingly used as a platform for running a wide variety of applications including those involving gaming, virtual reality and natural language processing (Abolfazli *et al.* (2014); Wang *et al.* (2015); Wang *et al.* (2018)). Based on the Ericsson Mobility Report 2020 (Ericsson (2020)), smartphone usage is projected to reach approximately 8.9 billion by 2025, representing a significant growth from the current subscription level of 8.1 billion. The diversity of supported applications tends to stress MD capabilities in terms of data processing, storage, and battery lifetime (Kumar and Lu (2010)).

Mobile computation offloading (MCO) can be used to improve MD performance by running computational tasks on a remote cloud server rather than executing them locally (Noor *et al.* (2018); Kwon *et al.* (2016); Ba *et al.* (2013)). Since the energy needed for task execution is incurred by the cloud server, a reduction in MD energy consumption can often be obtained. During MCO, wireless communications is used by the MD to communicate

with the cloud server. This interaction incurs MD energy use that would not otherwise exist if the task were executed at the MD. MCO also incurs added latency due to the time needed for the MD to interact with the cloud server (Alameddine *et al.* (2019); Liu *et al.* (2021)). An *edge server* (ES) located close to the network base stations (BSs) is typically used to reduce this additional delay and energy use by providing high interconnection bandwidth between the BS and the ES (Huawei Inc. (2016)). These networks are referred to as *wireless edge networks*. It has been estimated that tens of billions of network edge devices will be deployed in the future (Mao *et al.* (2017)).

A key issue in MCO deals with the question of whether a given task should be executed locally or offloaded. This offloading decision is not a trivial one for many reasons (Chen *et al.* (2021)). For example, the increased use of MCO may create competition for the limited ES computational capacity that is used for the remote task execution (Muñoz *et al.* (2015); Nath and Wu (2020)). Offloading decisions are also more problematic when the MD interacts with the ES over wireless transmission channels that may change randomly during the computation offload. For these reasons, it is important to incorporate the temporal evolution of the system into the offloading decision making process (Yue *et al.* (2022); Hekmati *et al.* (2020)). This includes the latencies incurred due to the queueing delays experienced by offloaded tasks as they wait for ES execution.

Prior work has also considered the question of how to configure system resources so that MCO is best accommodated (Muñoz *et al.* (2015); Du *et al.* (2018); Yang *et al.* (2019); Zhang *et al.* (2018); Chen *et al.* (2020b); Nath *et al.* (2020)). These are the issues that are considered in the thesis and involve the tradeoffs between wireless communication and edge server capacity assignment and how these affect the delay performance experienced by the MDs. The wireless and execution capacity assignment problem in MCO can be

informally stated as follows. A network leaseholder (NL) purchases both wireless channel capacity and edge server execution services, subject to a cost budget constraint. The leased resources are then used to provide MCO to a large set of MDs (Yi *et al.* (2021)). In this case, the NL simply purchases services from the network owner (NO), who prices the cost of unit wireless channel and computational resources.

The acceptable delay for task execution is often tightly constrained for many applications, such as those involving gaming, virtual reality, and object recognition (Chen *et al.* (2020a)). Task execution deadline constraints significantly increase the difficulty of the problem compared to prior work without completion time requirements or that uses mean delay alone as the performance criterion (Deng *et al.* (2020); Zaw *et al.* (2021)). The variability in acceptable delay for different classes of tasks further complicates MCO decisions. This is especially true when offloaded tasks contend for wireless channel and computational resources in an uncontrolled fashion, and may impair the advantages of MCO, since it is easy to see that minimizing the used energy and satisfying the task delay constraints are competing objectives, i.e., in an effort to satisfy tight latency constraints, costly energy-wise MCO decisions may be necessary. Thus, the problem of task scheduling so that delay constraints are satisfied and energy consumption is minimized, becomes a difficult problem for MCO. The thesis studies the use of task class partitioning (TCP) to address this problem.

As an application of MCO, physical systems (PSs) need to offload the task of running their digital twins (DTs) to a more powerful server than running the DTs on themselves. For this, state-related data should be periodically uploaded to the server that hosts the DTs in order to keep real-time synchronization between the two. DT has attracted a lot of attention from both industry and academia since it was proposed over a decade ago. A DT

is a virtual representation of a PS that can represent both the static and dynamic behavior of the PS (Vaezi *et al.* (2022)). In order to maintain real-time synchronization between the PS and its DT, the PS periodically sends state-related data to a server that hosts the DT. At the server, the data is processed so that PS features can be provided by the DT to other objects on its behalf (Lu *et al.* (2021)). In order to maintain tight synchronization between the two, the DTs of the PSs are often placed at the ES that is located within the same wireless network as the PSs in wireless edge networks. By taking advantage of the resources available in the digital space, DTs have proven to be beneficial in various application areas, such as predictive maintenance and optimization in industrial manufacturing (Talkhestani and Weyrich (2020)) and intelligent network resource management (Dai and Zhang (2022); Lu *et al.* (2021)).

A DT implementation is considered *fully functional* if it can provide sufficient information of the PS required by an application in terms of age of information (AoI) (Vaezi *et al.* (2023)) and/or level of accuracy (Barricelli *et al.* (2019); Vaezi *et al.* (2022); Tang *et al.* (2022); Shen *et al.* (2021)). A real PS usually has a large number of features. The level of accuracy of each feature that should be supported by its DT depends on the requirements of the applications that interact with it (Stegmaier *et al.* (2022); Yiping *et al.* (2021); Lu *et al.* (2020)). Different DT feature accuracies are defined by different DT *models*, which describe how PS inputs are processed so that the feature can be generated. As one would expect, features with higher levels of accuracy will typically use models that require more input data from the PS and higher levels of computation at the DT (Paldino *et al.* (2022); Wang *et al.* (2020)). In the wireless edge networks, communication between the PS and its DT involves wireless transmission. Hence, the quality of a DT feature is affected by both the wireless transmission quality and the computational capacity of the ES that hosts the

DT. For an edge network that supports DTs for a large number of PSs, an issue is how to provide the best DT feature accuracy under limited wireless channel and edge computing resources. This is one of the problems that this thesis considers.

1.2 Contributions and Thesis Organization

This work considers the problem of resource management for mobile computation offloading and digital twins in wireless edge networks. The contributions in the thesis are summarized below.

This thesis first considers the problem of assigning the wireless communication bandwidth and the ES capacity used for the task execution, so that task completion deadline constraints are satisfied. The thesis includes contributions for both soft and hard task completion deadline constraints. The soft deadline case aims to create assignments so that the probability of task completion time deadline violation does not exceed a given violation threshold. In the hard deadline case, it creates resource assignments where task completion time deadlines are always satisfied. This is done by incorporating concurrent local execution (CLE) into the problem formulations. The objective is to minimize the average power consumption of the MDs, subject to a cost budget constraint for obtaining the communication and computation resources. The problems are first formulated as mixed integer nonlinear programs. Approximate solutions are then obtained by decomposing the problems into a collection of convex subproblems that can be efficiently solved. Results are presented that demonstrate the quality of the proposed solutions, which can achieve near optimum performance over a wide range of system parameters.

The thesis then introduces algorithms for static task class partitioning (STCP) in MCO

in order to further improve the performance of computation offloading under task completion time constraints. The objective is to partition a given set of task classes into two sets that are either executed locally by the MD or those classes that are permitted to contend for remote ES execution. The goal is to find the task class partition that gives the minimum mean MD power consumption. The thesis generates these partitions for both soft and hard task completion deadlines. Both basic static task class partitioning (BSTCP) and joint static task class partitioning and network resource allocation (JSTCP) problems are introduced. In the former, STCP is applied for a network with preallocated resources, while in the latter STCP is jointly studied with network resource allocation subject to an additional resource cost budget constraint. The proposed partitioning algorithms are based on heuristic class ordering methods. The thesis introduces two class ordering methods, a simpler one based on a task latency criterion, and an hierarchical version that first sorts and groups classes based on a mean power consumption criterion and then orders the task classes within each group based on a task completion time criterion. A variety of simulation results are presented that demonstrate the excellent performance of the proposed solutions for both given and optimized network resource assignments.

Finally, the DT model selection problem is studied in the third part of the thesis, where the DTs of multiple PSs are hosted at an ES in a wireless edge network. Each DT periodically communicates with its PS, and uses these updates to implement *features* that reflect the real behaviour of the PS. A given feature can be implemented using different *models* that create the feature with differing levels of system accuracy. The objective is to maximize the minimum feature accuracy for the requested features by making appropriate model selections subject to wireless channel and ES resource availability. The model

selection problem is first formulated as an NP-complete integer program. It is then decomposed into multiple subproblems, each consisting of a modified Knapsack problem. A polynomial-time approximation algorithm is proposed using dynamic programming to solve it efficiently, by violating its constraint by at most a given factor. A generalization of the model selection problem is then given and the thesis proposes an approximation algorithm using dependent rounding to solve it efficiently with guaranteed small constraint violations. A variety of simulation results are presented that demonstrate the excellent performance of the proposed solutions.

The rest of the thesis is organized as follows. Chapter 2 gives a brief background on MCO and DTs in wireless edge networks. In Chapter 3, the wireless and service allocation for MCO with task deadlines is investigated. Then, Chapter 4 introduces algorithms for task class partitioning in MCO. Both BSTCP and JSTCP problems are introduced and studied in the soft and hard class deadline cases. In Chapter 5, the digital twin model selection for feature accuracy under constrained resources in wireless edge networks is studied. The thesis is concluded in Chapter 6 with suggestions for possible future work.

The work in this thesis has resulted in the following publications.

- **H. Chen**, T. D. Todd, D. Zhao and G. Karakostas, "Wireless and service allocation for mobile computation offloading with task deadlines," Early access in *IEEE Transactions on Mobile Computing*, 2023. (doi: 10.1109/TMC.2023.3301577)

- **H. Chen**, T. D. Todd, D. Zhao and G. Karakostas, "Joint wireless and service allocation for mobile computation offloading with job completion time and cost constraints," In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, TX, USA, 2022, pp. 1278-1283. (doi: 10.1109/WCNC51071.2022.9771792)

- **H. Chen**, T. D. Todd, D. Zhao and G. Karakostas, "Task class partitioning for mobile computation offloading," Early access in *IEEE Internet of Things Journal*, 2023. (doi: 10.1109/JIOT.2023.3294887)

- **H. Chen**, T. D. Todd, D. Zhao and G. Karakostas, "Digital twin model selection for feature accuracy," Submitted to *IEEE Internet of Things Journal*, July 2023.

- **H. Chen**, T. D. Todd, D. Zhao, and G. Karakostas, "Digital twin model selection for feature accuracy in wireless edge networks," Accepted for publication in *2023 IEEE 34rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Toronto, ON, CA, 2023, pp. 1–6.

Chapter 2

Background

2.1 Introduction

This chapter provides a brief background on mobile computation offloading (MCO) and digital twin (DT) concepts in wireless edge networks. First, we start with a brief introduction of mobile edge computing (MEC), its characteristics and advantages. Then, we overview mobile computation offloading, discussing its challenges. Following this, we introduce digital twins, their applications and advantages. We then discuss DTs for wireless edge networks and some challenges that arise from their use in this application. Finally, related resource management issues in wireless edge networks are discussed.

2.2 Mobile Edge Computing

With the rapid development of wireless communication technology and intelligent end user devices, the use of mobile devices (MDs) as a platform for running a wide range of applications has increased dramatically. This includes those involving gaming, virtual reality,

face recognition, and natural language processing. Many of these applications impose strict demands on computation resources, leading to high MD energy consumption. However, due to their physical size limitations, MDs are inherently resource-constrained and possess limited computation capacities and short battery lifetimes. The conflict between computationally-intensive applications and the lightweight MDs with limited resources poses a challenge for the development of next-generation networks (Guo *et al.* (2018)). Mobile cloud computing (MCC) was once regarded as a promising way to relieve this conflict by providing convenient access to a shared computational resource pool in the remote internet cloud. For latency-sensitive applications however, the long transmission distances between mobile devices and the remote cloud often makes MCC infeasible (Jiang *et al.* (2021a)).

To address this issue, mobile edge computing (MEC) was proposed as a way to integrate cloud computing functionalities into radio access networks (RANs) by having edge servers (ESs) located close to the base stations (BSs), as shown in Figure 2.1. This was first proposed by a European Telecommunications Standards Institute Industry Specification Group (ETSI ISG) in December 2014. Compared to MCC, MEC can help reduce traffic congestion by offloading computation-intensive tasks to the ESs close to the MDs. In addition, MEC can reduce the end-to-end latency and MD energy consumption since the energy consumed for task execution is incurred by the edge server (Wang *et al.* (2017); Mach and Becvar (2017)). The main characteristics of mobile edge computing are summarized as follows.

- *Localization deployment*: In practical designs, it is often convenient to have the ESs located close to the wireless base stations (Taleb *et al.* (2017); Porambage *et al.* (2018)). The deployed servers can operate independently and provide computing

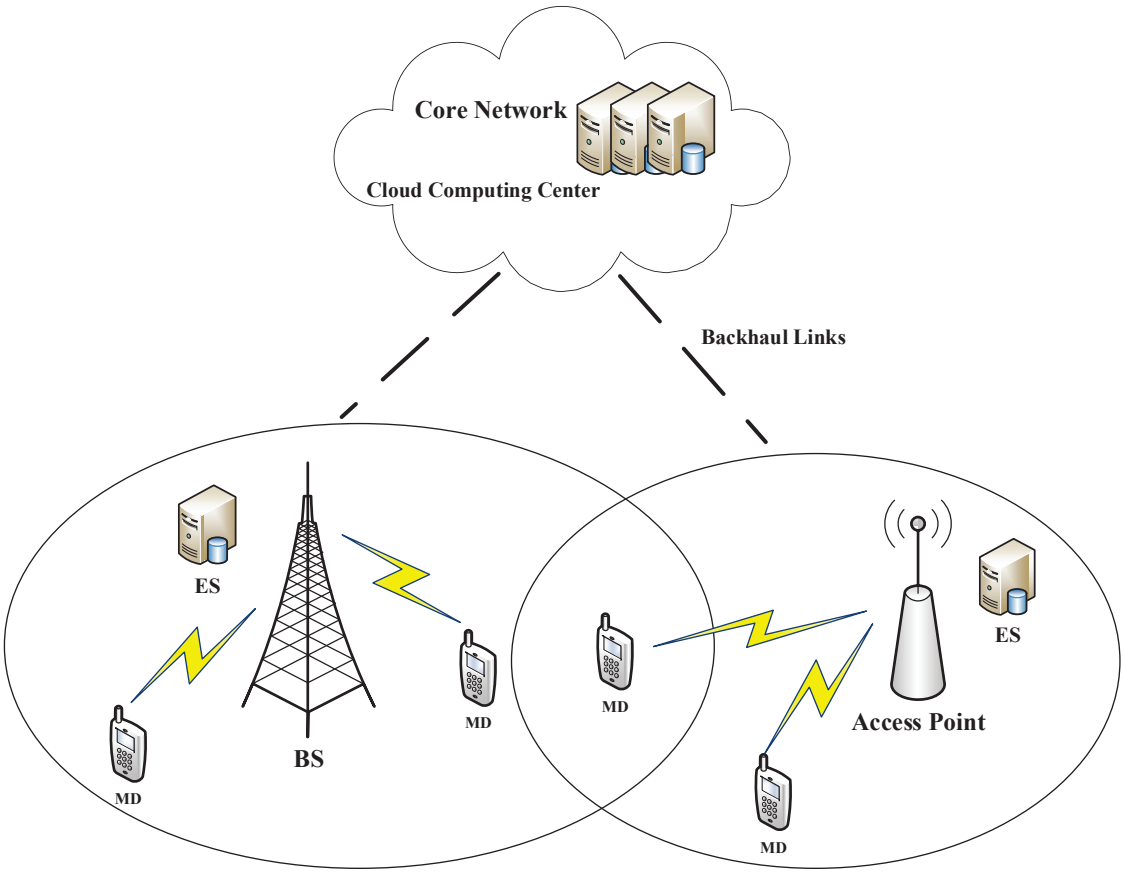


Figure 2.1: Architecture of MEC

services to the MDs. At the same time, the co-located ESs can have access to local data and resources.

- *Proximity*: Thanks to the deployed ESs in close proximity to the MDs, an ES can leverage user information based on the privacy protection mechanisms and analyze user behavior patterns in order to provide customized services to the users, thereby enhancing the quality of experience (QoE) (Tran *et al.* (2017)). Additionally, due to the reduction of the physical distance, the wireless transmission power of the MDs can be reduced, resulting in MD energy savings and extended device battery life.

- *Reduction of latency*: The execution of computational tasks is performed at the wireless edge networks in close proximity to the MDs, which avoids the transmission of task requests over backhaul links and the core internet. This enables MDs to experience low latency and high bandwidth services.

2.3 Mobile Computation Offloading

Mobile computation offloading (MCO) is an emerging technology to improve MD performance by allowing MDs to offload its computation-intensive tasks to a more resourceful server rather than executing them locally (Noor *et al.* (2018); Kwon *et al.* (2016); Ba *et al.* (2013)). This can help speed up task execution and help save the energy usage of the MDs. Since the energy needed for task execution is incurred by the computing server, a reduction in MD energy consumption can often be obtained (Gu *et al.* (2019); Zhang *et al.* (2017); Shi *et al.* (2018); Zhou *et al.* (2019)). Wireless communications is used by the MD to communicate with the computing server during MCO, which incurs wireless energy use that would not otherwise exist if the task were executed locally. MCO also incurs added latency due to the time needed for the MD to interact with the server (Alameddine *et al.* (2019); Liu *et al.* (2021)). Considering the emergence of numerous latency-constrained applications, offloading to servers located at the network edge is preferred, since it helps reduce this added latency.

In general, a complete MCO offloading cycle includes three time periods: 1) the uploading time of the input data needed for the execution of the computation; 2) the execution time of the task in the computing server; and, 3) the download time to return the execution results to the MD. In MCO, the task completion time is often dependent on the uploading time, since task execution times may be very short due to the powerful execution capability

of the server. The time needed to download the results to the MD may also be fairly short. Therefore, the data transmission rate and wireless channel conditions may have a significant impact on MCO performance (Masoudi and Cavdar (2021); Chen *et al.* (2020a)).

The question of whether a given task should be offloaded or not is crucial. Furthermore, there is the question of how much of the task should be offloaded (Mach and Becvar (2017)). In general, MCO decisions may result in the following three cases:

- *Local execution*: The complete execution of the task is performed locally at the MD.
- *Full offloading*: The complete task is offloaded by the MD and the execution of the task is performed at the server.
- *Partial offloading*: A portion of the task is executed locally while the rest is offloaded and executed at the server.

There are several key challenges in mobile computation offloading (Dab *et al.* (2019); Sheng *et al.* (2020); Du *et al.* (2018); Yang *et al.* (2019); Zhang *et al.* (2018); Chen *et al.* (2019b)), which are summarized as follows.

- An offloading decision for a given task should be made by considering not only how it will affect the MD, but also how it will affect other MDs that may also be using MCO.
- Since the offloading performance can be significantly affected by wireless channel conditions and capacity, an appropriate allocation of wireless channel bandwidth is needed when there is contention for this capacity. The random nature of wireless channel conditions should also be taken into account.

- Since the computing capability of an ES is typically lower than that of an internet cloud server, an efficient allocation of the ES computing resources may be needed when there is contention for these resources.

2.4 Digital Twins

A digital twin (DT) is a virtual representation of a real physical system (PS), which can represent and extend the system's behavior when interacting with other objects (Vaezi *et al.* (2022)). This enables applications to interact with the DT, rather than having them each communicate with the PS directly. By maintaining synchronization with the PS over time, a DT can also provide information and enable features that use historical PS data, which would typically be unavailable otherwise (Lu *et al.* (2021)). There are many practical use cases for DTs, including those in industrial manufacturing such as in system reconfiguration, predictive maintenance, optimization, and consistency checking. These illustrate the benefits of the DT concept throughout the entire product life cycle (Talkhestani and Weyrich (2020)). DTs have also been successfully applied in many other areas, such as aviation (Glaessgen and Stargel (2012); Barricelli *et al.* (2019)), healthcare and telemedicine (Wickramasinghe *et al.* (2021); Erol *et al.* (2020); Ricci *et al.* (2021)), smart homes (Chakrabarty *et al.* (2020); Cascone *et al.* (2021)), and smart cities (Lee *et al.* (2020); Raes *et al.* (2021); El Marai *et al.* (2020)).

Digital twins can provide *features* that are derived from, and reflect the behavior of its PS (Barricelli *et al.* (2019)). When one of these features is made available to an external application, there is an associated level of accuracy compared to what would be possible in the ideal case (Vaezi *et al.* (2022); Shen *et al.* (2021)). This deviation is referred to as *similarity* or *accuracy*, which is an important performance indicator (Barricelli *et al.*

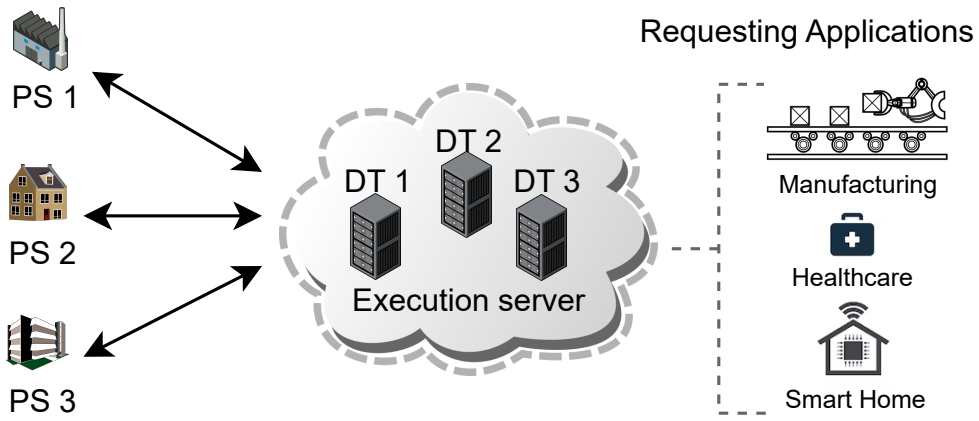


Figure 2.2: An illustration of digital twins

(2019); Vaezi *et al.* (2022); Tang *et al.* (2022); Shen *et al.* (2021)). When feature accuracy is the highest, a feature is indistinguishable from the best that would be ideally possible. In practical situations, however, it is often difficult or expensive to realize this level of similarity. Examples of this is where a "noisy" version of PS state information may be used as DT input, or when the feature generation includes approximations to reduce DT computation, so that real-time performance is improved (Vaezi *et al.* (2022)).

Different DT feature accuracies are defined by different DT *models*, which describe how PS inputs are processed so that the feature can be generated. As one would expect, features with higher levels of accuracy will typically use models that require more input data from the PS and higher levels of computation at the DT (Stegmaier *et al.* (2022); Yiping *et al.* (2021); Lu *et al.* (2020); Paldino *et al.* (2022); Wang *et al.* (2020)). For example, the work in (Stegmaier *et al.* (2022)) proposes a DT approach that enables Accuracy-as-a-Service for resistance-based electrical sensors. In order to reduce costs, DT models with differing feature accuracy are used in the control of a production system where the sensors are used.

The DTs of PSs are commonly hosted at the servers that the PSs are associated with, as shown in Figure 2.2. Each DT communicates with its associated PS, so that updates in the PS state can be incorporated into the features provided by the DT. This communication typically happens periodically and is referred to as *synchronization* (Han *et al.* (2022)). When a synchronization update occurs, the updating process involves uploading of the transferred data and processing at the server so that any features can be updated and made available to the requesting applications. As mentioned above, the amount of processing needed, and the amount of data transferred when synchronization occurs is a function of the level of accuracy associated with the model and the feature that is being provided.

2.4.1 Digital Twins in Wireless Edge Networks

Synchronization between a PS and its DT is essential for the DT to provide the required features to the requesting applications. Therefore, appropriate networking support is a critical component for the DT implementation. In order to keep tight synchronization between the two, a DT should often be placed at an edge server that is located within the same wireless network as the PS (Tang *et al.* (2022); Jiang *et al.* (2021b)). This helps to reduce the time needed to interact with each other, compared to placing a DT at a remote cloud server. A growing amount of work has considered the issues arising from DTs in wireless edge networks (Dai and Zhang (2022); Lu *et al.* (2021); Zhou *et al.* (2022)).

In wireless edge networks, the quality of a DT feature is affected by two main factors, i.e., the wireless transmission quality between the PS and the ES and the computation capacity of the ES that hosts the DT. The random nature of wireless channel conditions and the limited channel bandwidth resources affect the wireless transmission quality and the data transmission rate from the PS to the ES. Furthermore, since multiple DTs may

be co-located at the same ES, the synchronization update processing must share the ES computation capacity, and this can limit the achievable accuracy of the provided features. It is important to investigate how to provide the best DT feature accuracy under limited wireless channel and edge computing resources, for a wireless edge network that supports DTs for a large number of PSs.

2.5 Resource Management

With the rapid development of mobile communications, the number of mobile devices has dramatically increased, and the requirements for applications running on mobile devices are also becoming increasingly demanding. Hence, good management of the limited resources is important in order to provide the required MD quality of service (QoS) (Shen *et al.* (2022)). In particular, in wireless edge networks, the computation capacity of the ES located at the BS is typically small, compared to a powerful cloud server. The wireless channel bandwidth provided to the MDs is also constrained based on a cost budget. For an edge network that serves a large number of MDs, an issue is how to take the most advantage of the limited resources (Zhang *et al.* (2018); Nath *et al.* (2020); Li *et al.* (2022a)).

There are several key aspects of resource management that this thesis considers for MCO and DTs in wireless edge networks, which are listed below.

- *Wireless channel allocation*: The wireless data rate depends on the wireless channel conditions. However, the radio resources may be very limited and therefore an efficient and effective wireless channel allocation is needed in order to satisfy the stringent QoS requirements of the MDs.

- *Computing resource allocation*: In wireless edge networks, a potentially large number of MDs may need access to ES computing resources in order to meet application demands. As a result, there may be a significant competition for these limited computing resources. The question of how to allocate the ES computing resources is an important one.

Note that the mentioned aspects of resource management can be jointly considered in order to provide better performance (Yi *et al.* (2021); Chen *et al.* (2020a); Zaw *et al.* (2021)).

Chapter 3

Wireless and Service Allocation for MCO with Task Deadlines

3.1 Introduction

This chapter addresses the problem of assigning computational and wireless channel resources for mobile computation offloading (MCO), subject to task execution completion time deadlines. It includes formulations for both *soft and hard task completion deadlines*. In the *hard deadline* case, the completion time deadline given for every task must *always* be satisfied, i.e., each task must be uploaded and executed by the time that its associated deadline expires. This is a very stringent requirement, which is accomplished by including concurrent local execution (CLE) (Hekmati *et al.* (2020)) into the problem formulation. In CLE, local execution of the task may be initiated while offloading is ongoing, so that the task completion deadline is always met. In the *soft deadline* case, task completion times are permitted to violate their given deadlines, but the probability that this happens must be below a given violation threshold (Park *et al.* (2016); Li *et al.* (2022b)). Soft deadlines

use a statistical deadline constraint and depending on the chosen threshold, the deadline violation probability can be set as permissively as desired. In the remainder of the chapter, we define multiple task classes, with class-specific deadlines and constraint violation probabilities. Note that one can view the hard deadline case as a special case of soft deadlines, where the deadline violation factor is set to zero, meaning that no task deadlines can be violated. However, the hard deadline formulation requires CLE, which is not needed in the soft deadline case. As a result, the solution methods in the soft and hard deadline cases are different.

The inclusion of task deadline constraints significantly increases the difficulty of the problem compared to that of prior work with no completion time requirements or that uses a mean delay criterion (Deng *et al.* (2020); Zaw *et al.* (2021)). In order to obtain solutions to the problem, a queuing model is used to obtain the delay distribution experienced by tasks that are offloaded to the edge server (ES) (Yue *et al.* (2022); Zaw *et al.* (2021)). This model is incorporated into the resulting optimization problems, which are formulated as mixed integer nonlinear programming problems (MINLPs) that are computationally hard to solve exactly. Approximate solutions are obtained by decomposing the non-convex nonlinear formulation into a collection of convex subproblems that can be solved efficiently, and then picking the best of these solutions.

A variety of results are presented that characterize the tradeoffs between task deadline violation, average mobile device (MD) power consumption and the cost budget. The results show the quality of the proposed solutions, which can achieve close-to-optimum performance for a wide range of system parameters. The results also show that with CLE, the proposed solution not only guarantees to respect *all* hard task completion deadlines, but does so with only slightly higher MD power consumption when compared to the soft

task completion deadlines solution with a small deadline violation probability. On the other hand, this chapter shows that there is an apparent trade-off in the case of soft task completion deadlines between the average power consumption and the deadline violation probability. Namely, the average MD power consumption of our solution is significantly reduced when a higher deadline violation probability is tolerable.

The main contributions of this chapter are summarized below.

- This chapter addresses the problem of assigning computational and wireless channel resources for MCO, subject to task execution completion time deadlines. The work is the first that generates joint resource assignments for both soft and hard task deadlines using very general system modelling assumptions compared to prior work. The soft deadline case aims to create assignments so that the probability of task completion time deadline violation does not exceed a given violation threshold. In the hard deadline case, the work is also unique in that it creates resource assignments where task completion time deadlines are always satisfied. This is done by incorporating CLE into the problem formulations. For this reason, this is the first work that obtains system resource assignments for MCO that ensure that task completion time deadlines are always satisfied.

- Modeling both soft and hard job completion time targets significantly increases the difficulty of the problem compared to prior work with no completion time requirements or that uses a mean delay criterion (Deng *et al.* (2020); Zaw *et al.* (2021)). In both deadline cases, the chapter addresses this by incorporating an ES queueing system into the problem formulation that models the delay distribution experienced by arriving tasks. The assignment problem is addressed by numerically inverting the estimated probability generating function of task completion time and incorporating the

resulting probability density function (PDF) into the optimizations. These resource assignments are obtained under very general modeling assumptions, where the wireless channels are modeled as arbitrary base station (BS) specific sets of Markov processes and task execution times have a general probability distribution.

- The problems are first formulated as MINLPs, with *integral* decision variables for the number of wireless channels reserved, and a *continuous* decision variable for the portion of ES reserved. Even the relaxations of these MINLPs are difficult to solve, since they are non-convex. Hence, instead of following the common practice of solving the relaxation and rounding the fractional solution, we break the original non-convex MINLPs into collections of *convex* subproblems, that can be solved efficiently. This is achieved by the discretization of the *continuous* variable and the replacement of the *discrete* channel variables by approximate functions of the *continuous* blocking probabilities. Our solutions are approximate, and their accuracy depends on both the discretization granularity and the approximation functions used for blocking probabilities. On the other hand, they are based on very general assumptions, i.e., the existence of convex upper bound approximations of the inversion of blocking probabilities. The more restricted the system model is, the better these approximations are.

The rest of this chapter is organized as follows. In Section 3.2 the prior work most related to this chapter is reviewed. The system model and problem formulation is then described in Section 3.3. In Section 3.3.1, the general design problem is first considered assuming soft task completion time deadlines, where the probability of deadline violation is bounded. Following this, in Section 3.3.2 a formulation is described when task completion times are subject to hard deadlines. The problem formulations in both cases are non-convex

and difficult to deal with directly using conventional optimization approaches. In Section 3.4, approximation solutions are proposed where the original problems are decomposed into convex subproblems that can be efficiently solved. Both the soft and hard deadline cases are considered in Sections 3.4.1 and 3.4.2. Section 3.5 then introduces some common system assumptions used in the remainder of the chapter when solving the optimizations. Both the soft and hard deadline cases are then treated in detail in Sections 3.5.1 and 3.5.2. In Section 3.6 simulation results that demonstrate the proposed designs are given. Both the single class and multiple classes of tasks cases are considered in Sections 3.6.1 and 3.6.2. Finally, we present a summary of this chapter in Section 3.7.

3.2 Related Work

A large amount of prior MCO work considers the problem based on system state inputs sampled at task generation times, i.e., the models assume that the system is static throughout the offload period (Muñoz *et al.* (2015); Du *et al.* (2018); Yang *et al.* (2019); Zhang *et al.* (2018); Chen *et al.* (2020b); Nath *et al.* (2020); Chen *et al.* (2019b); Mu *et al.* (2020); Dab *et al.* (2019); Chen *et al.* (2020a); Geng *et al.* (2018); Chen *et al.* (2018); Masoudi and Cavdar (2021); Apostolopoulos *et al.* (2023)). Instead, the work in (Yue *et al.* (2022); Deng *et al.* (2020); Zaw *et al.* (2021); Li *et al.* (2022b)) considers that the wireless channels may change randomly during the offload.

When considering task offloading completion time, a latency minimization problem is investigated in (Ren *et al.* (2018)), average delay of task completion is considered in (Deng *et al.* (2020); Zaw *et al.* (2021)), a user satisfaction utility function is optimized in (Apostolopoulos *et al.* (2023)) based on the desired and actual task completion time and energy consumption. When tasks have hard delay constraints, the system may become

Table 3.1: Related Work Summary

References	Joint channel and computation resource assignment	Soft task deadlines	Hard task deadlines	Resource expense	Temporal evolution
Chen *et al.* (2020a); Chen *et al.* (2019b); Mu *et al.* (2020); Masoudi and Cavdar (2021)			✓		
Chen *et al.* (2020b); Ren *et al.* (2018)	✓				
Yi *et al.* (2021)			✓	✓	
Deng *et al.* (2020); Zaw *et al.* (2021)	✓				✓
Yue *et al.* (2022)			✓		✓
Li *et al.* (2022b)	✓	✓			✓
This chapter	✓	✓	✓	✓	✓

infeasible (Chen *et al.* (2020a, 2019b); Mu *et al.* (2020); Geng *et al.* (2018); Yue *et al.* (2022); Masoudi and Cavdar (2021)) and tasks can be dropped (Yue *et al.* (2022)) when the required hard task completion time cannot be satisfied.

Instead of considering a flow of tasks with random arrival times, as in this chapter, (Li *et al.* (2022b)) makes offloading decisions for tasks in the current time slot, where task offloading with statistical quality of service (QoS) guarantees (i.e., tasks are allowed to complete before a given deadline with a probability above a given threshold) is considered.

Besides task completion time, reducing energy consumption is another common objective in mobile computation offloading. Examples of this include minimizing the total

energy consumption of all MDs (Mu *et al.* (2020); Geng *et al.* (2018)), minimizing the energy consumption of the entire MCO system (Chen *et al.* (2019b)), or optimizing a utility function that is a weighted sum of task completion time and energy consumption (Chen *et al.* (2020b); Nath *et al.* (2020); Chen *et al.* (2018)).

Prior work has considered the optimization of communication and computation resources to improve MCO performance (Muñoz *et al.* (2015); Du *et al.* (2018); Yang *et al.* (2019); Zhang *et al.* (2018); Chen *et al.* (2020b); Nath *et al.* (2020)). The work in (Chen *et al.* (2019b); Mu *et al.* (2020); Chen *et al.* (2020a); Masoudi and Cavdar (2021); Yi *et al.* (2021)) focuses on the effect of radio resource allocations on offloading decisions. More specifically, task uploading decisions are jointly optimized with wireless channel assignments (Chen *et al.* (2020a, 2019b); Yi *et al.* (2021); Masoudi and Cavdar (2021)), channel transmission time scheduling (Mu *et al.* (2020); Yi *et al.* (2021)), and MD transmission power allocations (Chen *et al.* (2020a); Masoudi and Cavdar (2021); Yi *et al.* (2021)) for the task uploading. Comparing to binary offloading decisions, where an MD either offloads the entire task to the edge server or executes the task locally, partial offloading provides more flexibility in MCO (Chen *et al.* (2020b); Apostolopoulos *et al.* (2023); Peng *et al.* (2021); Feng *et al.* (2021); Cao *et al.* (2019); Mu *et al.* (2019)).

Table 3.1 summarizes the work described above that is most related to this chapter, and compares it to this chapter on five key properties:

Joint channel and computation resource assignment: The column denotes work where channel and computation resource assignments are jointly generated. Our work differs from the rest in that we assign aggregate channel resources from the network operator to each BS so that it can support its associated mobile device population, i.e., we do not allocate channel and computation resources of each BS and ES to

individual MDs.

Soft task deadlines: The work selected in this column considers some form of soft (i.e., statistical) task deadlines. However, the models we use in this chapter are quite different with more general underlying assumptions. Since our soft deadline model aims to set bounds on the probability of task deadline violation, we model the complete delay distribution experienced by executed tasks. This includes the BS channel delay (which is modeled by BS specific Markov processes) and the queueing delay experienced at the ES, where execution times can have a general distribution.

Hard task deadlines: Although there is other work selected in this column, a significant difference exists compared with this chapter. Our work *always* satisfies all hard task deadlines by incorporating the CLE mechanism into the modeled system. The related work, instead, considers the existence of hard deadlines as a problem constraint that may result in problem infeasibility, which can never happen in our case.

Resource expense: This column denotes work where the resources provided to the MDs are charged by a third-party (e.g., network operator). The work selected considers computational resource expense but not on the wireless BS side. A network profit maximization problem is studied where an expense budget is not considered, unlike the case in our work.

Temporal evolution: Temporal evolution means that the offload periods may include stochastic changes to the wireless channels and the ES, so that this information must be modeled in the problem formulation, as in our work. The randomness modeled in the selected work has different underlying assumptions compared to our work.

3.3 System Model and Problem Formulation

As shown in Figure 3.1, we consider a network that consists of N BSs that are owned and operated by a network owner (NO). The set of BSs is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$ and indexed by $n \in \mathcal{N}$. The network also contains an ES. Tasks generated by an MD can be offloaded through the wireless network and executed on the ES.

The NO permits a network leaseholder (NL) to rent wireless communication and ES computational capacity that the NL can use for mobile computation offloading for its MDs. When this is done, for each BS n, there are up to K_n available channels that can be selected by the NL. The cost of renting a channel from BS n is set by the NO to α_n. When a channel is included in the agreement, the NO agrees to provision its network so that sufficient resources are available to allow the traffic generated on the channel to be carried to the ES with an acceptable delay with a high degree of certainty. Since the ES is located at the edge of the network, we focus on the dominant sources of delay, i.e., wireless access at the BSs and task execution at the ES (Huawei Inc. (2016)).

In order to use the computing resources at the ES, the NL must also lease CPU resources at the ES. The cost (based on the number of CPU cycles per second) for leasing on the CPU resource is denoted by β. The maximum available CPU speed for rental is f^{C} CPU cycles per second.

When an agreement is made between the NO and NL, x_n is defined as the number of channels from BS n that are included, and $y \in [0, 1]$ is defined as the fraction of maximum CPU speed at the ES that is included, i.e., the CPU speed available for the NL will be yf^{C}. It is assumed that the NL has a cost budget, denoted by $B^{\max}$. Accordingly, the total rent

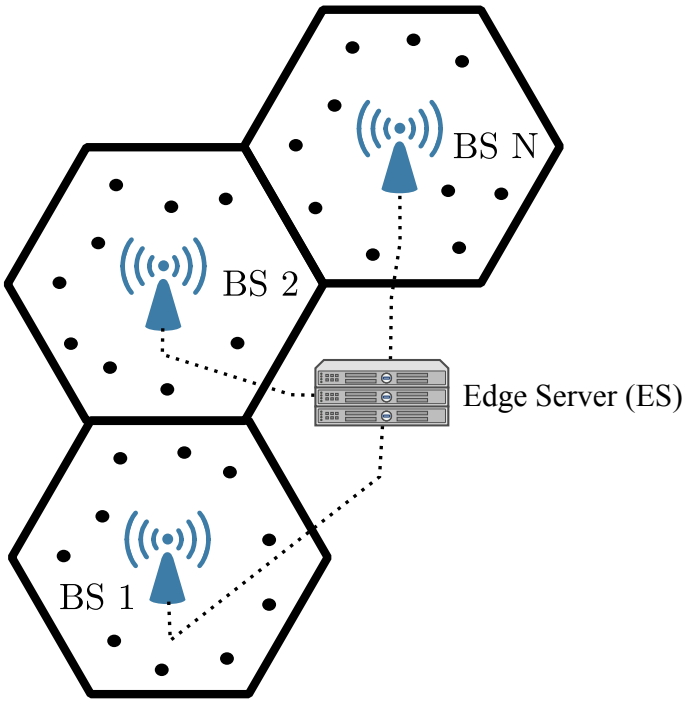


Figure 3.1: System Model

must satisfy the following constraint:

$$\sum_{n=1}^{N} \alpha_n x_n + \beta y f^{\mathrm{C}} \leq B^{\max}. \tag{3.3.1}$$

There are J classes of tasks generated by the MDs, which may need to be offloaded to the ES. Let $\mathcal{J} = \{1, 2, \ldots, J\}$ be the set of task classes. The class j of a task is defined by parameters s_j, q_j, and d_j, where s_j is the input data size in bits, q_j is the computation load in number of CPU cycles, and d_j is the deadline of the task in seconds. In what follows, $\tilde{d}_j = \lfloor d_j/\tau \rfloor$ is the task deadline rounded down to time slots of the same duration τ as the wireless transmission time slots (see below). The probability of a task generated by an MD belonging to class j is denoted by P_j^{C}; we assume that this probability is known, e.g., by observing the past history of offloading requests.

Our objective is to create a NO/NL contract for MCO. In MCO, tasks generated by an MD can be executed either locally (at the MD itself) or offloaded through the network and executed on the ES. We focus on two goals, each depending on how *hard* the task deadline constraint is. Our first goal is to accomplish this so that the mean mobile power consumption is minimized subject to the cost budget constraint and such that the probability that task execution deadline violation is bounded, i.e., the deadline constraints can be violated, i.e., *deadline constraints are soft.* Our second goal is to create a power-efficient, budget-respecting assignment that respects *all* task deadlines, i.e., *deadline constraints are hard*; for that purpose we will employ CLE (Hekmati *et al.* (2020)).

We model the wireless channels between the MDs and the BSs as discrete-time Markov processes. It is assumed that there are I_n channel models for BS n, which are a function of the radio propagation environment that the MDs experience at that BS. $\mathcal{I}_n = \{1, 2, \ldots, I_n\}$ is the set of all wireless channel models in BS n. For each of the channel models, the Markovian transition probabilities are defined in the usual way, i.e., given the channel state in the current time slot, there is a probability associated to its transition to another state in the next time slot. The time slot duration is defined to be τ seconds. A class j task, offloaded to BS n by the MD, encounters channel model k with probability $P^{\mathrm{G}}_{n,j,k}$; as with task generation probabilities P^{C}_j above, we assume that this probability is also known, e.g., by observing the past history of offloading requests.

To obtain the design, the decision to offload the execution of a task is made using a *local execute on blocking* (LEB) mechanism as follows. When an MD in BS n generates a class j task, the MD offloads the task if at least one of the x_n channels is available for immediate use. Otherwise, the MD executes the task locally. When a channel is available, the MD begins the offload by uploading the s_j task bits needed for execution on the ES.

The LEB mechanism is useful in that either local execution or remote offloading is initiated immediately at task release time, which may be advantageous when task deadlines are tight. It also provides a simple mechanism for assessing when the current level of local congestion is high, which would suggest that local execution is beneficial.

Tasks arrive at BS n according to a stationary process with average arrival rate λ_n tasks per second. According to the LEB mechanism, a new task is blocked from BS channel access if all the x_n channels are busy with uploading other tasks. We denote the task blocking probability at BS n by $P_{\mathrm{B}n}(x_n)$, which is a function of x_n. For the sake of notation simplicity, we use $P_{\mathrm{B}n}$ in the rest of the chapter. Let p^{L} be the power needed in the MD to process tasks. When a class j task is blocked from offloading and executed locally, the local execution time is given as $L_j = q_j/f$, where f is the MD's execution speed in number of CPU cycles per time slot[1]. Define $\bar{L}$ as the average local execution time of tasks. Since the task blocking is caused by channel access, which is the same for all task classes, we have $\bar{L} = \sum_{j=1}^{J} P_j^{\mathrm{C}} L_j$. The average energy consumption for executing a task locally is given by $p^{\mathrm{L}}\bar{L}$. Consider all the tasks that are generated in BS n and blocked from offloading in one second, then the mean energy for executing these tasks locally is

$$E_n^{\mathrm{L}}(x_n) = P_{\mathrm{B}n}\lambda_n p^{\mathrm{L}}\bar{L}, \tag{3.3.2}$$

which is the average power consumption of the MDs.

The wireless upload transmission time $t_{n,j,k}^{\mathrm{W}}$ of a jth class task in BS n when the wireless channel model is k, is measured in time slots. The mean wireless upload transmission time $\bar{t}_{n,j,k}^{\mathrm{W}}$ for jth class tasks in BS n according to channel model k can be calculated, since

[1] L_j is normally measured in CPU cycles, but in order to apply CLE and to simplify the system, we round it up to a multiple of τ.

$\Pr[t^{\mathrm{W}}_{n,j,k} = l]$ can be computed for all l from channel model k. Moreover, the mean wireless transmission time $\bar{t}^{\mathrm{W}}_n$ for BS n is

$$\bar{t}^{\mathrm{W}}_n = \sum_{j=1}^{J} \sum_{k=1}^{I_n} P^{\mathrm{C}}_j P^{\mathrm{G}}_{n,j,k} \bar{t}^{\mathrm{W}}_{n,j,k}. \tag{3.3.3}$$

Under the stated assumptions, the aggregate mean task arrival rate λ at the ES is given by

$$\lambda = \textstyle\sum_{n=1}^{N} (1 - P_{\mathrm{B}n}) \lambda_n. \tag{3.3.4}$$

As is normally the case for stability in a single server queueing system, the following constraint must always be satisfied,

$$\lambda < \mu^{\mathrm{C}}, \tag{3.3.5}$$

where μ^{C} denotes the mean service rate at the ES, i.e, $\mu^{\mathrm{C}} = yf^{\mathrm{C}} / \sum_{j=1}^{J} P^{\mathrm{C}}_j q_j$. As will become clear later, we can relax this constraint to $\lambda \leq yf^{\mathrm{C}} / \sum_{j=1}^{J} P^{\mathrm{C}}_j q_j$ without affecting our proposed solutions.

Let $t^{\mathrm{C}}_{n,j,k}$ be the delay (including both queueing and execution time) experienced by a jth class task from BS n at the ES, under wireless channel model k. It takes continuous values, and $\Pr[t^{\mathrm{C}}_{n,j,k} \leq t]$, for any $t \geq 0$, is a function of λ and μ^{C}. In what follows, $\tilde{t}^{\mathrm{C}}_{n,j,k}$ is the discretization of $t^{\mathrm{C}}_{n,j,k}$, measured in time slots; its distribution is calculated by

$$\Pr[\tilde{t}^{\mathrm{C}}_{n,j,k} = b] = \Pr[t^{\mathrm{C}}_{n,j,k} \leq b\tau] - \Pr[t^{\mathrm{C}}_{n,j,k} \leq (b-1)\tau] \tag{3.3.6}$$

for any number of time slots $b \geq 0$.

3.3.1 Problem Formulation with Soft Deadlines

We consider the distribution of total delay for an offloaded task, which is the sum of the *data upload* delay $t^{\mathrm{W}}_{n,j,k}$ and the *task execution at ES* delay $t^{\mathrm{C}}_{n,j,k}$, for BS n, task class j, and channel model k. Note that both delays are random variables. As mentioned earlier, the data transmission delay from the BS to the ES is negligible. In addition, in this chapter we consider the case of a very small amount of data returned once the execution is completed, and, therefore, we consider only uploading delays between MD and BS.

We now give a formal definition of soft task deadlines. Following common practice (e.g., Park *et al.* (2016)) in modelling soft deadlines along the lines of QoS requirements, a jth class task in BS n under wireless channel model k, must have a total delay satisfying

$$\Pr[t^{\mathrm{W}}_{n,j,k} + t^{\mathrm{C}}_{n,j,k} \leq d_j] \geq 1 - \varepsilon_j, \tag{3.3.7}$$

where $0 < \varepsilon_j \leq 1$ is the (given) tolerated probability that the completion time of a class j task exceeds its deadline.[2] Note that $t^{\mathrm{W}}_{n,j,k}$ takes discrete values (number of time slots), $t^{\mathrm{C}}_{n,j,k}$ takes discrete values (number of CPU cycle periods), while d_j is continuous (in seconds), so (3.3.7) assumes that all quantities are first converted to secs. Its LHS is a function of x_n, y.

The joint probability distribution of total delay is

$$\Pr[t^{\mathrm{W}}_{n,j,k} + t^{\mathrm{C}}_{n,j,k} \leq d_j] = \sum_{l=1}^{l_{\max}} \Pr[t^{\mathrm{W}}_{n,j,k} = l] \Pr[t^{\mathrm{C}}_{n,j,k} \leq d_j - l\tau], \tag{3.3.8}$$

where $l_{\max} = \lfloor (d_j - q_j/yf^{\mathrm{C}})/\tau \rfloor$ is the maximum value that l can take, since q_j/yf^{C} is the execution time at the ES without queueing.

[2]The case $\varepsilon_j = 0$ corresponds to the case of hard deadlines, and will be dealt with in the next section.

The average power consumption of MDs in BS n to upload tasks that are granted channels for offloading is

$$E_n^{\mathrm{T}}(x_n) = (1 - P_{\mathrm{B}n})\lambda_n p^{\mathrm{T}} \bar{t}_n^{\mathrm{W}}, \tag{3.3.9}$$

where p^{T} is the transmission energy per time slot used by the MD for uploading the task bits. Therefore, the expected average power consumption of the MDs for uploading and executing tasks arriving at BS n is $E_n^{\mathrm{L}}(x_n) + E_n^{\mathrm{T}}(x_n)$.

Our objective is to create an allocation that minimizes $E_n^{\mathrm{L}}(x_n) + E_n^{\mathrm{T}}(x_n)$ under the cost budget and deadline constraints (4.7.1) and (3.3.7). The problem can be formulated as follows:

$$\min_{\mathbf{x},y} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(x_n) + E_n^{\mathrm{T}}(x_n)] \text{ s.t.} \tag{3.3.10}$$

$$\sum_{n=1}^{N} \alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{3.3.11}$$

$$\Pr[t_{n,j,k}^{\mathrm{W}} + t_{n,j,k}^{\mathrm{C}} \leq d_j] \geq 1 - \varepsilon_j, \quad \forall n, j, k \tag{3.3.12}$$

$$(f^{\mathrm{C}} / \sum_{j=1}^{J} P_j^{\mathrm{C}} q_j) y \geq \lambda \tag{3.3.13}$$

$$x_n \in \{0, 1, \ldots, K_n\}, \qquad \forall n \in \mathcal{N} \tag{3.3.14}$$

$$0 \leq y \leq 1. \tag{3.3.15}$$

Constraints (3.3.11) and (3.3.12) are constraints (4.7.1) and (3.3.7). Constraint (3.3.13) is the (relaxed) queue stability requirement for ES; it is equivalent to (3.3.5), since equality leads to infinite mean queueing delay, which is never optimal. The optimization problem (3.3.10)-(3.3.15) is a MINLP problem. Constraint (3.3.14) ensures that the number of channels assigned does not exceed the maximum number available in each BS. Even the

fractional relaxation of MINLP problem (3.3.10)-(3.3.15) is non-convex, due to its objective and constraints (3.3.12), and, as a result, it is computationally inefficient to solve it exactly. Hence we are going to propose approximate solutions for it.

3.3.2 Problem Formulation with Hard Deadlines

For the case of *hard* deadline constraints, i.e., when the task deadline *must* be respected, we employ CLE (Hekmati *et al.* (2020)). In CLE, local execution of the task may be initiated while offloading is ongoing, so that the task deadline is always met, even if offloading fails to finish in time due to the stochastic nature of the wireless channels. Guaranteeing task completion before its deadline may incur additional costs (due to potentially simultaneous local and remote execution of the same task).

When CLE is employed, and in order to ensure that the local execution of a task from class j finishes by its deadline, the latest feasible starting time for local execution is

$$t_j^{\mathrm{L}} = \tilde{d}_j - L_j + 1. \tag{3.3.16}$$

The expected wireless transmission power is still given by (3.3.9). However, due to the overlap of offloading and local execution because of CLE, there is an extra mean power consumption due to a (potential) overlap with local execution. This expected overlap power consumption is

$$E^{O}_{n,j,k}(x_n, y) = (1 - P_{\mathrm{Bn}})\lambda_n \cdot \sum_{t=t_j^{\mathrm{L}}}^{\tilde{d}_j} \sum_{l=1}^{t-\lceil \frac{q_j}{yf^{\mathrm{C}}\tau} \rceil} \Pr[t^{\mathrm{W}}_{n,j,k} = l] \Pr[\tilde{t}^{\mathrm{C}}_{n,j,k} = t - l] \cdot p^{\mathrm{L}}(t - t_j^{\mathrm{L}} + 1), \tag{3.3.17}$$

where t is the number of time slots needed to complete the offloaded task, and $(t - t_j^{\mathrm{L}} + 1)$ is the offloading and local execution overlap. Note that $\Pr[\tilde{t}^{\mathrm{C}}_{n,j,k} = t - l]$ is a function of x_n and y.

In case the task offloading goes beyond the finish of the local execution of a task, there is an extra power consumption incurred, whose expected value is

$$E^{B}_{n,j,k}(x_n, y) = (1 - P_{\mathrm{Bn}})\lambda_n \cdot \sum_{t=\tilde{d}_j+1}^{+\infty} \sum_{l=1}^{t-\lceil \frac{q_j}{yf^{\mathrm{C}}\tau} \rceil} \Pr[t^{\mathrm{W}}_{n,j,k} = l] \Pr[\tilde{t}^{\mathrm{C}}_{n,j,k} = t - l] p^{\mathrm{L}} L_j. \quad (3.3.18)$$

Hence, the expected power consumption of MDs for offloaded tasks in BS n in one second is

$$E^{\mathrm{C}}_n(x_n, y) = E^{T}_n(x_n) + \sum_{j=1}^{J} \sum_{k=1}^{I_n} P^{\mathrm{C}}_j P^{\mathrm{G}}_{n,j,k} [E^{O}_{n,j,k}(x_n, y) + E^{B}_{n,j,k}(x_n, y)], \quad (3.3.19)$$

and the expected power consumption of MDs for tasks arriving at BS n in one second is $E^{\mathrm{L}}_n(x_n) + E^{\mathrm{C}}_n(x_n, y)$.

As before, our objective is to minimize the total expected power consumption of the MDs for uploading and executing the tasks that are generated in one second, but now

subject to hard deadline constraints. The problem is formulated as follows:

$$\min_{\mathbf{x},y} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(x_n) + E_n^{\mathrm{C}}(x_n, y)] \text{ s.t.} \tag{3.3.20}$$

$$\sum_{n=1}^{N} \alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{3.3.21}$$

$$(f^{\mathrm{C}} / \sum_{j=1}^{J} P_j^{\mathrm{C}} q_j) y \geq \lambda \tag{3.3.22}$$

$$x_n \in \{0, 1, \ldots, K_n\}, \quad \forall n \in \mathcal{N} \tag{3.3.23}$$

$$0 \leq y \leq 1. \tag{3.3.24}$$

3.4 General Approximate Allocation Solutions

In this section, we propose approximate solutions for optimization problems (3.3.10)-(3.3.15) and (3.3.20)-(3.3.24), by decomposing them into convex optimization subproblems which can be solved efficiently.

3.4.1 Approximate Solution for Soft Deadlines

In this subsection, we propose an approximate solution for the optimization problem (3.3.10)-(3.3.15) by decomposing it into several convex subproblems that can be solved efficiently, solve them, and then keep the best solution. More specifically, we discretize variable $y \in [0, 1]$ by breaking $[0, 1]$ into Y equal segments, so that y takes values $y_a = a/Y$, for $a = 0, 1, \ldots, Y$. With y fixed, we show that the relaxation of (3.3.10)-(3.3.15) can be approximated by a convex optimization problem, which can be solved in polynomial time. The resulting (fractional) x_n's are then rounded to integer values (and this is another source

of suboptimality for our solution method). After solving the resulting $Y+1$ problems, we output the minimum solution x^*, y^*. Obviously, the quality of the approximation depends on the discretization parameter Y.

We consider the relaxed version of problem (3.3.10)-(3.3.15), i.e., constraint (3.3.14) has been replaced by $x_n \geq 0, \forall n$. With y fixed, we show that the non-convex problem (3.3.10)-(3.3.15) can be transformed into an equivalent convex optimization problem with the $P_{\mathrm{B}n}$'s as the decision variables.

Lemma 1. *When y is fixed, constraints* (3.3.12)*,* (3.3.13) *can be replaced by constraint*

$$\sum_{n=1}^{N}(1-P_{\mathrm{B}n})\lambda_n \leq \lambda^*. \tag{3.4.1}$$

Proof. Note that $\Pr[t^{\mathrm{W}}_{n,j,k} + t^{\mathrm{C}}_{n,j,k} \leq d_j]$ is a monotonically decreasing function of the aggregate mean task arrival rate λ. Hence, by binary search in the range $[0, yf^{\mathrm{C}}/\sum_{j=1}^{J} P^{\mathrm{C}}_j q_j]$, we can approximate within any desired accuracy the maximum possible value of λ that satisfies constraints (3.3.12) for all n, j, k. Let λ^* be this maximum value (note that $\lambda^* < \mu^{\mathrm{C}}$, so stability is ensured). Using (3.3.4), the lemma follows. □

Next, we note that the blocking probability $P_{\mathrm{B}n}$ is monotonically decreasing in x_n. Let $P^{\min}_{\mathrm{B}n}$ be the blocking probability when $x_n = K_n$. Then we have the following

Lemma 2. *When y is fixed, constraints* (3.3.14) *can be replaced by the equivalent constraints*

$$P^{min}_{\mathrm{B}n} \leq P_{\mathrm{B}n} \leq 1, \ \forall n \in \mathcal{N}. \tag{3.4.2}$$

Constraint (3.3.11) is the only remaining constraint with an explicit dependence on the

x_n's. Since $P_{\mathrm{B}n}$ is a function of x_n, one could potentially use its inverse to replace x_n with a function of $P_{\mathrm{B}n}$. However, such an inversion function may not exist explicitly (and even if it does, it may be non-convex). In its stead, we can use a convex upper bound approximation F of the inversion of blocking probability, so that

$$x_n \leq F(P_{\mathrm{B}n}),\ \forall n \in \mathcal{N}. \tag{3.4.3}$$

Hence, the new convex optimization problem that approximates the original one when y is fixed, is the following:

$$\min_{\mathbf{P}_{\mathrm{B}}} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(P_{\mathrm{B}n}) + E_n^{\mathrm{T}}(P_{\mathrm{B}n})] \text{ s.t.} \tag{3.4.4}$$

$$\sum_{n=1}^{N} \alpha_n F(P_{\mathrm{B}n}) \leq B^{\max} - \beta f^{\mathrm{C}} y \tag{3.4.5}$$

$$\sum_{n=1}^{N} (1 - P_{\mathrm{B}n})\lambda_n \leq \lambda^* \tag{3.4.6}$$

$$P_{\mathrm{B}n}^{\min} \leq P_{\mathrm{B}n} \leq 1, \qquad \forall n \in \mathcal{N}. \tag{3.4.7}$$

After solving (3.4.4)-(3.4.7) and obtaining the $P_{\mathrm{B}n}$'s, we can compute the largest integral x_n^* which achieves a blocking probability equal to or bigger than $P_{\mathrm{B}n}$, for all $n \in \mathcal{N}$.

Algorithm GCASD (cf. Algorithm 1) codifies the solution method described above.

Theorem 3. *Algorithm GCASD runs in* $O(Y(\mathcal{L} + \log \frac{\mu^{\mathrm{C}}}{\epsilon} + N \log K_{\max}))$ *time, where* $O(\mathcal{L})$ *is the running time for solving convex program* (3.4.4)-(3.4.7).

Proof. Line 4 of the algorithm applies binary search in $[0, \mu^{\mathrm{C}}]$ in order to get a λ^* within ϵ of the optimal and takes time $O(\log \frac{\mu^{\mathrm{C}}}{\epsilon})$. Line 5 solves the convex problem (3.4.4)-(3.4.7) in time $O(\mathcal{L})$, e.g., by interior point methods (cf. Ch. 11 of Boyd and Vandenberghe (2014)).

Algorithm 1 General Case Approximation for Soft Deadlines (GCASD)

Input: $\lambda_n, p^{\mathrm{T}}, p^{\mathrm{L}}, \alpha_n, K_n, \beta, f^{\mathrm{L}}, f^{\mathrm{C}}, Y, s_j, d_j, q_j, \varepsilon_j, P_j^{\mathrm{C}}, P_{n,j,k}^{\mathrm{G}}$, PDFs of $t^{\mathrm{W}}, t^{\mathrm{C}}$
Output: resource assignments x^*, y^*

1: $cost^* = \infty$
2: **for all** $a = 0, \ldots, Y$ **do**
3: $y = a/Y$
4: Obtain λ^*, the upper bound of λ, by binary search in $[0, \mu^{\mathrm{C}}]$
5: $[P_{\mathrm{B}}, cost]$ = [solution, objective] of (3.4.4)-(3.4.7)
6: x_{int} = max integral x with blocking probabilities $\geq P_{\mathrm{B}}$
7: **if** $cost < cost^*$ **then**
8: $x^* = x_{int}; y^* = y; cost^* = cost$
9: **end if**
10: **end for**
11: **return** x^*, y^*

Line 6 takes time $O(N \log K_{\max})$, by applying binary search in the range $[0, K_{\max}]$ for each $x_n, n = 1, 2, \ldots, N$ (recall that $K_{\max}$ is the largest K_n). Hence every iteration of the for-loop of lines 2-10 runs in time $O(\mathcal{L} + \log \frac{\mu^{\mathrm{C}}}{\epsilon} + N \log K_{\max})$, and there are $O(Y)$ iterations (recall that Y is the granularity of y). The theorem follows. □

3.4.2 Approximate Solution for Hard Deadlines

In this subsection, we use a similar approach in order to solve (3.3.20)-(3.3.24). Here we decompose the original problem into several subproblems by discretizing both variable y as before, *and* λ. Then, for every possible (fixed) pair (y, λ), the non-convex problem (3.3.20)-(3.3.24) can be transformed into a convex optimization problem with $P_{\mathrm{B}n}$ as its decision variables, which can be solved in polynomial time. By calculating the pair (y^*, λ^*) whose subproblem achieves minimum average power consumption, integer values x_n^* for the original optimization problem are obtained from $P_{\mathrm{B}n}^*$.

In more detail, we discretize $y \in [0, 1]$ by breaking $[0, 1]$ into Y equal segments, and

then we discretize $\lambda \in [0, yf^{\mathrm{C}}/\sum_{j=1}^{J} P_j^{\mathrm{C}} q_j]$ by breaking interval $[0, yf^{\mathrm{C}}/\sum_{j=1}^{J} P_j^{\mathrm{C}} q_j]$ into Λ equal segments. At iteration (m, i) of this discretization, $y = y^{(m)}$ and $\lambda = \lambda^{(i)}$ are fixed. Then $\Pr[\tilde{t}_{n,j,k}^{\mathrm{C}} = t - l]$ can be calculated directly for any t and l, and the original optimization problem (3.3.20)-(3.3.24) becomes

$$\min_{\mathbf{x}} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(x_n) + E_n^{\mathrm{C}}(x_n)] \text{ s.t.} \tag{3.4.8}$$

$$\sum_{n=1}^{N} \alpha_n x_n \leq B^{\max} - \beta f^C y^{(m)} \tag{3.4.9}$$

$$\sum_{n=1}^{N} (1 - P_{\mathrm{B}n}) \lambda_n \leq \lambda^{(i)} \tag{3.4.10}$$

$$x_n \in \{0, 1, \ldots, K_n\}, \qquad \forall n \in \mathcal{N}. \tag{3.4.11}$$

This is still a non-convex non-linear integer program, which cannot be solved efficiently. As in Section 3.3, and by using (3.4.2)-(3.4.3), it becomes

$$\min_{\mathbf{P}_{\mathrm{B}}} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(P_{\mathrm{B}n}) + E_n^{\mathrm{C}}(P_{\mathrm{B}n})] \text{ s.t.} \tag{3.4.12}$$

$$\sum_{n=1}^{N} \alpha_n F(P_{\mathrm{B}n}) \leq B^{\max} - \beta f^C y^{(m)} \tag{3.4.13}$$

$$\sum_{n=1}^{N} (1 - P_{\mathrm{B}n}) \lambda_n \leq \lambda^{(i)} \tag{3.4.14}$$

$$P_{\mathrm{B}n}^{\min} \leq P_{\mathrm{B}n} \leq 1, \qquad \forall n \in \mathcal{N}. \tag{3.4.15}$$

Problem (3.4.12)-(3.4.15) is a convex program and can be solved efficiently. Hence, we can obtain the optimal blocking probabilities $P_{\mathrm{B}n}^*$, corresponding to a pair $(y^{(m)}, \lambda^{(i)})$. We can compute the largest integral x_n^* which achieves blocking probabilities no smaller than

Algorithm 2 General Case Approximation for Hard Deadlines (GCAHD)

Input: $\lambda_n, p^{\mathrm{T}}, p^{\mathrm{L}}, \alpha_n, K_n, \beta, f^{\mathrm{L}}, f^{\mathrm{C}}, Y, s_j, d_j, q_j, \Lambda, P_j^{\mathrm{C}}, P_{n,j,k}^{\mathrm{G}}$, PDFs of $t^{\mathrm{W}}, t^{\mathrm{C}}$
Output: resource assignments x^*, y^*

1: $cost^* = \infty, y = 0, \lambda = 0$
2: **while** $y \leq 1$ **do**
3: **while** $\lambda \leq y f^{\mathrm{C}} / \sum_{j=1}^{J} P_j^{\mathrm{C}} q_j$ **do**
4: $[P_{\mathrm{B}}, cost] =$ [solution, objective] of (3.4.12)-(3.4.15)
5: $x_{int} =$ max integral x with blocking probabilities $\geq P_{\mathrm{B}}$
6: **if** $cost < cost^*$ **then**
7: $x^* = x_{int}; y^* = y; cost^* = cost$
8: **end if**
9: $\lambda = \lambda + \frac{y f^{\mathrm{C}} / \sum_{j=1}^{J} P_j^{\mathrm{C}} q_j}{\Lambda}$
10: **end while**
11: $y = y + \frac{1}{Y}$
12: **end while**
13: **return** x^*, y^*

$P_{\mathrm{B}n}^*$, for all $n \in \mathcal{N}$, by using binary search based on the fact that the P_{Bn}'s are decreasing functions of the x_n's. After collecting the solutions for all iterations (m, i), we output the minimum cost one $\mathbf{x}^*, y^*$.

Algorithm GCAHD (cf. Algorithm 2) codifies the solution method described above.

Theorem 4. *Algorithm GCAHD runs in $O(Y\Lambda(\mathcal{L} + N \log K_{\max}))$ time, where $O(\mathcal{L})$ is the running time for solving convex program* (3.4.12)-(3.4.15).

Proof. Line 4 solves convex problem (3.4.12)-(3.4.15) in time $O(\mathcal{L})$, e.g., by interior point methods (cf. Ch. 11 of Boyd and Vandenberghe (2014)). Line 5 takes time $O(N \log K_{\max})$, by applying binary search in the range $[0, K_{\max}]$ for each $x_n, n = 1, 2, \ldots, N$ (recall that $K_{\max}$ is the largest K_n). Therefore, an iteration of the inner while-loop (lines 3-10) takes time $O(\mathcal{L} + N \log K_{\max})$, for a total of Λ iterations, while the outer while-loop (lines 2-12) runs for a total of Y iterations (recall that Y and Λ are the granularity of y and λ respectively). The theorem follows. □

3.5 Task Arrival and Offloading Assumptions

In the remainder of this chapter, we assume that tasks arrive from the MDs at BS n according to a Poisson process with mean arrival rate λ_n. The Poisson process assumption is commonly made in this type of situation, since the number of mobile devices in a given coverage area is typically quite large, each contributing to a small fraction of the total load (Khintchine (1932)). In this case, we can invoke the *insensitivity property* of the Erlang B formula, to compute the probability of blocking at each BS (Burman (1981)). Note that, typically, the Erlang B result is derived using the $M/M/N/N$ Markovian queue, which assumes exponentially distributed channel upload (i.e., service) times (Daley and Servi (1998)). Due to insensitivity, the result holds for any service time distribution with the same mean. Therefore, the blocking probability for a task arriving at BS n is

$$P_{\mathrm{B}n} = \left(\frac{\lambda_n}{\mu_n^{\mathrm{W}}}\right)^{x_n} \frac{1}{x_n!} \left[\sum_{r=0}^{x_n} \left(\frac{\lambda_n}{\mu_n^{\mathrm{W}}}\right)^{r} \frac{1}{r!}\right]^{-1} \tag{3.5.1}$$

where μ_n^{W} denotes the mean service rate, which can be calculated by $\mu_n^{\mathrm{W}} = 1/\bar{t}_n^{\mathrm{W}}$. Function (3.5.1) is convex in x_n (Messerli (1972)).

Note that due to the Poisson process task arrival assumption, the channel state sampled by arriving tasks is given by the steady-state equilibrium probability distribution of the Markovian channel at that MD. This follows from the PASTA rule (Wolff (1982)).

We assume that the aggregate task arrival process at ES is Poisson (Shanbhag and Tambouratzis (1973)), and, therefore, arriving tasks sample the asymptotic equilibrium state distribution of ES. This approximation is justified due to the mixing of arrivals at ES from BSs operating independently. In this case, ES can be modeled as an $M/G/1$ queue, whose waiting time is given by the random variable w^{C}. Given λ and knowledge of the data

upload distribution, the distribution of w^{C} can be obtained by numerical inversion of the probability generating function of system waiting time for $M/G/1$ (Khintchine (1932)). In this case, the execution time of a task at the ES depends only on which class it belongs to, i.e., $t^{\mathrm{C}}_{n,j,k} = t^{\mathrm{C}}_j$, for all n and k, and $t^{\mathrm{C}}_j = w^{\mathrm{C}} + q_j/yf^{\mathrm{C}}$. Thus, $\Pr[t^{\mathrm{W}}_{n,j,k} + t^{\mathrm{C}}_j \leq d_j]$ can be easily obtained.

When applying algorithms GCASD (Algorithm 1) and GCAHD (Algorithm 2) in this case, the upper bound F used in problem (3.4.4)-(3.4.7) and (3.4.12)-(3.4.15) becomes Berezner *et al.* (1998):

$$x_n \leq \frac{\lambda_n}{\mu^{\mathrm{W}}_n}(1 - P_{\mathrm{B}n}) + \frac{1}{P_{\mathrm{B}n}},\ \forall n. \tag{3.5.2}$$

3.5.1 Approximation with Soft Deadlines

In this case, problem (3.4.4)-(3.4.7) becomes:

$$\min_{\mathbf{P}_{\mathrm{B}}} \sum_{n=1}^{N} [E^{\mathrm{L}}_n(P_{\mathrm{B}n}) + E^{\mathrm{T}}_n(P_{\mathrm{B}n})] \text{ s.t.} \tag{3.5.3}$$

$$\sum_{n=1}^{N} \alpha_n (\frac{\lambda_n}{\mu^{\mathrm{W}}_n}(1 - P_{\mathrm{B}n}) + \frac{1}{P_{\mathrm{B}n}}) \leq B^{\max} - \beta f^{\mathrm{C}} y \tag{3.5.4}$$

$$\sum_{n=1}^{N} (1 - P_{\mathrm{B}n})\lambda_n \leq \lambda^* \tag{3.5.5}$$

$$P^{\min}_{\mathrm{B}n} \leq P_{\mathrm{B}n} \leq 1, \qquad \forall n \in \mathcal{N}. \tag{3.5.6}$$

Problem (3.5.3)-(3.5.6) is convex, and Algorithm 1 can be implemented efficiently according to Theorem 3.

3.5.2 Approximation with Hard Deadlines

In this case, problem (3.4.12)-(3.4.15) becomes

$$\min_{\mathbf{P}_{\mathrm{B}}} \sum_{n=1}^{N} [E_n^{\mathrm{L}}(P_{\mathrm{B}n}) + E_n^{\mathrm{C}}(P_{\mathrm{B}n})] \text{ s.t.} \tag{3.5.7}$$

$$\sum_{n=1}^{N} \alpha_n (\frac{\lambda_n}{\mu_n^{\mathrm{W}}}(1 - P_{\mathrm{B}n}) + \frac{1}{P_{\mathrm{B}n}}) \leq B^{\max} - \beta f^C y^{(m)} \tag{3.5.8}$$

$$\sum_{n=1}^{N} (1 - P_{\mathrm{B}n})\lambda_n \leq \lambda^{(i)} \tag{3.5.9}$$

$$P_{\mathrm{B}n}^{\min} \leq P_{\mathrm{B}n} \leq 1, \qquad \forall n \in \mathcal{N}. \tag{3.5.10}$$

Problem (3.5.7)-(3.5.10) is convex, and Algorithm 2 can be implemented efficiently according to Theorem 4.

3.6 Simulation Results

In this section, we present simulation results to demonstrate the performance of our proposed algorithms GCASD (Algorithm 1) and GCAHD (Algorithm 2). We adopt the two-state Gilbert-Elliot channel model (Gilbert (1960)), i.e., the channel states change by following a Markov chain with two states, "Good" (G) and "Bad" (B). This model is commonly used to characterize the effects of burst noise in wireless channels, where the channel can abruptly transition between good and bad conditions (Blazek and Mecklenbräuker (2018)). The Gilbert-Elliot channel is a difficult one for computation offloading algorithms to deal with compared to those where there is much more correlation in the channel quality as the offloading progresses. Let B_g and B_b, respectively, be the data transmission rate when the channel is in the G and B states. We consider that all channels have the same B_g

and B_b values but differ in their state transition probabilities that result in different propagation models. The transition probabilities for propagation model k in BS n are denoted as $P_{n,k}^{\mathrm{GG}}$, $P_{n,k}^{\mathrm{GB}}$, $P_{n,k}^{\mathrm{BG}}$, and $P_{n,k}^{\mathrm{BB}}$. In each time slot, the channel state Markov chain transitions in accordance with these probabilities. Denote $\pi_{n,k}^{\mathrm{G}}$ and $\pi_{n,k}^{\mathrm{B}}$, respectively, as the stationary probabilities of a channel in BS n for propagation model k being in the G and B states. Two sets of simulations are performed with set 1 for single class of tasks and set 2 for multiple classes of tasks. Default parameters used in the simulations are summarized in Table 3.2. The parameter settings that we use were taken from the references (Yi *et al.* (2021); Masoudi and Cavdar (2021); Yue *et al.* (2022)). These references summarize parameter settings for various types of applications including those that are inherently delay sensitive, such as gaming, face recognition and healthcare use. We intentionally use a wide range of parameter values based on the referenced ranges so that we can make conclusions that apply in general settings.

Table 3.2: Default Parameters

Parameter	Value in set 1	Value in set 2
τ	1 s	
p^{L}	250 mW	
p^{T}	2.5 mW	
λ_n	11, 13, 15 tasks/s	
K_n	15, 15, 20	
α_n	1, 1, 1 \$	
β	0.3×10^{-6} \$	0.25×10^{-6} \$
f^{C}	75M cycles/s	200M cycles/s
f	1M cycles/s	2M cycles/s
$B^{\max}$	140 \$	90 \$
Bg, B_b	2M, 0 bits per time slot	5M, 1M bits per time slot
s_j	2M bits	5M, 10M, 15M bits
d_j	4 s	6, 11, 16 s
q_j	3M CPU cycles	10M, 20M, 30M CPU cycles

3.6.1 Simulation set 1: single class of tasks

In this subsection, we will assume that all the tasks generated at the MDs have the same data size s and same computation load q, i.e., $s_j = s$ and $q_j = q$ for all j. When the channel is in the G state, the transmission rate of the wireless channel allows a task to be uploaded within one time slot; while when the channel is in the B state, the data transmission rate is zero. Since there is only one class of the tasks, subscript j can be dropped from the notation.

Let $t^{\mathrm{W}}_{n,k}$ be the time needed for uploading a task in BS n with channel model k. The probability that one task in BS n with channel model k can be uploaded in l time slots is given as follows

$$\Pr[t^{\mathrm{W}}_{n,k} = l] = \begin{cases} \pi^{\mathrm{G}}_{n,k}, & \text{when } l = 1 \\ \pi^{\mathrm{B}}_{n,k} {P^{\mathrm{BB}}_{n,k}}^{l-2} P^{\mathrm{BG}}_{n,k}, & \text{when } l \geq 2 \end{cases} \tag{3.6.1}$$

The mean wireless transmission time of a task in BS n uploaded through a channel with propagation model k can be calculated as follows

$$\bar{t}^{\mathrm{W}}_{n,k} = \sum_{l=1}^{\infty} l \Pr[t^{\mathrm{W}}_{n,k} = l] = 1 + \frac{P^{\mathrm{GB}}_{n,k}}{{P^{\mathrm{BG}}_{n,k}}^2 + P^{\mathrm{GB}}_{n,k} P^{\mathrm{BG}}_{n,k}}. \tag{3.6.2}$$

Based on this, the mean wireless transmission time of the tasks in BS n is $\bar{t}^{\mathrm{W}}_{n} = \sum_{k=1}^{I_n} P^{\mathrm{G}}_{n,k} \bar{t}^{\mathrm{W}}_{n,k}$, where $P^{\mathrm{G}}_{n,k}$ is the probability that a task in BS n is uploaded through a channel with propagation model k.

With a single class of tasks, the ES server becomes an $M/D/1$ queueing system,

$t^{\mathrm{C}}_{n,j,k} = t^{\mathrm{C}}$ for all n, j and k, and the distribution of delay is given by Franx (2001)

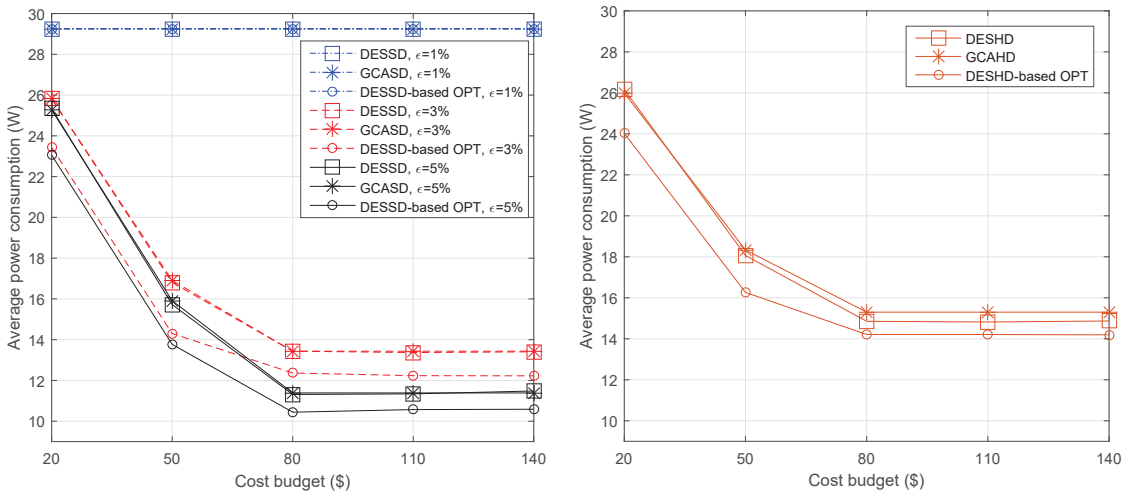
$$\Pr[t^{\mathrm{C}} \leq \hat{t}] = \left(1 - \frac{\lambda}{\mu^{\mathrm{C}}}\right) \sum_{z=0}^{\lfloor \hat{t}\mu^{\mathrm{C}} \rfloor} \frac{[\lambda(\frac{z}{\mu^{\mathrm{C}}} - \hat{t})]^z}{z!} e^{-\lambda(\frac{z}{\mu^{\mathrm{C}}} - \hat{t})} \tag{3.6.3}$$

where $\mu^{\mathrm{C}} = yf^{\mathrm{C}}/q$.

For comparison, we also run a discrete event simulation (DES) of the system using the x_n's and y solutions obtained from the proposed algorithms to validate our model assumptions, and these solutions are denoted as DESSD and DESHD, respectively, for the soft deadline (SD) and hard deadline (HD) cases. In addition, we simulate a DES-based OPT scheme for each proposed algorithm as follows. For GCASD, we first obtain all the possible combinations of x_n's under constraint (3.3.14); for a given combination of x_n's, we can obtain the solution of y based on (3.3.11) and (3.3.15), and then check if constraint (3.3.13) is satisfied based on the current set of x_n's and y. If not, we go to the next set of x_n's and repeat this procedure. If it is satisfied, we use this set of x_n's and y to run the DES for the system, and then check if (3.3.12) is satisfied. If not, we proceed to the next combination of x_n's and repeat the above procedure. If the constraints are satisfied, we save the obtained average power. After going through all the possible combinations of x_n's, we obtain the minimum average power and the corresponding x_n's and y. For GCAHD, we first obtain all the possible combinations of x_n's under constraint (3.3.23); for a given combination of x_n's, we can obtain the solution of y based on (3.3.21) and (3.3.24), and then check if constraint (3.3.22) is satisfied based on the current set of x_n's and y. If not, we go to the next set of x_n's and repeat this procedure. If it is satisfied, we use this set of x_n's and y to run the DES for the system. Then, we save the obtained mean power consumption. After going through all the possible combinations of x_n's, we obtain the minimum average power and

the corresponding x_n's and y.

In the simulation, we consider a cellular network consisting of 3 BSs. There are two propagation models at each BS with transition probabilities $P_{n,1}^{\mathrm{GG}} = 0.9$, $P_{n,2}^{\mathrm{GG}} = 0.7$, $P_{n,1}^{\mathrm{BB}} = 0.1$, and $P_{n,2}^{\mathrm{BB}} = 0.3$ for $n = 1, 2, 3$. The probabilities of the different channel models in BS 1 are $P_{1,1}^{\mathrm{G}} = 0.8$ and $P_{1,2}^{\mathrm{G}} = 0.2$; and those in BSs 2 and 3 are $P_{2,1}^{\mathrm{G}} = 0.5$, $P_{2,2}^{\mathrm{G}} = 0.5$, $P_{3,1}^{\mathrm{G}} = 0.2$, and $P_{3,2}^{\mathrm{G}} = 0.8$.



(a) Soft deadlines (b) Hard deadlines

Figure 3.2: Average power consumption versus cost budget (Single class of tasks)

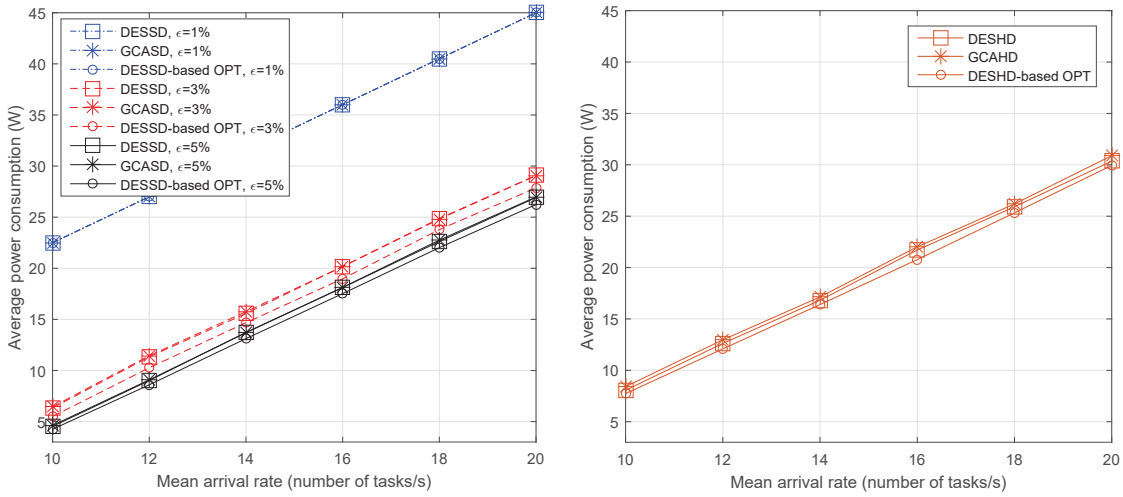
Figures 3.2a and 3.2b show the average power consumption of MDs versus $B^{\max}$ for the SD and HD cases, respectively. In Figure 3.2a, when the tolerable violation of latency ε is 1%, the average power consumption of MDs is a constant for all the solutions. This is because all the tasks are executed locally regardless of the cost budget, since the tight delay constraints cannot be satisfied if a task is offloaded. When ε is 3% or 5%, some tasks are allowed to be offloaded, and the average power consumption of the MDs decreases with $B^{\max}$ for all the solutions. This happens since, when the cost budget is small, the optimization is constrained by the cost budget, which limits the number of offloaded tasks;

and with the increase of $B^{\max}$, more channel and ES resource is available, leading to more MDs offloading their tasks. When $B^{\max}$ is large, the budget constraint is loose, and the task offloading completion is mainly affected by the changing wireless transmission conditions. Figure 3.2a also shows that the average MD power consumption decreases with ε for all the solutions, since larger ε makes it easier to meet the latency constraint through offloading, which results in more offloaded tasks and saves power in the MDs.

By comparing the average MD power consumption for $\varepsilon = 3\%$ and $\varepsilon = 5\%$ in Figure 3.2a, it is seen that the gap is small when the cost budget is small. The gap then increases as the cost budget increases, and finally becomes constant when the cost budget is sufficiently large. When the cost budget is low, the number of channels is small, which forces most tasks to be executed locally, regardless of the value of ε. As the cost budget increases, more channels are available, and the offloading decisions are determined by both ε and the available channel resources. When the cost budget is sufficiently high, the offloading decisions are mainly determined by the value of ε. The figure also shows that the average MD power consumption using GCASD is almost the same as using DESSD, which validates the model and approximations used in designing GCASD. The performance of GCASD is also close to DESSD-based OPT, which further shows good performance of the former.

By comparing Figures 3.2a and 3.2b, it can be seen that the average MD power consumption for the HD case is slightly higher than that for the SD case with $\varepsilon = 3\%$ and much lower than that for the SD case with $\varepsilon = 1\%$. For the SD case, when $\varepsilon = 1\%$, the tight (soft) delay constraint forces all the tasks to be executed locally, which results in the highest average power consumption of the MDs; and the power consumption decreases as ε increases and more tasks are allowed to be offloaded. Without having to use CLE, the SD solutions result in lower average MD power consumption than the corresponding HD

solutions. However, this is at a price that up to ε of the tasks do not meet their completion deadlines. On the other hand, using CLE in the GCAHD only incur slightly higher power consumption of the MDs compared to GCASD when $\varepsilon = 3\%$ For the HD case, the total average power consumption of the MDs decreases with $B^{\max}$ when $B^{\max}$ is small and becomes a constant when $B^{\max}$ becomes larger for all schemes, which is the same as that of the SD case with $\varepsilon = 3\%$ and 5%.

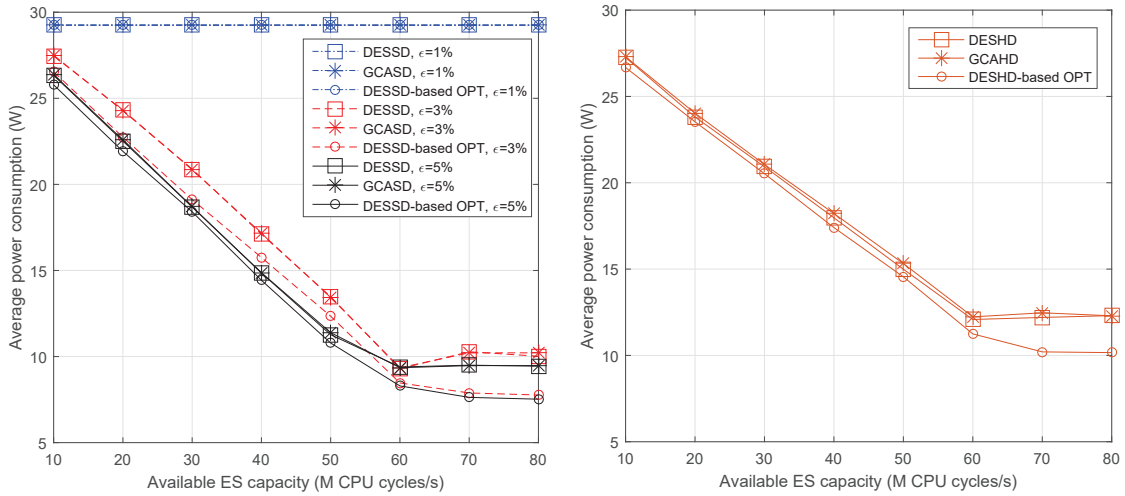


(a) Soft deadlines (b) Hard deadlines

Figure 3.3: Average power consumption versus mean arrival rate (Single class of tasks)

Figures 3.3a and 3.3b show the average power consumption versus λ_n (same for all BSs) for the SD and HD cases, respectively. The figures show that the power consumption increases linearly with λ_n for all schemes, since both the local execution power and the uploading transmission power are proportional to the mean task arrival rate. The average MD power consumption using GCAHD is close to that using GCASD with $\varepsilon = 3\%$ but much lower than that using GCASD with $\varepsilon = 1\%$. This demonstrates that the use of CLE in GCAHD is minimized, while always ensuring the HD of the tasks. Figure 3.3a shows that the performance of GCASD is very close to DESSD and DESSD-based OPT; and

Figure 3.3b shows that the performance of GCAHD is very close to DESHD and DESHD-based OPT. These observations are consistent with the ones from Figures 3.2a and 3.2b. This further demonstrates the good performance of GCASD and GCAHD and validates the model and approximations used in designing the proposed algorithms.



(a) Soft deadlines (b) Hard deadlines

Figure 3.4: Average power consumption versus available ES capacity (Single class of tasks)

Figures 3.4a and 3.4b show the average power consumption of the MDs versus f^{C}, which is the ES capacity, for the SD and HD cases, respectively. For the SD case with $\varepsilon = 1\%$, all tasks are executed locally; and when $\varepsilon = 3\%$ and 5%, offloading is possible for some tasks, and the number of tasks that can be offloaded increases with the ES capacity, resulting in lower power consumption of the MDs. As the ES capacity is sufficiently high, the average power consumption of MDs becomes a constant, since the offloading decisions are determined by the cost budget which limits the number of wireless channels for uploading tasks. Note that the slight increase in average power consumption when f^{C} is between 60 and 80 is caused by the discretization errors of variable y in algorithms 1

and 2. Increasing the Y values in the algorithms helps reduce the discretization errors but significantly increase the amount of time for running the simulations. Comparing the average power consumption of the HD and the SD cases shown in Figures 3.4a and 3.4b, we have consistent observations as in previous figures.

3.6.2 Simulation set 2: multiple classes of tasks

In this subsection, tasks have multiple classes. The two-state Gilbert-Elliot channels are considered. Let B_g and B_b, respectively, be the data transmission rates when a channel is in the G and B states.

Given the channel state transision probabilities, the distribution of wireless transmission time $t^{\mathrm{W}}_{n,j,k}$ for uploading a class j task in BS n through a channel with propagation model k can be calculated from (Hekmati *et al.* (2020)).

At the ES, the system of serving the uploaded tasks becomes an $M/G/1$ queueing system. Let B be a random variable representing the execution time of the tasks. We have $\Pr[B = \frac{q_j}{yf^{\mathrm{C}}}] = P^{\mathrm{C}}_j$, then the probability density function of B can be written as

$$f_B(\tilde{b}) = \sum_{j=1}^{J} \Pr\left[B = \frac{q_j}{yf^{\mathrm{C}}}\right] \delta\left(\tilde{b} - \frac{q_j}{yf^{\mathrm{C}}}\right) = \sum_{j=1}^{J} P^{\mathrm{C}}_j \delta\left(\tilde{b} - \frac{q_j}{yf^{\mathrm{C}}}\right), \tag{3.6.4}$$

and the Laplace-Stieltjes transform of $f_B(\tilde{b})$ is given by

$$g(s) = \sum_{j=1}^{J} P^{\mathrm{C}}_j e^{-\frac{q_j}{yf^{\mathrm{C}}}s}. \tag{3.6.5}$$

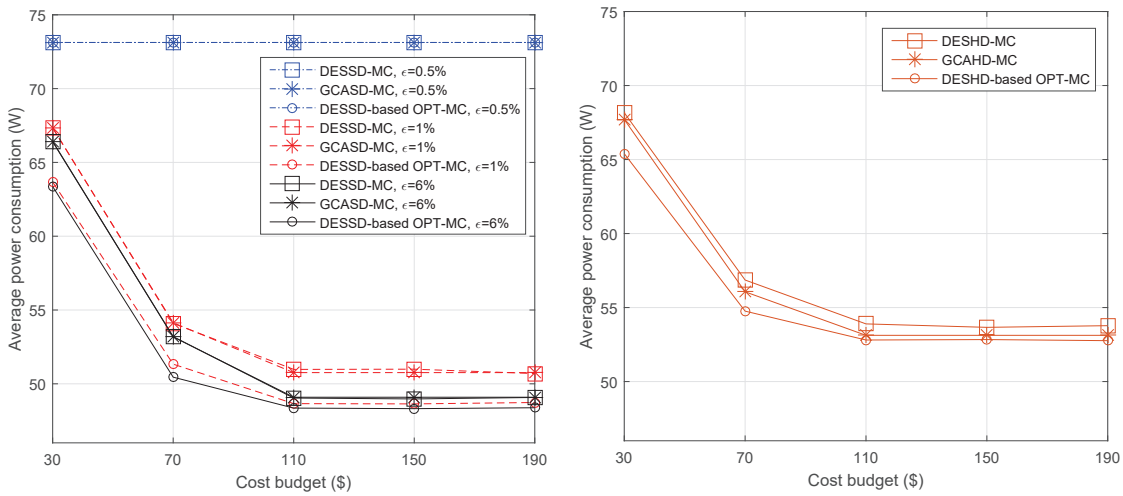
The Laplace-Stieltjes transform of the probability density function of queuing time w^{C} is

given by the Pollaczek-Khinchine transform (Khintchine (1932)) as

$$W^*(s) = \frac{(1-\lambda\bar{b})s}{s-\lambda(1-g(s))}, \tag{3.6.6}$$

where $\bar{b}$ is the mean of B. The distribution of w^{C} can be obtained by numerical inversion of (3.6.6).

In the simulation, we consider a cellular network consisting of 3 BSs, 3 task classes, and 2 channel propagation models. The channel state transition probabilities are $P_{n,1}^{\mathrm{GG}} = 0.9$, $P_{n,1}^{\mathrm{BB}} = 0.1$, $P_{n,2}^{\mathrm{GG}} = 0.6$, and $P_{n,1}^{\mathrm{BB}} = 0.4$ for $n = 1, 2, 3$. The probabilities of accessing channels with different propagation models in BS 1 are $P_{1,1}^{\mathrm{G}} = 0.8$ and $P_{1,2}^{\mathrm{G}} = 0.2$; those in BSs 2 and 3 are $P_{2,1}^{\mathrm{G}} = 0.5$, $P_{2,2}^{\mathrm{G}} = 0.5$, $P_{3,1}^{\mathrm{G}} = 0.2$, and $P_{3,2}^{\mathrm{G}} = 0.8$. The probabilities of a task belonging to different classes are $P_1^{\mathrm{C}} = 0.6$, $P_2^{\mathrm{C}} = 0.3$, and $P_3^{\mathrm{C}} = 0.1$.

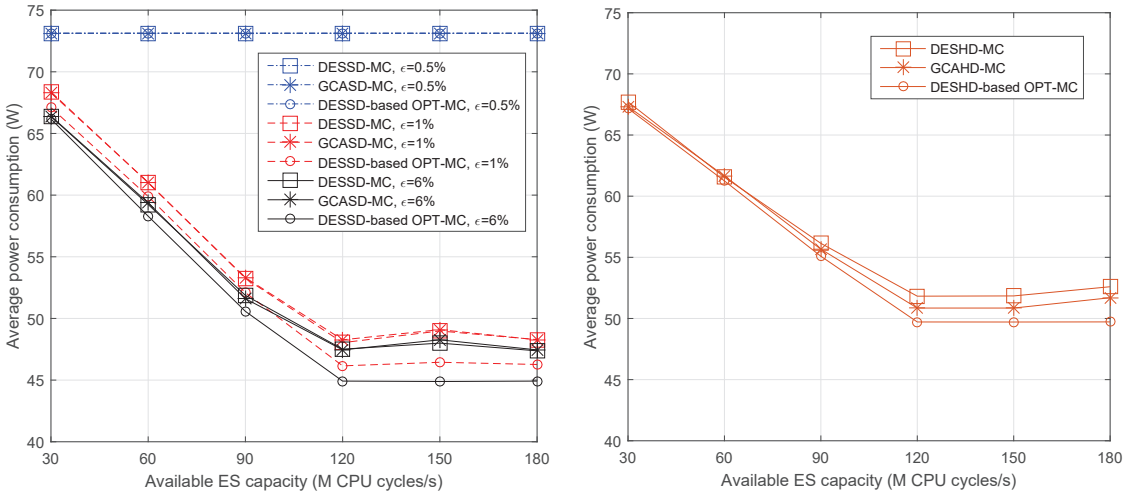


(a) Soft deadlines (b) Hard deadlines

Figure 3.5: Average power consumption versus cost budget (Multiple classes of tasks)

Figures 3.5a and 3.5b show the average power consumption of MDs versus $B^{\max}$ for

the SD and HD cases, respectively. In Figure 3.5a, when ε is 0.5%, all the tasks are executed locally regardless of the cost budget, since offloading cannot satisfy the tight delay constraints. When ε is 1% or 6%, the average power consumption of MDs decreases with $B^{\max}$ and then becomes a constant. By comparing the power consumption of the MD in the SD and HD cases, we can see that the average power consumption of MDs for the HD case is slightly higher than that for the SD case with $\varepsilon = 1\%$ and much lower than that for the SD case with $\varepsilon = 0.5\%$. Figures 3.6a and 3.6b show the total average power consumption of the MDs versus f^{C}. All the results show that our GCASD and GCAHD solutions achieve the average power consumption performance that is very close to DES-based OPT, and the observations in the multi-class simulations are consistent with the single-class simulations.



(a) Soft deadlines (b) Hard deadlines

Figure 3.6: Average power consumption versus available ES capacity (Multiple classes of tasks)

3.7 Summary

In this chapter, we have studied joint communication and computation resource management for mobile computation offloading. The objective is to minimize the average power consumption of the mobile devices subject to the completion time requirements of applications and a cost budget for the network resources. Our results show good performance of the proposed solutions. This work will be extended in the next chapter by taking into consideration the service requirements of the tasks in different task classes when making the offloading decisions in order to further improve the average power consumption of the mobile devices.

Chapter 4

Task Class Partitioning for Mobile Computation Offloading

4.1 Introduction

This chapter studies the use of *task class partitioning* (TCP) for mobile computation offloading (MCO) to address the problem of task scheduling so that delay constraints are satisfied and energy consumption is minimized. Class partitioning can be used in cases where mobile device (MD) tasks belong to one of a given set of task classes that can each be executed locally or offloaded. Each traffic class definition includes the data upload and execution requirements of the class and its execution deadline, i.e., the time by which task execution should be completed. Class partitioning is motivated by the following simple example: An offloaded traffic class with high computational needs and a loose execution delay constraint may unduly occupy the edge server (ES), thus preventing more time-constrained tasks from using MCO. In this case, it may be beneficial to pre-assign certain task classes to local execution so that others can more successfully exploit remote

offloading. In TCP, the task classes are therefore partitioned into two sets, those task classes to be executed locally and those that may be offloaded for remote execution. Depending on the adaptability of class definition, TCP can be either *static*, when the task classes are pre-defined and their definition does not change throughout the lifetime of the system, or *dynamic*, when the class definition can adapt to a changing environment. It is clear that the dynamic TCP is a sequence of static TCP instances, each corresponding to a period of stable environment parameters. In this chapter, we study the static version of TCP, i.e., STCP.

In STCP, the number of possible task class partitions is clearly exponential in the number of task classes. A way to overcome this is to prioritize the classes based on the advantages that they provide and their MCO compatibility with other classes. The chapter proposes two class partitioning approaches. In the first, task classes are prioritized according to their delay constraints. The second approach uses a hierarchical ordering that first prioritizes task classes according to power consumption, and then prioritizes task classes within groups of similar power demand according to task delays. We show that the hierarchical algorithm produces significantly better solutions than the first one. Both are easily implementable and computationally efficient. As a result, making the offloading decisions requires little communication and computation overhead and the delay introduced by waiting for the offloading decisions is low, which is advantageous when task deadlines are tight.

The chapter considers two different design problems. In the *Basic Static Task Class Partitioning (BSTCP)* problem, the wireless channel capacities at each base station (BS) and the ES computational capacity are given beforehand. The problem in this case is to create the task class partition that minimizes the average MD power consumption while satisfying the task execution deadline constraints. We show that the proposed static class

partitioning algorithms can significantly reduce the MD power consumption without violating task delay constraints. The second problem is where the channel and execution resources are not given, and the algorithms not only partition the task classes as before, but determine the channel and execution resource assignments, subject to a cost budget constraint, as well. This is referred to as the *Joint Static Task Class Partitioning and Network Resource Allocation* (JSTCP) problem.

Due to the static nature of the partitioning, the proposed algorithms are intended for use in situations where the set of task classes remain relatively stable over the time periods of interest. This is not a strong assumption, since although the mixture of tasks may continuously change, the *classes* in which these tasks belong may not do so. When task class partitioning is combined with resource allocation in JSTCP, the problem is inherently one of static resource allocation. The static assignment of wireless channel bandwidth and computational capacity is subject to a cost budget and resources are not re-allocated during network operation. In a design scenario using our algorithms for example, a network designer could assign resources based on their knowledge (or prediction) of the characteristics of the traffic flow that would be impinging on the resulting network. In a practical design, this would likely be done using worst-case traffic assumptions (e.g., anticipated "busy hour" statistics.). The algorithms in this chapter could easily be adapted to more quasi-static situations, where resource assignments are updated periodically based on evolving worst-case (busy-hour) traffic conditions. Overall, static TCP is a natural first step towards more sophisticated settings, e.g., dynamic TCP, which we leave for future work.

The main contributions of this chapter are summarized below.

- STCP is proposed as a way of improving the performance of computation offloading under task execution time constraints.

- Two main task class partitioning algorithms are proposed. The simpler algorithm uses a latency-based task class ordering. The more sophisticated hierarchical algorithm first sorts the task classes into sets based on their mean power consumption, and within each set, the task classes are further ordered using a task completion time criterion.

- The design algorithms consider both soft and hard task deadlines. In soft deadlines (SDs), the probability of task completion deadline violation is upper bounded by a class-specific value. Since the soft delay constraints are based on satisfying statistical bounds, the chapter models the detailed delay distribution of the tasks. The models permit arbitrary task upload and execution time distributions with any set of Markovian channel models. In the hard deadline (HD) case, task deadlines must *always* be respected. This is done by including Concurrent Local Execution (CLE) (Hekmati *et al.* (2020)) in the algorithms. The chapter provides the only known mechanism for class partitioning that always achieves this goal.

- Both BSTCP and JSTCP problems are introduced. In the former, STCP is applied for a network with preallocated resources, while in the latter STCP is jointly studied with network resource allocation. Both BSTCP and JSTCP are studied in the soft and hard class deadline constraint cases.

- A variety of simulation results are presented that demonstrate the good performance of the proposed algorithms. In the JSTCP case, our algorithms perform well compared to both a state-of-the-art no-partitioning algorithm and the optimal no-partitioning algorithm.

The remainder of the chapter is organized as follows. In Section 4.2, the prior work

most related to this chapter is reviewed. The system model is then described in Section 4.3. In Section 4.4, the general task offloading decision problems for both SD and HD are formulated. Following this, in Section 4.5, the partitioning algorithms are proposed. In Section 4.6, the partitioning algorithms are used to obtain solutions of the BSTCP problem. In Section 4.7, the JSTCP problem is formulated under a resource cost budget for both SD and HD cases. In Section 4.8 simulation results that demonstrate the proposed designs are given. Finally, we present a summary of this chapter in Section 4.9.

4.2 Related Work

This section reviews the most recent literature that relates to our work. In MCO, task completion time and MD energy consumption are two important performance metrics. In terms of task completion time, some work considers tasks with hard completion deadlines under static wireless channel conditions (Li *et al.* (2020); Ren and Xu (2021); Tan *et al.* (2022)), in which case the fixed channel gains make it possible to predict the wireless transmission time for an offloaded task. When the channel conditions vary with time, the uncertainty in future channel condition results in uncertainty in the completion time of offloaded tasks. In this case, an offloaded task may be discarded if it is not completed before the required deadline (Yue *et al.* (2022)). Given the difficulty to satisfy hard completion deadlines for offloaded tasks, some work considers mean delay performance of tasks in MCO, such as (Wu *et al.* (2020); Wu and Wolter (2018)), and other work makes offloading decisions to achieve a certain statistical delay bound (Qu *et al.* (2021); Li *et al.* (2022b)). In addition to the time-varying wireless channel conditions, the random task arrival process also causes uncertainty in the future network conditions, which affects the MCO performance and makes it difficult to support tasks with strict completion time requirements. In (Deng

et al. (2020); Zaw *et al.* (2021)), offloading decisions for tasks are made by considering the statistic information of task arrivals and the mean completion time of the tasks. Different from the existing work, we study MCO with both time-varying wireless transmission channels and random task arrivals, and consider both hard and soft task completion deadlines.

Offloading decisions in MCO are often jointly considered with network resource allocations to improve the offloading performance. For example, in (Zeng *et al.* (2022); Fang *et al.* (2022)), no constraints are specified for task completion time. Instead, the objective is to optimize a cost or utility function that is defined as a weighted sum of the MD energy consumption and average task completion time by jointly coordinating the co-channel interference and the task offloading decisions. In (Tan *et al.* (2022)), orthogonal channel allocations are jointly optimized with the task offloading decisions in order to minimize the MD energy consumption subject to hard task completion time under static channel conditions.

For tasks with strict completion time requirements, making the offloading decisions should take the minimal amount of time upon the task arrival. To make the offloading decisions, communication time may be required to collect input information needed for making the decisions and deliver the decisions, depending on the specific algorithms, and computation time is needed to run the decision making algorithms (Wu *et al.* (2020); Wu and Wolter (2018)). However, most existing work ignores the decision making time upon the task arrivals. In our work, the class partitioning is performed offline; and upon a task arrival, if the task class is in the set that is allowed to offload, the MD only needs to check if a channel is available in order to decide whether the task can be offloaded. The response time to make the offloading decision is low.

An offloading decision can be binary, which means that a given task is either entirely

executed at the MD or offloaded to an ES. Instead, in partial offloading, different portions of a task are allowed to be executed simultaneously at both the MD and the ES (Mu *et al.* (2020)) or at multiple ESs (Wu *et al.* (2018)). The additional flexibility of partial offloading helps reduce the task completion time, while saving energy for the MDs. In networks with highly dynamic topology, such as vehicular networks, partial offloading allows an MD to offload different portions of a task to different computing devices in order to take advantage of the short connection time to these computing devices (Wang *et al.* (2021, 2018)). Also, a task can be divided continuously (Mu *et al.* (2019, 2020); Wu *et al.* (2018)), or it can be modelled as mutually related subtasks and a binary offloading decision is made for each subtask (Wu *et al.* (2019); Wang *et al.* (2021, 2018)). Different from partial offloading and individual task partitioning, our work considers *task class* partitioning (TCP), which partitions task classes into two sets for either remote or local execution. TCP can be combined with partial task offloading. However, in order to focus on the benefit of TCP, we do not apply partial offloading for individual tasks, i.e., after we partition task classes into locally executed or offloaded, each task belonging to the latter is either entirely offloaded (if a channel is available for data uploading) or executed locally (when no channel is available). We leave the study of the combination of TCP with partial task offloading to future work.

4.3 System Model

This chapter considers the problem where wireless channels at a set of base stations and ES capacity is leased for computation offloading. Let N be the total number of BSs, $\mathcal{N} = \{1, 2, \ldots, N\}$ be the set of BSs and x_n be the number of wireless channels that are used in BS n. Since the ES is located at the edge of the network, we focus on the dominant sources of delay, i.e., wireless access at the BSs and task execution at the ES (Huawei Inc. (2016)).

Tasks generated by an MD can be offloaded through the wireless network and executed on the ES. Let f^{C} be ES capacity (in the number of CPU cycles per second) and a fraction y, $0 \leq y \leq 1$, of the capacity is used.

There are J classes of tasks generated by the MDs. Let $\mathcal{J} = \{1, 2, \ldots, J\}$ be the set of task classes. A class j task is defined by parameters s_j, q_j, and d_j, where s_j (in bits) is the input data size needed for processing the task, q_j (in number of CPU cycles) is the computation load, and d_j (in seconds) is the task deadline. A class j task may be executed locally, in which case $T_j^{\mathrm{L}} = q_j/f^{\mathrm{L}}$ and $p^{\mathrm{L}}T_j^{\mathrm{L}}$, respectively, are the amounts of time and energy needed to process the task, where f^{L} is the local processing speed in number of CPU cycles per second and p^{L} is the local processing power in Watts. We assume that $T_j^{\mathrm{L}} < d_j$ for all $j \in \mathcal{J}$ so that the delay requirement can always be satisfied if a task is executed locally.

A task may be processed remotely at the ES. Let $t_{n,j}^{\mathrm{OFF}}$ be the total amount of time for offloading a class j task generated by an MD in BS n, then $t_{n,j}^{\mathrm{OFF}} = t_{n,j}^{\mathrm{W}} + t_j^{\mathrm{C}}$, where $t_{n,j}^{\mathrm{W}}$ and t_j^{C}, respectively, are the amount of time for uploading the task to the ES and the amount of time the task experiences at the ES before the completion of its execution. Note that $t_{n,j}^{\mathrm{W}}$ is a function of s_j and the wireless data transmission rate. We consider fixed transmission power at each MD and it adapts its transmission rate according to the current channel conditions. In this case, the amount of time needed to upload the s_j bits in order to offload a class j task is determined by the channel conditions.

All the offloaded tasks from different BSs and in different classes are processed at the ES according to a first-come-first-served queueing discipline. For a class j task, the execution time at the ES is $T_j^{\mathrm{C}} = q_j/(yf^{\mathrm{C}})$. In addition to execution time, queueing at the ES may incur additional latency. Let λ^{ES} be the mean aggregate task arrival rate at the ES

of all traffic classes from all BSs. The queueing delay of the offloaded tasks at the ES w^{C} is a function of λ^{ES}, and its statistics are the same for all tasks belonging to all classes and generated at all BSs. Since $t_j^{\mathrm{C}} = w^{\mathrm{C}} + T_j^{\mathrm{C}}$, we have $t_{n,j}^{\mathrm{OFF}} = t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}}$. Both $t_{n,j}^{\mathrm{W}}$ and w^{C} are random due to the randomness of the link gain of the wireless channel and the task arrival process at the ES, respectively. Therefore, $t_{n,j}^{\mathrm{OFF}}$ is also random.

4.4 Problem Formulation

The objective of this work is to minimize the mean power consumption of the MDs used for processing their tasks, when the latter can be either executed locally by their MD or be offloaded to the ES for execution, and under task delay constraints.

Both soft and hard delay constraints are considered. In the soft delay constraints case, the delay requirements of tasks are achieved with a predefined probability, i.e., $1 - \varepsilon_j$ with $\varepsilon_j \in (0, 1)$. Hence, our goal is to minimize the mean mobile power consumption such that the probability that task execution deadline violation is bounded, i.e., the deadline constraints *can be violated*, albeit rarely.

In the hard delay constraints case, the latency requirements of all the tasks should be *always satisfied*. Due to the random wireless channel conditions and task arrival process, guaranteeing the hard delay constraints is difficult for offloaded tasks. In (Hekmati *et al.* (2020)) we proposed to use concurrent local execution (CLE) when it is uncertain that an offloaded task can be completed before its deadline. More specifically, for an offloaded task, if it is not completed before time $t_j^{\mathrm{CLE}} = d_j - T_j^{\mathrm{L}}$, the MD starts to execute the task locally and in parallel to the MCO. CLE aborts if the ES completes the task before d_j in order to save the energy consumption of the MD. Again, the goal is to make power efficient offloading decisions which respect all task deadlines.

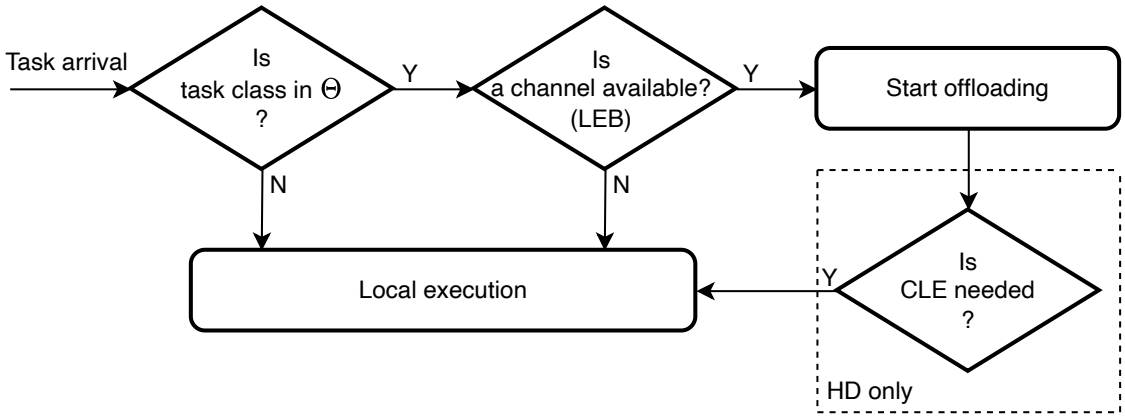


Figure 4.1: Offloading Decision Making Process

We extend the notion of a task class to the notion of a *BS-class* tuple as follows: $(n, j) \in \mathcal{N} \times \mathcal{J}$ is the set of class j tasks arriving at BS n. Note that this allows us to distinguish between the same task classes submitted to different BSs, so that the same task class may be treated differently at different BSs. In both soft and hard delay constraints cases, we adopt a two-step process in making the offloading decisions, as shown in Figure 4.1:

1. **Class set partitioning:** In this step, we partition the set $\mathcal{N} \times \mathcal{J}$ into subsets Θ and $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$. Tasks in BS-classes $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$ are always executed locally at the MDs. Offloading decisions for tasks from the BS-classes of Θ are made in the second step.

2. **Local execution on blocking (LEB):** This step applies only to tasks belonging in BS-classes of Θ. Upon the arrival of a class j task at BS n with $(n, j) \in \Theta$, if at least one of the x_n channels is available for immediate use, the MD reserves a free channel and starts uploading the task; otherwise the task is blocked from offloading and is executed locally at the MD. Note that LEB treats tasks from different task classes the same, and, therefore, within the same BS, the same blocking probability

applies to tasks of any class in $\mathcal{J}$.

Observations:

- The class set partitioning of the first step is performed offline, as preprocessing to MCO, and does not cause any additional delays to task processing. On the other hand, the LEB decision for a task is made immediately upon the task arrival. LEB requires minimal communication and computation overhead, and can be made with nearly zero additional delay. Overall, the extra delay introduced to the offloading decisions is negligible.

- Class set partitioning introduces a pre-offloading stage of 'task filtering', which will 'filter-out' tasks that would burden the offloading process due to their combination of parameters. For example, if tasks of class j have large data size s_j and very tight deadline d_j, then the tasks should not be offloaded, since they require a long channel uploading time and have a high probability of missing their deadlines.

- With this two-step process for making the offloading decisions, either local execution or remote offloading is initiated *immediately at task release time*, which may be advantageous when task deadlines are tight. It also provides a simple mechanism for assessing the ES workload at the MD. That is, when more wireless channels are used for uploading tasks, a higher computation load is at the ES, in which case executing tasks locally at the MDs is beneficial.

The problem formulation is presented first for soft completion deadlines, and then for hard completion deadlines.

4.4.1 Soft deadlines

With the soft delay constraints of the tasks, the power consumption of the MDs includes

- E^{DL}, which is the mean power consumption of the MDs for executing tasks belonging to classes in $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$,
- E^{L}, which is the mean power consumption of the MDs for executing tasks belonging to classes in Θ locally (these tasks are blocked for offloading due to no wireless channels upon their arrivals), and
- E^{T}, which is the mean power consumption of the MDs for uploading tasks belonging to classes in Θ.

The objective of class partitioning, together with the allocations of wireless and computation resources in case the latter are not given beforehand, is to minimize the mean power consumption of the MDs, i.e.,

$$\min E^{\mathrm{MD}} = E^{\mathrm{DL}} + E^{\mathrm{L}} + E^{\mathrm{T}} = \sum_{n=1}^{N} \left(E_n^{\mathrm{DL}} + E_n^{\mathrm{L}} + E_n^{\mathrm{T}} \right), \tag{4.4.1}$$

where E_n^{DL} is the portion of E^{DL} consumed by the MDs in the nth BS, E_n^{L} is the portion of E^{L} consumed by the MDs in the nth BS, and E_n^{T} is the portion of E^{T} consumed by the MDs in the nth BS. These "BS n portions" are given below.

Let $\lambda_{n,j}$ be the mean arrival rate of class j tasks in BS n. The local energy consumption of the MDs in BS n for processing class j tasks with $(n, j) \notin \Theta$ generated in one second is

$$E_n^{\mathrm{DL}} = \sum_{j:(n,j) \in (\mathcal{N} \times \mathcal{J}) \setminus \Theta} \lambda_{n,j} T_j^{\mathrm{L}} p^{\mathrm{L}}. \tag{4.4.2}$$

Since this is the mean energy consumption for tasks generated in one second, it is equivalent to the mean power consumption of the MDs for processing the tasks over a long term. Similarly, E_n^{L} and E_n^{T} given below are also the mean power consumption of the MDs.

For BS n, let P_n^{B} be the task blocking probability for all tasks with $(n,j) \in \Theta$ due to no channel being available upon their arrival. Recall that all tasks are treated the same at a BS for accessing a channel, and therefore, the blocking probability does not depend on task class j. How to find P_n^{B} is given in Section 4.6.2. The mean power consumption of all the MDs at BS n for processing these tasks is

$$E_n^{\mathrm{L}} = \sum_{j:(n,j)\in\Theta} P_n^{\mathrm{B}} \lambda_{n,j} T_j^{\mathrm{L}} p^{\mathrm{L}}. \tag{4.4.3}$$

The unblocked tasks in Θ are offloaded through the wireless channels to the ES. The mean power consumption of the MDs in BS n for uploading these tasks is

$$E_n^{\mathrm{T}} = \sum_{j:(n,j)\in\Theta} (1 - P_n^{\mathrm{B}}) \lambda_{n,j} \, \bar{t}_{n,j}^{\mathrm{W}} \, p^{\mathrm{T}}, \tag{4.4.4}$$

where $\bar{t}_{n,j}^{\mathrm{W}}$ is the mean of $t_{n,j}^{\mathrm{W}}$, and can be calculated based on the statistics of the channel conditions as shown in Section 4.6.1.

In the soft task deadlines case, the objective of minimizing the mean MD power consumption is subject to the delay constraints

$$\Pr[t_{n,j}^{\mathrm{OFF}} \le d_j] = \Pr[t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}} \le d_j] \ge 1 - \varepsilon_j \tag{4.4.5}$$

for all n and j. Finding the distribution of $t_{n,j}^{\mathrm{OFF}}$ is given in Section 4.6.4.

4.4.2 Hard deadlines

The objective of minimizing the mean power consumption of the MDs can be written as

$$\min E^{\mathrm{MD}} = E^{\mathrm{DL}} + E^{\mathrm{L}} + E^{\mathrm{T}} + E^{\mathrm{CLE}}, \tag{4.4.6}$$

where the expressions for E^{DL}, E^{L}, and E^{T} are same as in the soft deadline case. When the task deadlines are *hard* and must be respected, CLE Hekmati *et al.* (2020) is used, i.e., the local execution of a task belonging to class j is initiated at $t_j^{\mathrm{CLE}} = d_j - T_j^{\mathrm{L}}$, if offloading is still ongoing.

The additional power consumption of the MDs incurred by CLE is

$$E^{\mathrm{CLE}} = \sum_{(n,j)\in\Theta} \left(E_{n,j}^{\mathrm{O}} + E_{n,j}^{\mathrm{B}}\right), \tag{4.4.7}$$

where $E_{n,j}^{\mathrm{O}}$ is the amount of local execution power when offloading is completed before the task completion deadline and $E_{n,j}^{\mathrm{B}}$ is the amount of local execution power when offloading continues beyond the task deadline.

When $t_{n,j}^{\mathrm{OFF}} < d_j$, the local execution time incurred by CLE is $t_{n,j}^{\mathrm{OFF}} - t_j^{\mathrm{CLE}}$ because the CLE stops as soon as the offloading is completed; and when $t_{n,j}^{\mathrm{OFF}} \geq d_j$, the local execution time incurred by CLE is T_j^{L}. Based on these two cases, $E_{n,j}^{\mathrm{O}}$ and $E_{n,j}^{\mathrm{B}}$ are

$$E_{n,j}^{\mathrm{O}} = (1-P_n^{\mathrm{B}})\lambda_{n,j}p^{\mathrm{L}}\int_{t=t_j^{\mathrm{CLE}}}^{d_j}(t - t_j^{\mathrm{CLE}})\Pr\left[t_{n,j}^{\mathrm{OFF}} = t\right]dt, \tag{4.4.8}$$

$$E_{n,j}^{\mathrm{B}} = (1 - P_n^{\mathrm{B}})\lambda_{n,j}p^{\mathrm{L}}T_j^{\mathrm{L}}\int_{d_j}^{\infty}\Pr\left[t_{n,j}^{\mathrm{OFF}} = t\right]dt. \tag{4.4.9}$$

4.5 Class Partitioning

One of the main contributions of this work is the development of class partitioning algorithms that lead to the efficient implementation of the two-step computational offloading decisions described in Section 4.4. Specifically, the algorithms determine to partition the set $(\mathcal{N} \times \mathcal{J})$ into sets Θ and $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$, so that tasks belonging to class j in BS n proceed to the second step of offloading decisions if $(n, j) \in \Theta$, while other tasks are automatically executed locally.

Instead of going through all $2^{NJ} + 1$ subsets of $\mathcal{N} \times \mathcal{J}$ in order to find the best Θ, we first sort the NJ (n, j)-tuples in $\mathcal{N} \times \mathcal{J}$ into an ordered list $\eta_1, \eta_2, \ldots, \eta_{NJ}$ according to a certain criterion, and then partition this list into subsets $\Theta = \{\eta_1, \eta_2, \ldots, \eta_\theta\}$ and $(\mathcal{N} \times \mathcal{J}) \setminus \Theta = \{\eta_{\theta+1}, \eta_{\theta+2}, \ldots, \eta_{NJ}\}$, where $\theta = |\Theta|$. The rest of the section introduces sorting criteria and methods to determine θ.

4.5.1 Partitioning based on delay constraints (PDC)

The first (and simpler) method orders task classes according to the delay constraints (4.4.5). The intuition behind it is that if there is a class of tasks with extremely tight deadline, they should not be allowed to offload, since it will most likely require too many resources to satisfy their deadline constraint, assuming the deadline can indeed be satisfied by offloading. As a result, the system will offload fewer tasks (with the rest executed locally on MDs). Even if such tasks are offloaded (possibly at great cost), there is still a high probability that offloading will fail to satisfy their tight delay constraints due to the stochastic nature of the service system.

The PDC method includes two algorithms, PDC-S for SDs given in Algorithm 3 and PDC-H for HDs given in Algorithm 4.

Soft deadlines

Let λ^{ES} be the aggregate mean task arrival rate at the ES. We notice that $\Pr[t^{\mathrm{W}}_{n,j} + w^{\mathrm{C}} + T^{\mathrm{C}}_j \leq d_j]$ in (4.4.5) is a decreasing function of λ^{ES}. Let μ^{C}_j be the number of class j tasks that the ES can process in one second, i.e., $\mu^{\mathrm{C}}_j = \frac{1}{T^{\mathrm{C}}_j}$. If one assumed that all offloaded tasks arriving at the ES come from a single BS n and belong to a single task class j, binary search in the range $[0, \mu^{\mathrm{C}}_j]$ can be used to determine the maximum λ^{ES} that would satisfy delay constraint (4.4.5) in this case. Namely,

$$\hat{\lambda}^{\mathrm{ES}}_{n,j} = \max\{\lambda^{\mathrm{ES}} : \Pr\left[t^{\mathrm{W}}_{n,j} + t^{\mathrm{C}}_j \leq d_j\right] \geq 1 - \varepsilon_j\}. \tag{4.5.1}$$

Note that although the values $\hat{\lambda}^{\mathrm{ES}}_{n,j}$, calculated by (4.5.1), correspond to the simple scenario of one BS and one task class, they can be used as an indication about the possibility that tasks belonging to different classes can meet their delay constraints if offloaded. Therefore, we use them in the general setting below when making class partitioning decisions.

If $\hat{\lambda}^{\mathrm{ES}}_{n,j} = 0$ for a given (n, j)-tuple, tasks in class j from BS n are executed locally, i.e., these (n, j)-tuples will not be included in Θ. This direct assignment to local execution makes sense, since the delay constraints for these tasks cannot be satisfied, even if they were the only kind of tasks offloaded to the ES. Next, a set $\mathcal{R}$ is formed to include all (n, j)'s with $\hat{\lambda}^{\mathrm{ES}}_{n,j} > 0$ as shown in line 2 of Algorithm 3, where $R = |\mathcal{R}|$. The task classes in set $\mathcal{R}$ are ordered in decreasing values of $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ (line 3), since the smaller $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ is, the tighter the corresponding delay constraint is. The algorithm employs a linear search process for the best θ, starting from 0 and going up to R. For a given θ, Θ is the set of the first θ (n, j)-tuples in the ordering of $\mathcal{R}$. Note that when $\theta = 0$, no tasks are allowed to offload; and when $\theta = R$, all the tasks with $\hat{\lambda}^{\mathrm{ES}}_{n,j} > 0$ are allowed to offload. The algorithm compares

the resulting mean MD power consumption for each θ, and returns the Θ corresponding to the θ that achieved the minimum mean MD power consumption.

Algorithm 3 PDC-S

Input: $\lambda_{n,j}, p^{\mathrm{T}}, p^{\mathrm{L}}, P_n^{\mathrm{B}}, f^{\mathrm{L}}, f^{\mathrm{C}}, s_j, d_j, q_j, \varepsilon_j$, PDFs of $t^{\mathrm{W}}, w^{\mathrm{C}}$
Output: task class partition $\Theta^* = \{\eta_1, \eta_2, \ldots, \eta_{\theta^*}\}$

Find $\hat{\lambda}_{n,j}^{\mathrm{ES}}$, for all n and j, from (4.5.1)
$\mathcal{R} = \{(n,j) : \hat{\lambda}_{n,j}^{\mathrm{ES}} > 0\}$; $R = |\mathcal{R}|$
Sort classes in $\mathcal{R}$ in decreasing order of $\hat{\lambda}_{n,j}^{\mathrm{ES}}$; let $\eta_1, \eta_2, \ldots, \eta_R$ be the resulting ordering
$\theta^* = 0$; $E_{\min} = \infty$
for all $\theta = 0, 1, 2, \ldots, R$ **do**
    $\Theta = \{\eta_1, \eta_2, \ldots, \eta_\theta\}$
    Find mean MD power consumption E^{MD}
    if $E^{\mathrm{MD}} < E_{\min}$ **then**
        $E_{\min} = E^{\mathrm{MD}}$; $\theta^* = \theta$
    end if
end for
return $\Theta^* = \{\eta_1, \eta_2, \ldots, \eta_{\theta^*}\}$

Hard deadlines

While (4.4.5) is an explicit constraint in the problem formulation for soft deadlines, the hard deadlines formulation does not contain such a constraint, since CLE guarantees the task completion before its deadline. However, (4.4.5) (and (4.5.1)) can still be used to obtain an ordering of the (n,j)-tuples. The basic procedure is as follows. Starting from $\Theta = \mathcal{N} \times \mathcal{J}$ and $\varepsilon_j = \varepsilon = 1$ for all j, the common ε is reduced in steps of size ω. For every reduction, we calculate values $\hat{\lambda}_{n,j}^{\mathrm{ES}}$ exactly as in the SD case by using the current ε in constraints (4.5.1). For each (n,j) with $\hat{\lambda}_{n,j}^{\mathrm{ES}} = 0$, we remove (n,j) from Θ. This process continues, until θ classes remain in Θ. Algorithm PDC-H (Algorithm 4) implements this process. Note that in Algorithm 4, θ decreases from NJ down to 0 (instead of increasing

from 0 to NJ), because this allows us to calculate our solutions for different θ with only one swiping decrease of ε down from 1.

Algorithm 4 PDC-H

Input: $\lambda_{n,j}, p^{\mathrm{T}}, p^{\mathrm{L}}, P_n^{\mathrm{B}}, f^{\mathrm{L}}, f^{\mathrm{C}}, s_j, d_j, q_j$, PDFs of $t^{\mathrm{W}}, w^{\mathrm{C}}$
Output: task class partition Θ^*

$\Theta = \mathcal{N} \times \mathcal{J}$; $E_{\min} = \infty$
$\varepsilon_j = \varepsilon = 1,\ \forall j \in \mathcal{J}$
for all $\theta = NJ, NJ-1, \ldots, 2, 1, 0$ **do**
  while $|\Theta| > \theta$ **do**
    for all $(n, j) \in \mathcal{N} \times \mathcal{J}$ **do**
      Find $\hat{\lambda}_{n,j}^{\mathrm{ES}}$ from (4.5.1)
      if $\hat{\lambda}_{n,j}^{\mathrm{ES}} = 0$ **then**
        $\Theta = \Theta \setminus \{(n, j)\}$
      end if
      if $|\Theta| = \theta$ **then**
        break;
      end if
    end for
    $\varepsilon = \varepsilon - \omega$; $\varepsilon_j = \varepsilon$ for all $j \in \mathcal{J}$
  end while
  Find mean MD power consumption E^{MD}
  if $E^{\mathrm{MD}} < E_{\min}$ **then**
    $E_{\min} = E^{\mathrm{MD}}$; $\Theta^* = \Theta$
  end if
end for
return Θ^*

4.5.2 Hierarchical partitioning (HP)

Unlike the PDC algorithms, the hierarchical partitioning algorithm takes into account both the mean power consumption and the task deadlines when ordering the tasks. This is implemented in two phases. In the first phase, task classes are ordered and grouped according to an estimated power consumption criterion. In the second phase, classes within each group

are further ordered using a second criterion that takes task deadlines into account. This partitioning method is given in Algorithm 5, where g is the index of the current group of (n, j)-tuples and h is the index of the first (n, j)-tuple in the current group.

In the first phase of ordering, we prefer to offload the tasks that can save more power by offloading than by executing locally. Thus, the sorting criterion is the difference between power consumption of the MD for offloading the task and executing it locally, which is defined as

$$\Phi^{\mathrm{E}}_{n,j} = p^{\mathrm{T}} \frac{s_j}{\bar{B}_{n,j}} - p^{\mathrm{L}} T^{\mathrm{L}}_j, \tag{4.5.2}$$

where $\bar{B}_{n,j}$ is the expected wireless transmission rate of class j tasks in BS n with the expectation taken over the random channel conditions. An example is provided in Section VIII. The (n, j)-tuples of $\mathcal{N} \times \mathcal{J}$ are ordered in increasing values of $\Phi^{\mathrm{E}}_{n,j}$. After this initial ordering, the (n, j)-tuples whose $\Phi^{\mathrm{E}}_{n,j}$ values differ by at most ξ (for some parameter $0 \leq \xi \leq 1$) are grouped together (lines 2-9).

Within the same group, the (n, j)-tuples are further ordered according to a second set of values defined as follows:

$$\Phi^{\mathrm{D}}_{n,j} = \frac{s_j/\bar{B}_{n,j} + q_j/(yf^C)}{d_j} = \frac{s_j}{\bar{B}_{n,j} d_j} + \frac{q_j}{yf^C d_j}. \tag{4.5.3}$$

That is, $\Phi^{\mathrm{D}}_{n,j}$ is a ratio between the minimum mean offloading time (sum of mean wireless transmission time and ES execution time) over the deadline of class j tasks. Note that $\Phi^{\mathrm{D}}_{n,j}$ does not consider any queueing delay at the ES. This is because the ES service system has a single queue for all offloaded tasks, so that all task classes experience the same mean queueing delay, and as a result, queueing delay does not affect the task class ordering.

Within each group, the (n, j)-tuples are sorted in increasing values of $\Phi^{\mathrm{D}}_{n,j}$ (lines 10-13).

Parameter ξ regulates the trade-off between the effects of delay constraints and power consumption on class ordering. When $\xi = 0$, there is one and only one BS-class in each group, and the sorting is based only on $\Phi^{\mathrm{E}}_{n,j}$, i.e., the second phase does not have any effect. When $\xi = 1$, all the BS-classes belong in one group after the first phase, and the sorting is solely based on $\Phi^{\mathrm{D}}_{n,j}$ in the second phase.

Finally, lines 15-22 find the partition that achieves the minimum MD power consumption.

Running times: Let $\mathcal{T}$ be the running time of finding the mean MD power consumption in line 7 of Algorithm 3, line 16 in Algorithm 4, and line 18 in Algorithm 5.

- For PDC-S given in Algorithm 3, line 1 takes time $O(NJ \log \frac{\mu^{\mathrm{C}}}{\epsilon})$, where μ^{C} is the maximum of μ^{C}_j and ϵ is the desired accuracy of the binary search, and line 3 takes time $O(NJ \log NJ)$. The running time of PDC-S is $O(NJ \log \frac{\mu^{\mathrm{C}}}{\epsilon} + NJ \log NJ + NJ\mathcal{T})$.

- For PDC-H given in Algorithm 4, within the for loop between lines 3 and 20, the running time of lines 4-15 is $O(\frac{1}{\omega} NJ \log \frac{\mu^{\mathrm{C}}}{\epsilon})$. The running time of PDC-H is $O(\frac{1}{\omega}(NJ)^2 \log \frac{\mu^{\mathrm{C}}}{\epsilon} + NJ\mathcal{T})$.

- For HP, the running time of Algorithm 5 is $O(NJ \log(NJ) + NJ\mathcal{T})$.

Observation: In line 7 of Algorithm 3, line 16 of Algorithm 4, and line 18 of Algorithm 5, finding the mean MD power consumption E^{MD} requires specific information regarding the task classes, wireless channel statistics, resource availability of the network, etc. In the following two sections, the mean MD power consumption is found for two scenarios, one with fixed x_n's and y, and the other with a cost budget that limits the total cost

Algorithm 5 HP

Input: $\lambda_{n,j}, p^{\mathrm{T}}, p^{\mathrm{L}}, P_n^{\mathrm{B}}, f^{\mathrm{L}}, f^{\mathrm{C}}, s_j, d_j, q_j, \bar{B}_{n,j}, \xi$, PDFs of $t^{\mathrm{W}}, w^{\mathrm{C}}$
Output: task class partition $\Theta^* = \{\eta'_1, \eta'_2, \ldots, \eta'_{\theta^*}\}$

Sort the (n,j)-tuples in $\mathcal{N} \times \mathcal{J}$ in increasing order of $\Phi^{\mathrm{E}}_{n,j}$; let $\eta_1, \eta_2, \cdots, \eta_{NJ}$ be the resulting ordering
$g = 1$; $h = 1$; $\mathcal{G}_1 = \{\eta_1\}$
for $j = 2$ **to** NJ **do**
    if $\left|\frac{\Phi^{\mathrm{E}}_{\eta_j} - \Phi^{\mathrm{E}}_{\eta_h}}{\Phi^{\mathrm{E}}_{\eta_h}}\right| < \xi$ **then**
        $\mathcal{G}_g = \mathcal{G}_g \cup \{\eta_j\}$
    else
        $g = g + 1$; $h = j$; $\mathcal{G}_g = \{\eta_h\}$
    end if
end for
$\mathcal{L} = \mathrm{null}$ ▷ $\mathcal{L}$ is a list of (n,j)-tuples
for $g = 1$ **to** G **do**
    Sort the (n,j)-tuples in group g in increasing order of $\Phi^{\mathrm{D}}_{n,j}$ and append the ordered tuples to $\mathcal{L}$
end for
Let $\mathcal{L} = \eta'_1, \eta'_2, \cdots, \eta'_{NJ}$ be the ordered (n,j) list
$\theta^* = 0$; $E_{\min} = \infty$
for $\theta = 0$ **to** NJ **do**
    $\Theta = \{\eta'_1, \eta'_2, \ldots, \eta'_\theta\}$
    Find mean MD power consumption E^{MD}
    if $E^{\mathrm{MD}} < E_{\min}$ **then**
        $E_{\min} = E^{\mathrm{MD}}$; $\theta^* = \theta$
    end if
end for
return $\Theta^* = \{\eta'_1, \eta'_2, \ldots, \eta'_{\theta^*}\}$

of the wireless channels and ES computing resources.

4.6 Preallocated Network Resources

In this section, we consider the BSTCP problem, i.e., the case where the available resources for the network have been preallocated, i.e., x_n's and y are fixed and given. We assume that

the arrival process of class j tasks at BS n follows a Poisson process with mean arrival rate $\lambda_{n,j}$. The assumption is a common one, and justified by the large number of MDs that generate the tasks (Khintchine (1932)). As a result, and because of the PASTA rule (Wolff (1982)), the ES service system can be modeled as an $M/G/1$ queue. We model the wireless channels between the MDs and the BSs as discrete-time Markov processes. The time slot duration is denoted by τ in seconds. The Markovian transition probabilities are defined in the usual way, i.e., given the channel state in the current time slot, there is a probability associated to its transition to another state in the next time slot. These probabilities are functions of the radio propagation environment that the MDs experience at the BS. Let $\mathcal{K}_n = \{1, 2, \ldots, K_n\}$ be the set of different possible Markovian wireless channel models of BS n, and let $K_n = |\mathcal{K}_n|$.

4.6.1 Distributions of $t^{\mathrm{W}}_{n,j}$

Define $P^{\mathrm{G}}_{n,j,k}$ as the probability that a class j task, offloaded by an MD in BS n, encounters channel model k. The distribution of $t^{\mathrm{W}}_{n,j}$ is given as

$$\Pr[t^{\mathrm{W}}_{n,j} = l] = \sum_{k \in \mathcal{K}_n} \Pr[t^{\mathrm{W}}_{n,j,k} = l] P^{\mathrm{G}}_{n,j,k}, \tag{4.6.1}$$

where $t^{\mathrm{W}}_{n,j,k}$ (in number of time slots) is the amount of time for uploading a class j task in BS n when the channel follows propagation model k. Given the Markov channel state transition probabilities, the distribution of $t^{\mathrm{W}}_{n,j,k}$ is given in (Hekmati *et al.* (2020)). The mean of $t^{\mathrm{W}}_{n,j}$ can be found as

$$\bar{t}^{\mathrm{W}}_{n,j} = \sum_{l=0}^{\infty} l \Pr[t^{\mathrm{W}}_{n,j} = l]. \tag{4.6.2}$$

4.6.2 Computing P_n^{B}

P_n^{B} is channel blocking probability for tasks in BS n with $(n,j) \in \Theta$. For BS n, the mean task arrival rate for all classes with $(n,j) \in \Theta$ is $\sum_{j:(n,j)\in\Theta} \lambda_{n,j}$, and the mean service time (uploading time through wireless transmissions) of the tasks is

$$\bar{t}_n^{\mathrm{W}} = \frac{\sum_{j:(n,j)\in\Theta} \lambda_{n,j} \bar{t}_{n,j}^{\mathrm{W}}}{\sum_{j:(n,j)\in\Theta} \lambda_{n,j}}. \tag{4.6.3}$$

Given that the task arrivals follow a Poisson process, the Erlang-B formula can be used to find P_n^B even for non-exponentially distributed task uploads, as in (Chen *et al.* (2023)). This is referred to as the *insensitivity property* of the formula (Burman (1981)). Thanks to this property, the Erlang-B result holds for any service time distribution with the same mean.

4.6.3 Distribution of w^{C}

The aggregate task arrival process at ES is Poisson (Shanbhag and Tambouratzis (1973)), and the service system at the ES is an $M/G/1$ queue. As arriving tasks sample the asymptotic equilibrium state distribution of the ES, the statistics of the queueing time experienced at the ES by tasks in different classes and from different BSs are the same. The queueing time distribution depends on the mean aggregate task arrival rate at the ES and the distribution of the execution time of the tasks.

The mean task arrival rate at the ES is

$$\lambda^{\mathrm{ES}} = \sum_{(n,j)\in\Theta} \lambda_{n,j}(1 - P_n^{\mathrm{B}}). \tag{4.6.4}$$

Let T^{C} be a random variable representing the execution time of a task arriving at the ES. The probability that $T^{\mathrm{C}} = T_j^{\mathrm{C}}$ is equal to the probability that a class j task arrives at the ES. Since

$$\Pr[T^{\mathrm{C}} = T_j^{\mathrm{C}}] = \frac{\sum_{n:(n,j)\in\Theta}(1 - P_n^{\mathrm{B}})\lambda_{n,j}}{\lambda^{ES}}, \tag{4.6.5}$$

the probability density function of T^{C} is

$$f_{T^{\mathrm{C}}}(\tilde{b}) = \sum_{j:(n,j)\in\Theta} \Pr\left[T^{\mathrm{C}} = T_j^{\mathrm{C}}\right] \delta\left(\tilde{b} - T_j^{\mathrm{C}}\right), \tag{4.6.6}$$

and the Laplace-Stieltjes transform of $f_{T^{\mathrm{C}}}(\tilde{b})$ is given by

$$g(s) = \sum_{j:(n,j)\in\Theta} \Pr[T^{\mathrm{C}} = T_j^{\mathrm{C}}] e^{-T_j^{\mathrm{C}} s}. \tag{4.6.7}$$

For the queuing time w^{C}, the Laplace-Stieltjes transform of its probability density function is given by the Pollaczek-Khinchine transform (Khintchine (1932)) as

$$W^*(s) = \frac{(1 - \lambda^{\mathrm{ES}}\bar{T}^{\mathrm{C}})s}{s - \lambda^{\mathrm{ES}}(1 - g(s))}, \tag{4.6.8}$$

where $\bar{T}^{\mathrm{C}}$ is the mean of T^{C}. The distribution of w^{C} can be obtained by numerical inversion of (4.6.8).

4.6.4 Distribution of $t_{n,j}^{\mathrm{OFF}}$

Given the distribution of $t_{n,j}^{\mathrm{W}}$ and w^{C}, we can find the distribution of $t_{n,j}^{\mathrm{OFF}} = t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}}$. Note that $t_{n,j}^{\mathrm{W}}$ and w^{C} are independent since the wireless channel uploading time is

independent of the ES service time.

In the soft deadline case, the joint probability distribution of total offloading delay $t_{n,j}^{\mathrm{OFF}}$ is

$$\begin{aligned}\Pr[t_{n,j}^{\mathrm{OFF}} \leq d_j] &= \Pr[t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}} \leq d_j] = \Pr[t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} \leq d_j - T_j^{\mathrm{C}}] \\ &= \sum_{l=1}^{\lfloor (d_j - T_j^{\mathrm{C}})/\tau \rfloor} \Pr[t_{n,j}^{\mathrm{W}} = l] \Pr[w^{\mathrm{C}} \leq d_j - T_j^{\mathrm{C}} - l\tau]. \end{aligned} \tag{4.6.9}$$

In the hard deadline case, in order to calculate the power consumption of MDs based on CLE, time-related variables are discretized into multiples of τ with $\tilde{a}$ be the discretized version of a. For d_j and T_j^{C}, their discretized versions are given as $\tilde{d}_j = \lfloor d_j/\tau \rfloor$ and $\tilde{T}_j^{\mathrm{C}} = \left\lceil \frac{T_j^{\mathrm{C}}}{\tau} \right\rceil$. Note that the discretization of d_j takes the floor of d_j/τ while that of T_j^{C} takes the ceiling of T_j^{C}/τ in order to respect the task delay constraint. With this, the distribution of the discretized version of $t_{n,j}^{\mathrm{OFF}}$ is

$$\Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right] = \sum_{l=1}^{t-\tilde{T}_j^{\mathrm{C}}} \Pr[t_{n,j}^{\mathrm{W}} = l] \Pr[w^{\mathrm{C}} = t - \tilde{T}_j^{\mathrm{C}} - l]. \tag{4.6.10}$$

4.6.5 Computing $E_{n,j}^{\mathrm{O}}$ and $E_{n,j}^{\mathrm{B}}$

With the discretization of time, the integrals in (4.4.8) and (4.4.9) become summations as follows:

$$E_{n,j}^{\mathrm{O}} = (1 - P_n^{\mathrm{B}})\lambda_{n,j}\, p^{\mathrm{L}} \sum_{t=\tilde{t}_j^{\mathrm{CLE}}}^{\tilde{d}_j} (t - \tilde{t}_j^{\mathrm{CLE}} + 1) \Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right], \tag{4.6.11}$$

$$E_{n,j}^{\mathrm{B}} = (1 - P_n^{\mathrm{B}})\lambda_{n,j}\, p^{\mathrm{L}}\, T_j^{\mathrm{L}} \sum_{t=\tilde{d}_j+1}^{\infty} \Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right], \tag{4.6.12}$$

where $\tilde{t}_j^{\mathrm{CLE}}$ is the discretized value of t_j^{CLE} in number of time slots, i.e., $\tilde{t}_j^{\mathrm{CLE}} = \left\lceil t_j^{\mathrm{CLE}}/\tau \right\rceil$. In (4.6.11) and (4.6.12), $\Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right]$ is given by (4.6.10).

4.7 Joint Task Class Partitioning and Network Resource Allocation

In this section we consider the JSTCP problem, i.e., solving TCP in a system similar to (Chen *et al.* (2023)). Instead of having a fixed number of channels from each BS and a fixed amount of computing resources at the ES, the considered network can lease channels from different BSs and computing resources from the ES, provided the total cost is within a predefined budget $B_{\max}$. Note that this budget is normalized to the monetary cost for a given time period, such as a day or a month. There are up to M_n channels at BS n, that can be selected. The cost of renting a channel from BS n is α_n. In order to use the computing resources at the ES, CPU resources must also be leased at the ES. The cost (based on the number of CPU cycles per second) for leasing on the CPU resource is denoted by β. The fraction of available CPU speed for rental is $y \in [0, 1]$, i.e., the CPU speed available is yf^{C}. Both x_n's and y are normalized to their respective monetary cost for the same time period as $B_{\max}$. In this case, the following constraint should be satisfied:

$$\sum_{n=1}^{N} \alpha_n x_n + \beta y f^{\mathrm{C}} \leq B^{\max}. \tag{4.7.1}$$

In this system, $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ and y are decision variables, P_n^{B}, w^{C}, and $t_{n,j}^{\mathrm{OFF}}$ all become functions of $\mathbf{x}$, y, and the task class partitioning, which complicates the calculations and the problem becomes joint task class partitioning and resource allocations.

Algorithms PDC-S and PDC-H require the computation of values $\hat{\lambda}_{n,j}^{\mathrm{ES}}$, given by (4.5.1). We make the simplifying assumption that the task distributions in each BS are all the same, i.e., the probability that a task arriving at a BS belongs to a certain class task $j \in \mathcal{J}$ is the same for all BSs. We also assume that this probability is known, e.g., by observing the past history of offloading requests.

Under these assumptions, criterion (4.5.1) is simplified as follows. We first calculate the values $\hat{\lambda}_{n,j}^{\mathrm{ES}}$ as in (4.5.1), and then define

$$\hat{\lambda}_{j}^{\mathrm{ES}} = \min_{n} \hat{\lambda}_{n,j}^{\mathrm{ES}}, \tag{4.7.2}$$

for all $j \in \mathcal{J}$, as the values used for ordering the task classes. In addition, the criteria used in the HP algorithms will be changed from $\Phi_{n,j}^{\mathrm{E}}$ and $\Phi_{n,j}^{\mathrm{D}}$ to Φ_{j}^{E} and Φ_{j}^{D}, respectively. As a result, in what follows we will be considering $\mathcal{J}$ instead of $\mathcal{N} \times \mathcal{J}$, and $\Theta \subseteq \mathcal{J}$.

4.7.1 Soft Deadlines

For the soft task deadline case, we rewrite the mean power consumption of the MDs in (4.4.1) as

$$E^{\mathrm{MD}} = E_{\Theta}^{\mathrm{MD}} + E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}, \tag{4.7.3}$$

where E_{Θ}^{MD} is the MD power consumption for the tasks belonging to classes in Θ, i.e., $E_{\Theta}^{\mathrm{MD}} = E^{\mathrm{L}} + E^{\mathrm{T}}$, and $E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}$ is the local processing power consumption of the MDs for the

tasks belonging to classes in $\mathcal{J} \setminus \Theta$, given by

$$E^{\mathrm{DL}}_{\mathcal{J}\setminus\Theta} = \sum_{n=1}^{N} \sum_{j\in\mathcal{J}\setminus\Theta} \lambda_{n,j} T^{\mathrm{L}}_j p^{\mathrm{L}}. \tag{4.7.4}$$

The general formulation of the joint class partitioning and resource allocation problem with soft deadlines is the following:

$$\min_{\Theta,\mathbf{x},y} E^{\mathrm{MD}}_{\Theta} + E^{\mathrm{DL}}_{\mathcal{J}\setminus\Theta} \text{ s.t.} \tag{4.7.5}$$

$$\sum_{n=1}^{N} \alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{4.7.6}$$

$$\Pr[t^{\mathrm{OFF}}_{n,j} \leq d_j] \geq 1 - \varepsilon_j, \forall n, j \tag{4.7.7}$$

$$x_n \in \{0, 1, \ldots, M_n\}, \forall n = 1, 2, \ldots, N \tag{4.7.8}$$

$$0 \leq y \leq 1 \tag{4.7.9}$$

$$\Theta \subseteq \mathcal{J} \tag{4.7.10}$$

Note that $\mathbf{x}$ and y do not affect $E^{\mathrm{DL}}_{\mathcal{J}\setminus\Theta}$. In general, problem (4.7.5)-(4.7.10) is computationally hard to solve exactly over all possible values of $\mathbf{x}, y, \Theta$. Instead, we would like to obtain an approximate solution, based on limiting Θ to the $J + 1$ subsets for a single task classes ordering.

Different y values may result in different orderings of task classes. Therefore, Algorithm PDC-S (Algorithm 3) and Algorithm HP (Algorithm 5) need to be incorporated into the approximation algorithm proposed in (Chen *et al.* (2023)) for computing a good solution $\mathbf{x}, y$. More specifically, the algorithm of (Chen *et al.* (2023)) discretizes variable $y \in [0, 1]$ by breaking $[0, 1]$ into equal segments. For each y, we use Algorithm PDC-S or Algorithm HP to obtain an ordered task class list. For each such Θ, line 7

of Algorithm PDC-S (Algorithm 3) and line 18 of Algorithm HP (Algorithm 5) become $E^{\mathrm{MD}} = E^{\mathrm{DL}}_{\mathcal{J}\setminus\Theta} + E^{\mathrm{MD}*}_{\Theta}$, where $E^{\mathrm{MD}*}_{\Theta}$ is the minimum mean power consumption of all MDs for task classes in Θ. The algorithm of (Chen *et al.* (2023)) is used to find an $\mathbf{x}$ close to the minimum power consumption $E^{\mathrm{MD}*}_{\Theta}$, i.e., for every y and every Θ of the ordering induced by y, a good $\mathbf{x}$ is computed. The details are briefly summarized as follows: The original problem (4.7.5)-(4.7.10) can be relaxed, by relaxing variables $0 \leq x_n \leq M_n$. With y and Θ fixed, the original relaxation is transformed into a convex program, by noticing that $\Pr[t^{\mathrm{OFF}}_{n,j} \leq d_j]$ is a monotonically decreasing function of the aggregate mean task arrival rate λ^{ES}. Therefore, binary search in the range $[0, yf^{\mathrm{C}}/\bar{q}]$, where $\bar{q}$ is the average computation load of a task that can be easily computed, can be used to approximately calculate the maximum possible value λ^* that satisfies constraints (4.7.7) for all n, j. Using (4.6.4), constraints (4.7.7) can be replaced by constraint

$$\sum_{(n,j)\in\Theta} \lambda_{n,j}(1 - P^{\mathrm{B}}_n) \leq \lambda^*. \tag{4.7.11}$$

Next, we note that the blocking probability P^{B}_n is monotonically decreasing in x_n. Let $P^{\mathrm{B}}_{n,\min}$ be the blocking probability when $x_n = M_n$, then the relaxation constraints $0 \leq x_n \leq M_n$ can be replaced by the equivalent constraints

$$P^{\mathrm{B}}_{n,\min} \leq P^{\mathrm{B}}_n \leq 1, \ \forall n \in \mathcal{N}. \tag{4.7.12}$$

Finally, x_n in constraint (4.7.6) can be replaced with any convex upper bound $F(P^{\mathrm{B}}_n)$ under the assumptions of Section 4.6, e.g., the one in (Berezner *et al.* (1998)): $x_n \leq$

$(\sum_{j:(n,j)\in\Theta}\lambda_{n,j})\bar{t}_n^{\mathrm{W}}(1-P_n^{\mathrm{B}})+1/P_n^{\mathrm{B}},\ \forall n$. After solving the new convex optimization problem on variables P_n^{B} that approximates the original one when y and Θ are fixed, and obtaining the P_n^{B}'s, we can compute the largest integral x_n^* which achieves a blocking probability equal to or bigger than P_n^{B}, for all $n \in \mathcal{N}$. In the end, the algorithm outputs the combination $\mathbf{x}^*, y^*, \Theta^*$ that achieves the minimum mean power consumption.

4.7.2 Hard Deadlines

The mean MD power consumption in the hard deadline case can be written as in (4.7.3), and the general joint resource allocation and class partitioning problem with hard deadlines is the following:

$$\min_{\Theta,\mathbf{x},y} E^{\mathrm{MD}} = E_{\Theta}^{\mathrm{MD}} + E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}} \text{ s.t.} \tag{4.7.13}$$

$$\sum_{n=1}^{N}\alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{4.7.14}$$

$$x_n \in \{0,1,\ldots,M_n\}, \forall n = 1,2,\ldots,N \tag{4.7.15}$$

$$0 \leq y \leq 1 \tag{4.7.16}$$

$$\Theta \subseteq \mathcal{J} \tag{4.7.17}$$

where $E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}$ is the same as in (4.7.4), and $E_{\Theta}^{\mathrm{MD}} = E^{\mathrm{L}} + E^{\mathrm{T}} + E^{\mathrm{CLE}}$.

Algorithms 4 or 5, together with the solution method of (Chen *et al.* (2023)) are used in this case in the similar way as in the soft deadline case, to produce good approximate solutions. More specifically, in addition to discretizing y we also discretize $\lambda^{\mathrm{ES}} \in [0, yf^{\mathrm{C}}/\bar{q}]$ by breaking interval $[0, yf^{\mathrm{C}}/\bar{q}]$ into Λ equal segments. For each fixed y, λ^{ES} and Θ, probability $\Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right]$ can be calculated directly for any t. By using (4.7.12) and $F(P_n^{\mathrm{B}})$,

the resulting convex program can be solved efficiently. Hence, we can obtain the optimal blocking probabilities $P_n^{\mathrm{B}*}$ and compute the largest integral x_n^* which achieves blocking probabilities no smaller than $P_n^{\mathrm{B}*}$, for all $n \in \mathcal{N}$, based on the fact that the P_n^{B}'s are decreasing functions of the x_n's. After collecting the solutions for all subproblems, we output the minimum average power consumption with $\mathbf{x}^*, y^*, \Theta^*$.

4.8 Simulation Results

In this section, we present simulation results that demonstrate the performance of our algorithms. Although our algorithms can be applied to any set of Markovian channel models, for our results we use a two-state Gilbert-Elliot channel model (Gilbert (1960)), i.e., the channel states change by following a Markov chain with two states, "Good" (G) and "Bad" (B). This model is commonly used to characterize the effects of burst noise in wireless channels, where the channel can abruptly transition between good and bad conditions (Blazek and Mecklenbräuker (2018)). Since the channel can make these abrupt transitions in quality, computation offloading decisions may be more difficult compared to cases where the channel states are more consistent as the channel offloading progresses. Let B_g and B_b be the data transmission rates when the channel is in the G and B states, respectively. It is assumed that all channels have the same B_g and B_b values but differ in their state transition probabilities, which result in different propagation models. The transition probabilities for propagation model k in BS n are denoted as $P_{n,k}^{\mathrm{GG}}, P_{n,k}^{\mathrm{GB}}, P_{n,k}^{\mathrm{BG}}$, and $P_{n,k}^{\mathrm{BB}}$. In each time slot, the channel state Markov chain transitions in accordance with these probabilities. Denote $\pi_{n,k}^{\mathrm{G}}$ and $\pi_{n,k}^{\mathrm{B}}$, respectively, as the stationary probabilities of being in the G and B states for a channel following propagation model k in BS n. The average wireless transmission rate

of jth class tasks in BS n as

$$\bar{B}_{n,j} = \sum_{k=1}^{K_n} P^{\mathrm{G}}_{n,j,k}(\pi^{\mathrm{G}}_{n,k} B_g + \pi^{\mathrm{B}}_{n,k} B_b). \tag{4.8.1}$$

Given the channel state transition probabilities, the distribution of wireless transmission time $t^{\mathrm{W}}_{n,j,k}$ for uploading a class j task in BS n through a channel with propagation model k can be calculated from (Hekmati *et al.* (2020)).

We consider a network with 3 BSs. Within each BS, there are two (Markovian) channel propagation models with transition probabilities $P^{\mathrm{GG}}_{n,1} = 0.9$, $P^{\mathrm{GG}}_{n,2} = 0.6$, $P^{\mathrm{BB}}_{n,1} = 0.1$, and $P^{\mathrm{BB}}_{n,2} = 0.4$ for $n = 1, 2, 3$. We consider that for given n and k, $P^{\mathrm{G}}_{n,j,k}$ values are the same for all $j \in \mathcal{J}$. That is, the probability of accessing the channels with a given propagation model in a given BS does not depend on the task class. Therefore, $P^{\mathrm{G}}_{n,j,k}$ can be reduced to $P^{\mathrm{G}}_{n,k}$ by dropping the subscript j, and their values are given as $P^{\mathrm{G}}_{1,1} = 0.8$, $P^{\mathrm{G}}_{1,2} = 0.2$, $P^{\mathrm{G}}_{2,1} = 0.5$, $P^{\mathrm{G}}_{2,2} = 0.5$, $P^{\mathrm{G}}_{3,1} = 0.2$, and $P^{\mathrm{G}}_{3,2} = 0.8$. Other default parameter values are summarized in Table 4.1. These parameter values are similar to those used in (Yue *et al.* (2022); Pham *et al.* (2021); Li *et al.* (2022c); Alameddine *et al.* (2019)) and varied during the simulation. We intentionally use a wide range of parameter values based on the referenced ranges so that we can make conclusions that apply in general settings. In addition, we chose different sets of parameters for the task classes in order to examine the performance of the proposed solutions.

4.8.1 Task class ordering criteria

Before looking at performance of the proposed class partitioning algorithms, we first examine the performance of the different class ordering criteria used in these algorithms. For

Table 4.1: Default Parameters

Parameter	Value
τ	10 ms
p^{L}	0.5 W
p^{T}	0.1 W
$\lambda_{n,j}$	0.4 tasks/s
α_n	1 /channel
β	2 /GHz
f^{L}	200M cycles/s
ε	3 %
ξ	40 %
Bg, B_b	3M, 0.5M bits per time slot

Table 4.2: Task class parameters, set 1

s_j (M bits)	4	8	12	16	24	28	32	46	50	54
q_j (M CPU cycles)	54	50	46	32	28	24	16	12	8	4
d_j (ms)	300	250	250	160	200	160	250	300	500	500

both PDC-S and PDC-H, the ordering of task classes are based on $\hat{\lambda}_{n,j}^{\mathrm{ES}}$'s, which are calculated based on the delay constraints. For HP, the ordering is based on both MD power consumption ($\Phi_{n,j}^{\mathrm{E}}$) and task delay constraints ($\Phi_{n,j}^{\mathrm{D}}$).

We first consider preallcoated network resources with $x_n = 15$ in all the BSs and the ES capacity $yf^{\mathrm{C}} = 25$G CPU cycles/s. There are 10 task classes and their parameter values are given in Table 4.2. In order to reflect the heterogenous nature of the classes, we consider 3 categories of task classes. The first 4 classes have relatively small input data sizes and large computational loads, the next 3 classes have moderate input data sizes and computational loads, and the last 3 classes have relatively large input data sizes and small computational loads. For each of the class ordering methods, after the task classes have been ordered, the set Θ includes the first θ classes in the class list, and the average MD power consumption based on this partitioning is collected and shown in Figure 4.2.

Table 4.3: Task class parameters, set 2

10 classes										
s_j (M bits)	2	3	6	9	30	32	35	70	75	80
q_j (M CPU cycles)	40	42	45	47	21	23	25	2	4	6
d_j (ms)	200	230	250	250	200	230	250	500	500	550

Table 4.4: Task class parameters, set 3

20 classes										
s_j (M bits)	1	2	3	5	6	8	10	31	33	35
q_j (M CPU cycles)	79	72	69	67	65	64	62	39	38	38
d_j (ms)	400	400	400	400	350	350	350	250	250	250
s_j (M bits)	35	37	39	40	61	63	65	65	76	78
q_j (M CPU cycles)	36	35	34	33	10	8	6	5	3	1
d_j (ms)	250	280	280	280	300	300	500	500	600	600

Next, we consider that the available amount of network resources is optimized based on task class partitioning, class completion deadlines, and resource cost budget. That is, for each of the class ordering methods, after the classes have been ordered and the first θ classes are included into the set Θ, the average MD power consumption is minimized by optimizing x_n's and y subject to a cost budget. We consider two sets of task classes with the parameter values given in Tables 4.3 and 4.4, respectively. For the case with 10 task classes given in Tables 4.3, the total number of wireless channels in each BS is $M_n = 20$, the total ES capacity is $f^{\mathrm{C}} = 20$ G CPU cycles/s, and the resource cost budget is 100. For the case with 20 task classes given in Tables 4.4, the total number of wireless channels in each BS is $M_n = 25$, the total ES capacity is $f^{\mathrm{C}} = 40$ G CPU cycles/s, and the resource cost budget is 220. These parameter settings are similar to those in (Pham *et al.* (2021); Liu *et al.* (2020); Šlapak *et al.* (2021)). The mean MD power consumption is shown in Figure 4.3 for different θ values.

Both figures show that when θ is relatively small, increasing θ helps reduce the mean

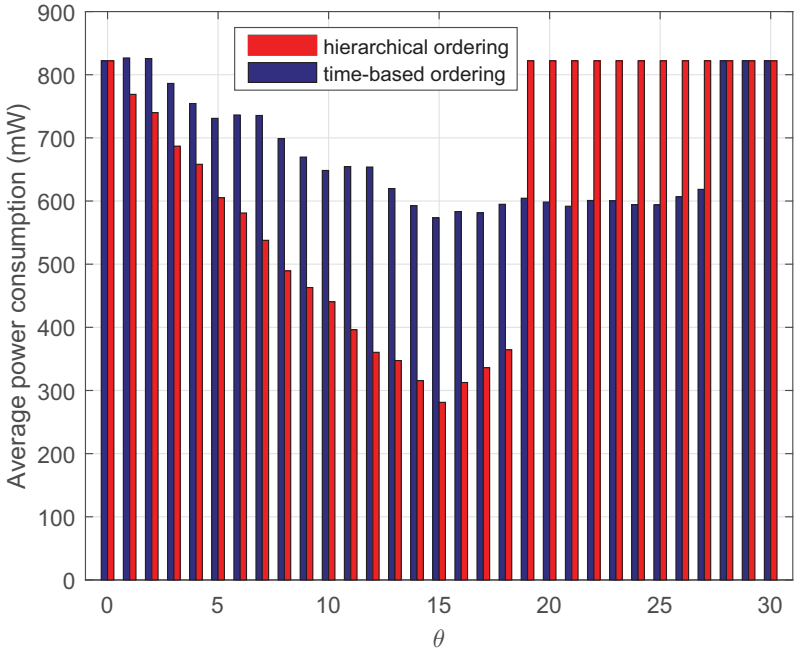
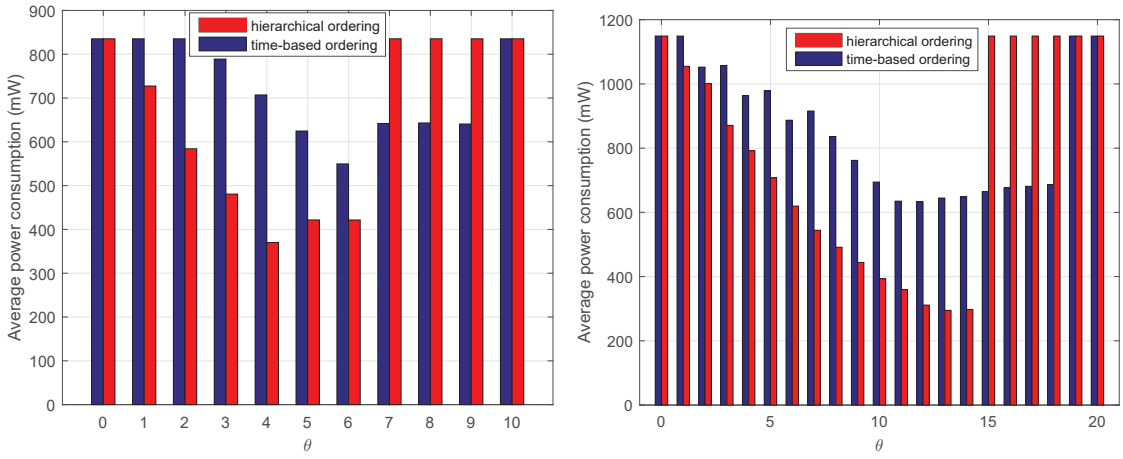


Figure 4.2: Average power consumption versus θ (BSTCP)

MD power consumption; furthermore, the power consumption decreases much faster using the hierarchical ordering than the delay-based ordering. However, as θ keeps increasing and beyond a certain value, the mean MD power consumption using both ordering criteria starts increasing, and the hierarchical ordering may result in much higher MD power consumption than the delay-based criterion. The figures demonstrate that for both class ordering criteria, allowing too many task classes to offload can increase the mean MD power consumption. This demonstrates the importance of task class partitioning. For example, Figure 4.2 shows that, for the delay-based class ordering, when θ is 28 or larger, all tasks are processed locally; and for the hierarchical class ordering, when θ is 19 or larger, all tasks are processed locally. Meanwhile, the figures also show that for each class ordering criterion, there is an optimum value of θ that minimizes the mean MD power consumption; and the minimum achievable MD power consumption using the hierarchical ordering is



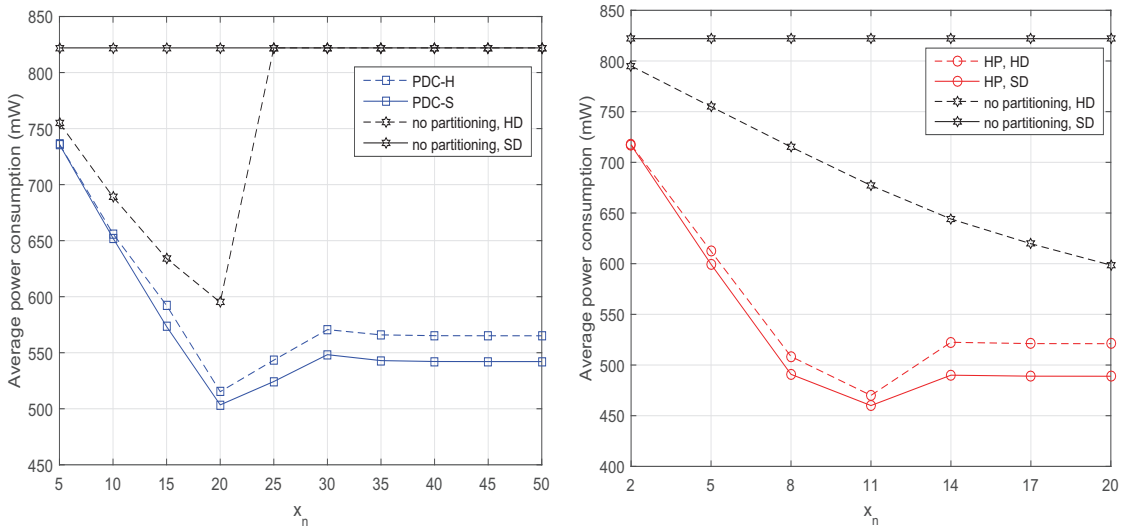
(a) 10 classes (b) 20 classes

Figure 4.3: Average power consumption versus θ (JSTCP)

much less than that using the delay-based ordering.

4.8.2 BSTCP

In this subsection, we examine BSTCP, i.e., the proposed task class partitioning algorithms with preallocated network resources. There are 10 task classes with their parameter values given in Table 4.2. Figure 4.4 shows the average MD power consumption versus x_n (same for all BSs) when $yf^{\mathrm{C}} = 25$ G CPU cycles/s. For comparison, we include the case of no class partitioning, i.e., all tasks can be offloaded, provided a channel is available upon the task arrival and the delay constraint can be satisfied (for SD only) (Chen *et al.* (2023)), which is an example of the state of the art that considers the computation offloading problem without class partitioning but with both the same hard and soft deadline definitions. Without class partitioning, the average MD power consumption is a constant for the SD case, since all tasks are executed locally based on the simulated parameter setting. For the HD case and without class partitioning, when x_n increases, the average power consumption



(a) PDC (b) HP

Figure 4.4: Average power consumption versus x_n

decreases first and then increases and finally becomes constant as x_n is sufficiently large.

By partitioning the task classes, both PDC and HP help significantly reduce the average MD power consumption for both SD and HD cases. For all the offloading solutions, provided offloading is possible, the mean MD power consumption decreases with the number of channels and then increases and finally keeps constant. This is because at first, increasing number of channels helps more tasks to offload, which reduces the MD power consumption; however too many offloaded tasks increases the queueing delay at the ES server, which increases the MD power consumption due to CLE (for HD) or results in more local executions (for SD). However, since the task arrival rates are fixed in the simulation, the maximum amount of traffic load at the ES stops increasing when x_n is sufficiently large, and therefore, the MD power consumption stops increasing with x_n.

Figure 4.5 shows the average MD power consumption versus y, where $x_n = 25$ for all n. Both PDC and HP help decrease the MD power consumption, and HP achieves

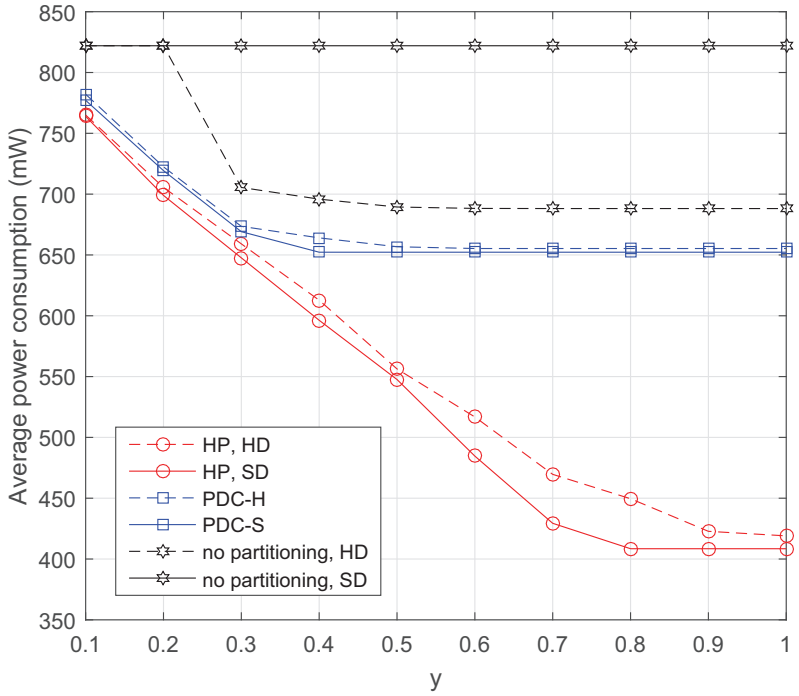


Figure 4.5: Average power consumption versus y

lower power consumption than PDC. When y is relatively small, the average MD power consumption using both PDC and HP decreases with y; and y is sufficiently large, the average MD power consumption becomes constant for both PDC and HP. This is because as y is sufficiently large, the queueing delay at the ES becomes almost zero, and further increasing y does not change the offloading performance. However, the y values at which the power consumption stops decreasing for PDC is much smaller than that for HP. This is an indication that HP allows more tasks to be offloaded, and therefore, achieves lower mean MD power consumption than PDC.

Figures 4.4 and 4.5 together indicate that for a given y value, there is an optimum number of channels to minimize the MD power consumption; and for given x_n, there is a minimum value of y that achieves the minimum MD power consumption. Next, we

examine the performance of joint task class partitioning and network resource allocations.

4.8.3 JSTCP

Table 4.5: Task class parameters, set 4

s_j (M bits)	3	8	10	16	46	51	59	82	90	99
q_j (M CPU cycles)	86	83	85	92	41	50	56	18	2	11
d_j (ms)	440	500	520	580	320	380	430	520	700	780

Table 4.6: Task class parameters, set 5

10 classes										
s_j (M bits)	2	4	6	7	8	10	32	35	39	75
q_j (M CPU cycles)	78	74	72	68	65	60	39	35	31	2
d_j (ms)	60	80	100	60	120	120	500	500	500	1000

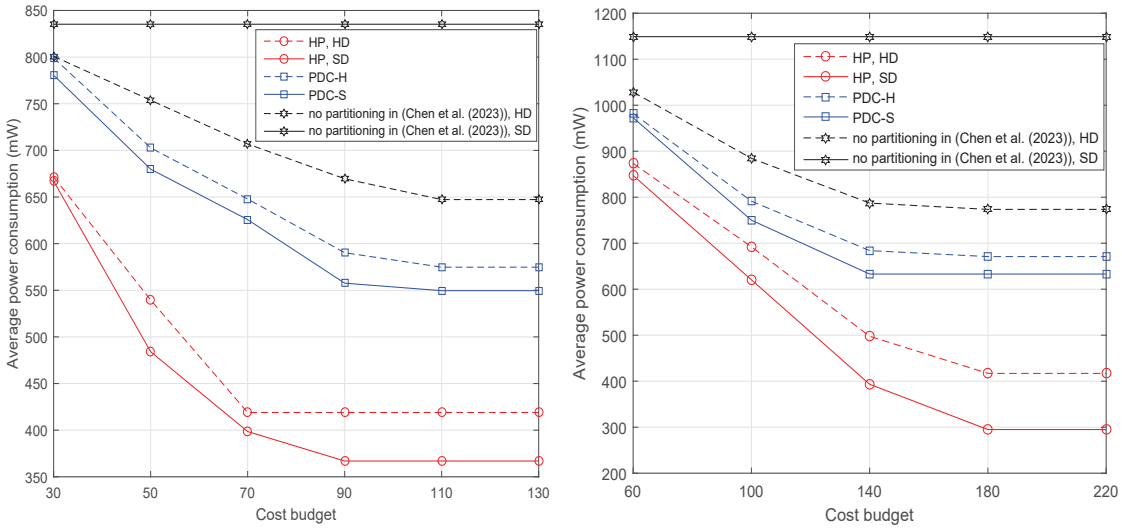
Table 4.7: Task class parameters, set 6

15 classes								
s_j (M bits)	2	3	6	8	10	31	33	35
q_j (M CPU cycles)	72	69	65	64	62	39	38	38
d_j (ms)	60	80	40	120	120	200	280	280
s_j (M bits)	37	40	62	63	65	76	77	
q_j (M CPU cycles)	35	33	10	10	8	2	1	
d_j (ms)	280	280	400	400	1000	1000	1000	

We consider two sets of task classes with the parameter values given in Tables 4.3 and 4.4, respectively. Default parameters used in this subsection are as follows. For the case with 10 task classes given in Tables 4.3, the total number of wireless channels in each BS is $M_n = 20$, the total ES capacity is $f^{\mathrm{C}} = 20$ G CPU cycles/s, and the resource cost budget is 100. For the case with 20 task classes given in Tables 4.4, the total number of

Table 4.8: Task class parameters, set 7

20 classes										
s_j (M bits)	2	4	6	30	31	32	33	33	34	35
q_j (M CPU cycles)	80	70	60	40	39	38	38	37	36	35
d_j (ms)	60	60	80	250	500	500	500	500	500	500
s_j (M bits)	35	36	37	38	38	39	40	60	70	80
q_j (M CPU cycles)	35	34	33	33	32	31	30	8	5	2
d_j (ms)	500	500	500	250	500	500	500	500	1000	1200

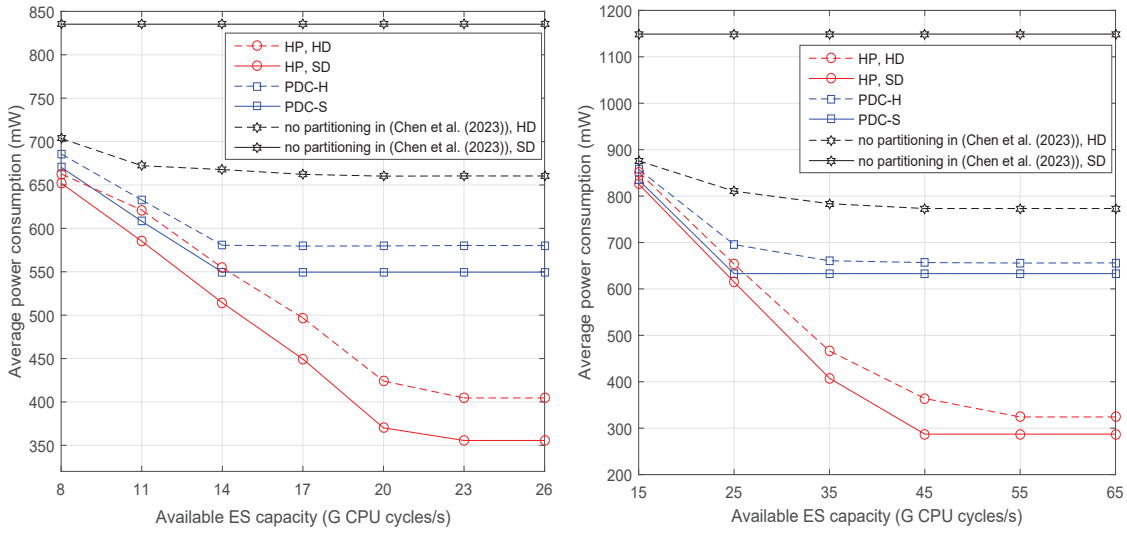


(a) 10 classes (b) 20 classes

Figure 4.6: Average power consumption versus cost budget (sets 2 and 3)

wireless channels in each BS is $M_n = 25$, the total ES capacity is $f^{\mathrm{C}} = 40$ G CPU cycles/s, and the resource cost budget is 220.

Figure 4.6 shows the average MD power consumption versus $B^{\max}$, which shows that the proposed joint class partitioning and resource allocation solutions can greatly reduce the average MD power consumption, compared to the no class partitioning achieved in (Chen *et al.* (2023)). When the cost budget is relatively small, the power reduction is small due to the limited amount of resources that limits the number of offloaded tasks. Note that



(a) 10 classes (b) 20 classes

Figure 4.7: Average power consumption versus ES capacity (sets 2 and 3)

all the tasks are executed locally for the SD case when there is no class partitioning. For all the other offloading cases, the cost budget increases, the average MD power consumption reduces, while using HP results in much lower average MD power consumption. When the cost budget is sufficiently high that results in negligible channel blocking probability and ES queueing delay, further increasing the cost budget does not help reduce the power consumption.

Figure 4.7 shows the average MD power consumption versus ES capacity, f^{C}. It is seen that when the ES capacity is relatively low, the difference of average MD power consumption between partitioning and no partitioning (Chen *et al.* (2023)) and between PDC and HP is small. The small amount of computing resource limits the number of offloaded tasks (without class partitioning) or task classes (with class partitioning), and as a result, most tasks are executed locally. As the ES capacity increases, the joint class partitioning and resource allocation solutions result in much lower average MD power consumption

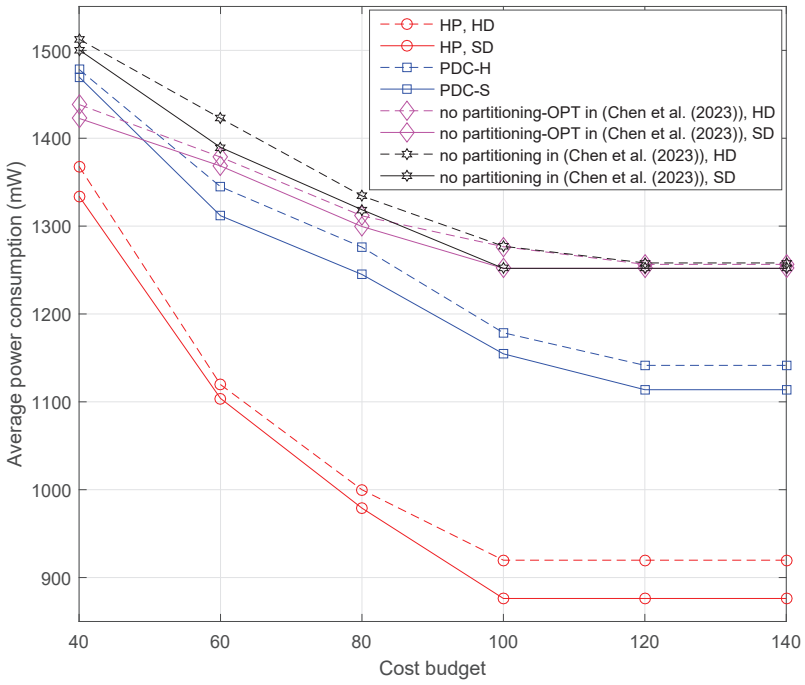


Figure 4.8: Average power consumption versus cost budget (set 4)

than the solutions without class partitioning. When the ES capacity is larger than a certain value, further increasing it will not reduce the MD power consumption since the offloading performance is limited by other factors such as cost budget. By comparing PDC and HP, it is seen that HP can take better advantage of the ES capacity to reduce the MD power consumption.

Next, we use a different set of the task class parameters as given in Table 4.5. We also add another set of comparisons with the optimum resource allocation obtained by exhaustive search for the no class partitioning case (Chen *et al.* (2023)). In this *no partitioning-OPT* case, the optimum resource allocation is obtained by exhaustive search without class partitioning. Figures 4.8 and 4.9 show the average MD power consumption. In Figure 4.8, the ES capacity is $f^{\mathrm{C}} = 30$ G CPU cycles/s; and in Figure 4.9, the cost budget is $B^{\max} = 100$. In both figures, the number of available wireless channels in each BS

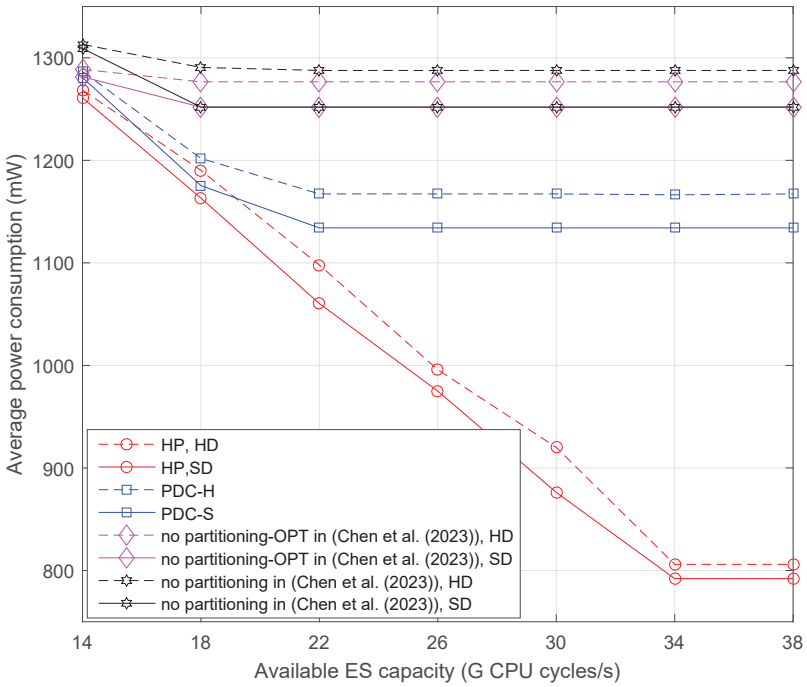


Figure 4.9: Average power consumption versus ES capacity (set 4)

is $M_n = 20$. Different from the results shown in Figures 4.6 and 4.7, the new task class parameters allow some tasks to be offloaded even for the SD case without class partitioning. However, the observations are consistent with before in terms of the effect of using the proposed joint class partitioning and resource allocation solutions on the average MD power consumption. In addition, our proposed algorithms can achieve significantly lower power consumption for the mobile devices than the optimal power consumption without class partitioning. This further demonstrates the importance of task class partitioning even when it is only static, and verifies the good performance of our proposed algorithms.

Finally, we consider the effect of parameter ξ in Algorithm 5. We consider three sets of task class parameters as given in Tables 4.6, 4.7, and 4.8. Other parameters used in the simulation include $x_n = 20$ for all BSs and ES capacity $yf^{\mathrm{C}} = 40$ G CPU cycles per second. The results are shown in Figure 4.10. For each set of the task classes, the average

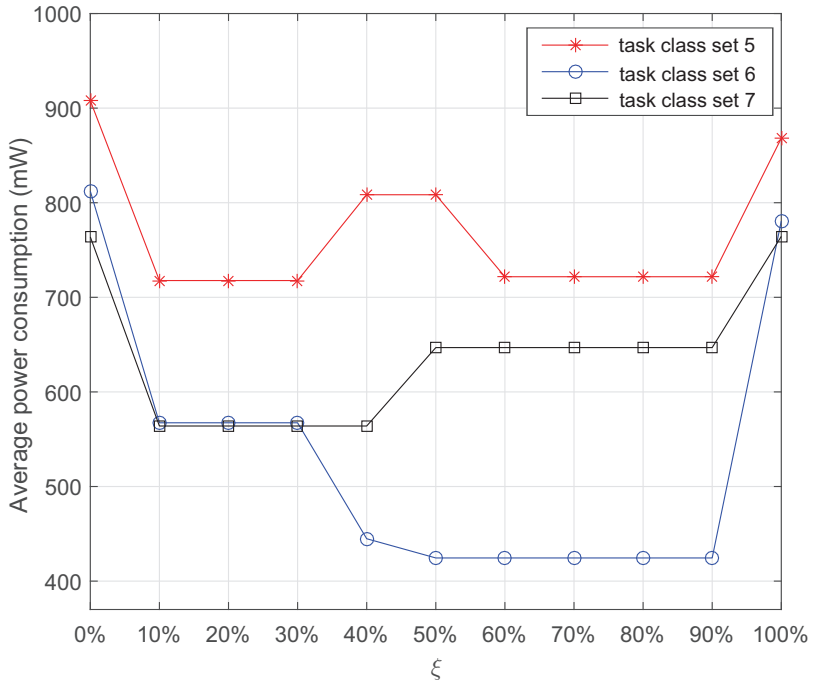


Figure 4.10: Average power consumption versus ξ in HP

MD power consumption is higher when ξ is 0 or 1. In another word, in order to reduce the average MD power consumption, the classes should not be ordered by $\Phi^{\mathrm{E}}_{n,j}$ only ($\xi = 0$) or $\Phi^{\mathrm{D}}_{n,j}$ only ($\xi = 1$). Although finding the optimum value of ξ that minimizes the average MD power consumption is not straightforward, we did notice that for each set of task class parameters, there is a relatively large range of ξ values during which the average MD power consumption is the minimum.

4.9 Summary

This chapter proposed using class partitioning to distinguish user applications when making MCO offloading decisions. The proposed static class partitioning solutions help to greatly reduce the MD power consumption while satisfying the application completion

time requirements. In the next chapter, we consider supporting digital twins as a special application in edge computing network. Unlike the scenarios in Chapters 3 and 4, all end devices periodically upload their feature-related data to a shared edge server for processing. We study the feature quality at the DTs by taking into consideration the shared data transmission channels and edge server computing resources.

Chapter 5

Digital Twin Model Selection for Feature Accuracy

5.1 Introduction

In this chapter, we consider the use of digital twins (DTs) in edge computing networks. The DT of a physical system (PS) is used to provide features on behalf of the PS, that can be accessed by other objects. A given feature is created using a data processing model that uses periodic updating from the PS to the DT as input. DT model selection accuracy problem is considered, where multiple PSs communicate with their DTs at an edge server (ES) that is located close to the PS's serving base station (BS). Each DT communicates with its associated PS, so that updates in the PS state can be incorporated into the features provided by the DT. We assume that this communication happens periodically and is referred to as *synchronization*. When a synchronization update occurs, the updating process involves processing at the ES so that features can be updated and made available to the requesting applications. The amount of processing needed, and the amount of data transferred when

synchronization occurs is a function of the level of accuracy associated with the model and the feature that is being provided. Since multiple DTs may be co-located at the same ES, the synchronization update processing must share the ES execution capacity, and this can limit the achievable accuracy of the provided features.

The objective of this chapter is to provide the best level of accuracy for a given set of feature requests by selecting an appropriate set of DT models. This is referred to as the *digital twin model selection problem* and is done using a max-min criterion, i.e., the objective is to maximize the minimum provided feature accuracy among the requested features, subject to the synchronization and ES execution constraints. We first formulate the problem as an integer program, which is reduced to an NP-complete feasibility problem. We then decompose it into multiple subproblems and show that each subproblem is a modified Knapsack problem. A polynomial-time approximation algorithm is proposed using a dynamic programming fully polynomial time approximation scheme (FPTAS) to solve it efficiently, by violating its constraints by a known factor. A generalization of the model selection problem is then given and an approximation algorithm using relaxation and dependent rounding is proposed to solve the problem efficiently with guaranteed small constraint violations. A variety of simulation results are presented that demonstrate the excellent performance of the proposed algorithms. The main contributions of this chapter are summarized below.

- A digital twin model selection problem is introduced. The objective is to make model selections that maximize the minimum provided accuracy among the requested features, subject to the PS/DT synchronization uploading and execution requirements. The problem is shown to be NP-complete.

- The model selection problem is decomposed into multiple subproblems, each consisting of a modified knapsack problem. A polynomial-time approximation algorithm

is proposed using dynamic programming (FPTAS) to solve it efficiently by violating its constraints by at most a given factor.

- A generalization of the model selection problem is given and an approximation algorithm is proposed using relaxation and dependent rounding to solve it efficiently with guaranteed small constraint violations.

- A variety of simulation results are presented that demonstrate the excellent performance of the proposed algorithms. It is verified that our proposed FPTAS algorithm is highly efficient when the system size is large. The approximate solution can be obtained efficiently by violating its constraints by no more than a given factor. We found that the proposed solution to the generalization achieves close-to-optimum feature accuracy and its constraint violation can be reduced by running additional rounds of the dependent rounding.

The remainder of the chapter is organized as follows. In Section 5.2, the prior work most related to this chapter is reviewed. The system model and problem formulation is then described in Section 5.3. Following this in Section 5.4, the polynomial-time approximation algorithm using dynamic programming (FPTAS) is proposed. In Section 5.5, a generalization of the model selection problem is given and an approximation algorithm using relaxation and dependent rounding is proposed. In Section 5.6 simulation results that demonstrate the proposed solutions are given. Finally, we present a summary of this chapter in Section 5.7.

5.2 Related Work

An increasing amount of work has considered the feature similarity/accuracy provided by a DT as an indicator of DT performance (Barricelli *et al.* (2019)). For example, the work in (Yiping *et al.* (2021)) proposes a method for DT-driven defect class recognition using a two-level deep learning architecture to detect and recognize novel classes. This includes a lifelong learning strategy to upgrade the recognition model. The proposed method has been shown to have a higher recognition accuracy for new defect classes compared to other pre-trained DT models. In order to realize DT-driven defect recognition, i.e., real-time defect recognition exploiting the real-time data collected by DT, the recognition model has to be updated (i.e., synchronized) in real-time. It can be seen from this work that different DT models created by different twinning methods can provide different accuracy levels for defect recognition. Reference (Lu *et al.* (2020)) gives an exploratory analysis of the geometric accuracy of digital twins generated for existing infrastructure using point clouds in the system operation and maintenance stage, especially for structural health monitoring purposes. A level of geometric accuracy (LOGA) is introduced as a measure of the twinning quality of the resulting digital twins. This work provides prospective twinning methods and deviation analysis for DT-driven structural health monitoring and indicates that the twinning method and LOGA strongly depend on the application and what kind of metadata is required. The work discussed above motivates the feature model selection problem considered in this chapter.

The work in (Paldino *et al.* (2022)) investigates a digital twin approach for improving estimation accuracy in dynamic thermal rating of transmission lines. Compared to existing physics-based standards, the estimation accuracy can be improved by adopting a

data-driven digital twin using machine learning for the physical sensor data and the conductor temperature. The DT is trained using a machine learning model on the measured data. After the training phase, the DT then acts as a complete virtual equivalent. The work in (Paldino *et al.* (2022)) also suggests another advantage of the DT, i.e., dimensionality reduction. The proposed approach considers which sensor measurements have a significant role through feature selection (Guyon and Elisseeff (2003)), which provides the system operator with meaningful information in terms of sensor importance. This is shown to reduce the amount of data collected without negatively impacting the model performance. This work also provides motivation for digital twin model selection based on feature accuracy that is considered in this chapter.

The work in (Dai and Zhang (2022)) integrates digital twins with vehicular edge computing (VEC) networks to implement adaptive network management and scheduling. It further proposes a DT empowered VEC offloading problem to minimize the total offloading latency, subject to deadline requirements and ES computation constraints. This is used to make offloading decisions and computational resource assignments. Reference (Lu *et al.* (2021)) proposes a wireless digital twin edge network framework and formulates the adaptive edge association problem considering the placement and migration of DTs. The work in (Zhou *et al.* (2022)) proposes a secure and latency-aware digital twin assisted resource scheduling algorithm for a 5G edge computing-empowered distribution grid. It develops a federated learning-based DT construction framework and formulates the DT-assisted resource scheduling problem to jointly optimize access scheduling, power control, and computational resource allocation. In the above work, the DTs of network elements are used for network management and resource allocations. Although the quality of the DTs directly affects the service quality to network users, the amount of network resources required to

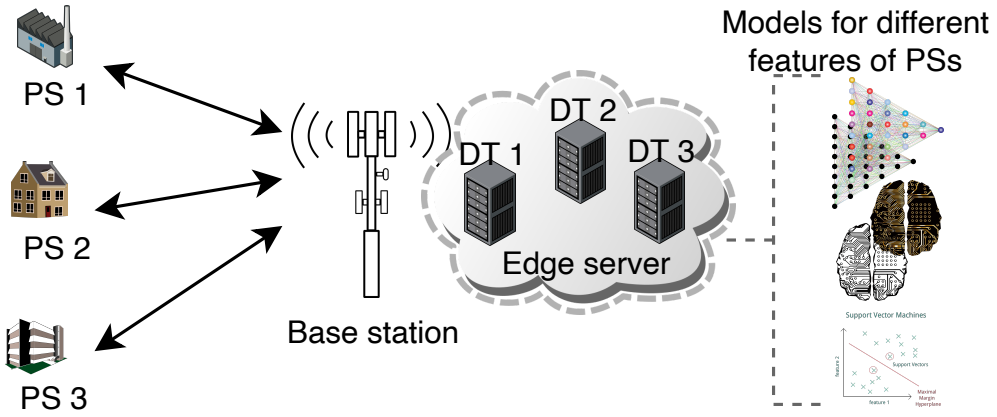


Figure 5.1: System Model

maintain the required quality of DTs has not been well studied. Unlike this work, in our work we study the best DT feature accuracy obtained using model selection, subject to resource constraints.

5.3 System Model and Problem Formulation

This chapter considers a wireless edge network where an ES hosts the DTs of multiple PSs, as shown in Figure 5.1. As is typical for DTs, each PS collects its own status data and periodically uploads the data to the DT, which then processes the data so that the different features implemented at the DT are synchronized with the PS state.

Let N be the total number of the PSs in the system and K be the total number of the features. Define $\beta_{i,k} \in \{0, 1\}$ as a binary variable representing the demand of PS i for feature k: if $\beta_{i,k} = 1$, PS i requires feature k, otherwise $\beta_{i,k} = 0$. Let $T_{i,k}$ be the data refreshing period for feature k of PS i. There are M_k models that can be used at the DT to obtain the state of feature k, each requiring different amounts of input data from the PS and computation load at the ES, and resulting in different accuracy to reflect the feature status

of the PS. Let $\mathcal{I}, \mathcal{K}, \mathcal{M}_k$ be the sets of PSs, features, and models for feature k, respectively.

Let $\Phi_{i,k,m}$ be the achieved accuracy for feature k of DT i when it is realized using model m, $s_{i,k,m}$ the corresponding amount of input data for the model, and $T_{i,k}$ the period that PS i uploads its feature-k related data to the ES. Let $x_{i,k,m} \in \{0, 1\}$ be the decision variable indicating whether the DT of PS i chooses to use model m to realize feature k. Note that $x_{i,k,m} = 0$ if $\beta_{i,k} = 0$. We assume that a DT can choose one and only one model to realize a given feature, i.e.,

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \ \ \forall i, k : \beta_{i,k} = 1. \tag{5.3.1}$$

When feature k is implemented by the i-th DT (i.e., $\beta_{i,k} = 1$), the achieved accuracy for this feature is

$$\Psi_{i,k} = \sum_{m=1}^{M_k} x_{i,k,m} \Phi_{i,k,m}. \tag{5.3.2}$$

Our objective is the maximization of the minimum accuracy for all features of all DTs, i.e., $\max_x \min_{i,k:\beta_{i,k}=1} \Psi_{i,k}$, while respecting computational and data transmission constraints, as well as the periodicity of DT updates.

More specifically, when the i-th PS uploads data to the ES, the data transmission rate R_i should be at least the amount required for the timely transmission of the input data needed by all the feature models, i.e.,

$$R_i \geq \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{s_{i,k,m}}{T_{i,k}} x_{i,k,m}. \tag{5.3.3}$$

Let F be the computation capacity of the ES in number of CPU cycles per second and $f_{i,k,m}$ denote the number of CPU cycles needed by the i-th DT in order to process feature

k using model m. The following capacity constraint must hold:

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{f_{i,k,m}}{T_{i,k}}x_{i,k,m} \leq F. \tag{5.3.4}$$

As a result of the discussion above, we obtain the following integer programming (IP) formulation of the problem where $\mathbf{x}_\beta = [x_{i,k,m},\ \forall i,k,m$ and $\beta_{i,k}=1]$:

$$\max_{\mathbf{x}_\beta} \min_{i,k:\beta_{i,k}=1} \Psi_{i,k} \text{ s.t.} \tag{IP}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k:\beta_{i,k}=1 \tag{5.3.5}$$

$$\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{s_{i,k,m}}{T_{i,k}}x_{i,k,m} \leq R_i, \qquad \forall i \tag{5.3.6}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{f_{i,k,m}}{T_{i,k}}x_{i,k,m} \leq F \tag{5.3.7}$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m \tag{5.3.8}$$

Equivalently, the problem can be formulated as follows:

$$\max_{\mathbf{x}_\beta, \tau} \tau \text{ s.t.} \qquad \text{(IP')}$$

$$\Psi_{i,k} \geq \tau, \qquad \forall i,k : \beta_{i,k} = 1$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k : \beta_{i,k} = 1$$

$$\sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{s_{i,k,m}}{T_{i,k}} x_{i,k,m} \leq R_i, \qquad \forall i$$

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{f_{i,k,m}}{T_{i,k}} x_{i,k,m} \leq F$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m$$

$$\tau \geq 0.$$

We observe that (5.3.2) implies that given i and k, the optimal $\Psi_{i,k}$ can take one of the $\Phi_{i,k,m}$ values, where $m = 1, 2, \ldots, M_k$. Therefore, the optimal τ can take one of at most $N \sum_k M_k$ values. Hence, in what follows we will assume that we have already 'guessed' (i.e., we try all possible values of) $\hat{\tau}$, which is the optimal τ. Given $\hat{\tau}$, we set $x_{i,k,m} := 0$ for all i, k, m that correspond to $\Phi_{i,k,m} < \hat{\tau}$, and reduce problem (IP') to the following simpler

feasibility problem:

$$\max_{\mathbf{x}_{\beta,\hat{\tau}}} 0 \text{ s.t.} \tag{FIP}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k : \beta_{i,k} = 1 \tag{5.3.9}$$

$$\sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{s_{i,k,m}}{R_i T_{i,k}} x_{i,k,m} \leq 1, \qquad \forall i \tag{5.3.10}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{f_{i,k,m}}{F T_{i,k}} x_{i,k,m} \leq 1 \tag{5.3.11}$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m \tag{5.3.12}$$

where $\mathbf{x}_{\beta,\hat{\tau}} = [x_{i,k,m},\ \forall i,k,m,\ \beta_{i,k} = 1 \text{ and } \Phi_{i,k,m} \geq \hat{\tau}]$.

Theorem 5. *Deciding whether problem* (FIP) *is feasible is* NP*-complete.*

Proof. It is straight-forward to see that (FIP) is in NP. To prove it is NP-complete, we reduce the KNAPSACK problem to it. Recall that the input to KNAPSACK is a set of n items, each item k with a value v_k and a weight w_k, a value V and a capacity W. KNAPSACK outputs 'yes' iff there is a subset of items of total value at least V and total weight at most W. Given an instance of KNAPSACK, we construct an instance of (FIP) as follows: We set $i = 1$ (i.e., we consider a single DT) and each feature corresponds to an item (i.e., $K = n$). Each feature has two models, say 1,2, where $x_{1,k,1} = 1$ corresponds to picking item k, while $x_{1,k,2} = 1$ corresponds to not picking it. We set $R_1 := W, F := \sum_{k=1}^{n} v_k - V$, and $T_{1,k} := 1, s_{1,k,1} := w_k, s_{1,k,2} := 0, f_{1,k,1} := 0, f_{1,k,2} := v_k\ \forall k$. Then it is clear that constraint (5.3.9) will either pick or not pick item k, constraint (5.3.10) requires that the picked item have total weight no more than W, and constraint (5.3.11) requires that the total value of the items *not* picked is at most $\sum_{k=1}^{n} v_k - V$. Hence, (FIP) is feasible iff the

KNAPSACK instance is a 'yes' one. □

Since there are no prospects of solving (FIP) in polynomial time, we will propose an approximation algorithm. Before presenting the proposed algorithm, we observe that (FIP) can be transformed into the following optimization problem

$$\min_{\mathbf{x}_{\beta,\hat{\tau}}} \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{f_{i,k,m}}{FT_{i,k}} x_{i,k,m} \text{ s.t.} \tag{OIP}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k : \beta_{i,k} = 1 \tag{5.3.13}$$

$$\sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{s_{i,k,m}}{R_i T_{i,k}} x_{i,k,m} \leq 1, \qquad \forall i \tag{5.3.14}$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m \tag{5.3.15}$$

and the feasibility of (FIP) is equivalent to achieving an objective value smaller than 1 in (OIP). In turn, (OIP) can be decomposed into N subproblems, with the ith one given as follows

$$\min_{\mathbf{x}_{\beta,\hat{\tau},i}} \sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{f_{i,k,m}}{FT_{i,k}} x_{i,k,m} \text{ s.t.} \tag{OIP$_i$}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall k : \beta_{i,k} = 1$$

$$\sum_{k=1}^{K}\sum_{m=1}^{M_k} \frac{s_{i,k,m}}{R_i T_{i,k}} x_{i,k,m} \leq 1,$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall k,m,$$

where $\mathbf{x}_{\beta,\hat{\tau},i} = [x_{i,k,m},\ \forall k,m,\ \beta_{i,k} = 1 \text{ and } \Phi_{i,k,m} \geq \hat{\tau}]$, and the feasibility of (FIP) is equivalent to checking that $\sum_{i=1}^{N} opt(\text{OIP}_i) \leq 1$.

To summarize, solving problem (IP) is reduced to the following steps:

1. Sort $\{\Phi_{i,k,m},\ \forall i,k,m\}$ in decreasing order as $\mathcal{O} = \{\Phi_1, \Phi_2, \Phi_3, \ldots\}$.
2. Starting with Φ_1, set $\hat{\tau}$ equal the current element of $\mathcal{O}$.
3. Set $x_{i,k,m} = 0$ if $\Phi_{i,k,m} < \hat{\tau}$ or $\beta_{i,k} = 0$.
4. Solve (OIP$_i$) for all i.
5. If $\sum_{i=1}^{N} opt(\text{OIP}_i) \leq 1$ then return $\hat{\tau}$, else (i.e., if one of the (OIP$_i$)'s is infeasible, or $\sum_{i=1}^{N} opt(\text{OIP}_i) > 1$) let $\hat{\tau}$ equal the next element in $\mathcal{O}$, or return **Infeasible** if the latter does not exist.

The general structure of our method is given in Algorithm 6.

Algorithm 6 General solution method

Input: $N, K, M_k, \Phi_{i,k,m}, s_{i,k,m}, f_{i,k,m}, T_{i,k}, F, R_i, \beta_{i,k}$
Output: model selection $\mathbf{x}$ and minimum feature accuracy $\hat{\tau}$ or Infeasible

Sort $\{\Phi_{i,k,m},\ \forall i,k,m\}$ in decreasing order as $\mathcal{O} = \{\Phi_1, \Phi_2, \Phi_3, \ldots\}$
for $\hat{\tau} = \Phi_1, \Phi_2, \Phi_3, \ldots$ **do**
    for all i, k, m **do**
        if $\Phi_{i,k,m} < \hat{\tau} \vee \beta_{i,k} = 0$ **then**
            $x_{i,k,m} = 0$
        end if
    end for
    Solve (OIP$_i$) for all i
    if (OIP$_i$) feasible $\forall i \wedge \sum_{i=1}^{N} opt(\text{OIP}_i) \leq 1$ **then**
        return solution $\mathbf{x}, \hat{\tau}$
    end if
end for
return Infeasible

▷ $\exists i$:(OIP$_i$) infeas., or $\sum_{i=1}^{N} opt(\text{OIP}_i) > 1$

5.4 An FPTAS for the problem

Since solving problem (IP) or (IP') is NP-complete (Theorem 5), we propose a polynomial-time approximation algorithm, that will guarantee a solution τ_s of (IP') with $\tau_s \geq \tau_{opt}$, where τ_{opt} is the optimal solution, but by violating constraint (5.3.7) by a factor of at most $(1+\varepsilon)$.

We show that subproblem (OIP$_i$) is a modified Knapsack problem, which can be approximately solved by a polynomial time algorithm using Dynamic Programming (DP). We transform the subproblem (OIP$_i$) into the following modified Knapsack problem:

TYPED KNAPSACK

Input: K types of items, with M_k items of type $k = 1, 2, \ldots, K$, weight $w_{k,m}$ and value $v_{k,m}$ for each item m of type k, number W.

Output: Pick *exactly* one item of each type, so that their total weight is at most W and their total value is minimized.

Note that TYPED KNAPSACK differs from the typical Knapsack problem in its types requirement and the minimization of its total value (as opposed to maximization). Let $M_{max} = \max_k M_k$.

5.4.1 FPTAS for TYPED KNAPSACK

We give a solution for TYPED KNAPSACK, based on a recursive definition of the minimum weight of items that can achieve a total value *exactly* equal to a given value V. We abuse notation a little and denote by $\mathcal{M}_k$ the set of items of type k.

Let $OPT(k, V)$ be the minimum weight of items from types $1, \ldots, k$ that yields value exactly V, when we pick exactly one item of each type. Then $OPT(k, V)$ is defined by recursion (5.4.1). In (5.4.1), cases 1 and 2 are the base cases when there is no item to

pick. In case 3, if all items of type k have values greater than V, then $OPT(k,V) = \infty$; otherwise, in case 4, $OPT(k,V)$ picks the item of type k achieving the minimum total weight. Note that if $OPT(k,V) = \infty$, then it is infeasible to achieve value exactly V using exactly one item from types $1, 2, \ldots, k$.

$$OPT(k,V) = \begin{cases} 0 & \text{if } k = 0, V = 0, \\ \infty & \text{if } k = 0, V > 0, \\ \infty & \text{if } k > 0, \min\limits_{m \in \mathcal{M}_k} v_{k,m} > V, \\ \min\limits_{m \in \mathcal{M}_k : v_{k,m} \leq V} \{w_{k,m} + OPT(k-1, V - v_{k,m})\} & \text{if } k > 0, \min\limits_{m \in \mathcal{M}_k} v_{k,m} \leq V, \end{cases} \tag{5.4.1}$$

A Dynamic Programming (DP) algorithm that solves TYPED KNAPSACK is given in Algorithm 7, where $V_{\max} = \sum_{k=1}^{K} \max_{m \in \mathcal{M}_k} v_{k,m}$.

Algorithm 7 DP algorithm for TYPED KNAPSACK problem

Input: $K, M_k, V_{\max}, w_{k,m}, v_{k,m}, W$
Output: minimum total value V^* or Infeasible

Find $OPT(k,V)$ for $0 \leq k \leq K$ and $0 \leq V \leq V_{\max}$ using DP by following (5.4.1)
$V^* = \infty$
for $V = 0 : V_{\max}$ **do**
    if $OPT(K,V) \leq W$ **then**
        $V^* = \min\{V^*, V\}$
    end if
end for
if $V^* = \infty$ **then**
    return Infeasible
else
    return V^*
end if

Unfortunately, the DP algorithm that solves TYPED KNAPSACK exactly is pseudo-polynomial, since it runs in time $O(KM_{\max}V_{\max})$ and $V_{\max}$ is pseudo-polynomial on the size of the input. However, we can apply the well-known value-scaling of the classical

Knapsack problem to derive a Fully Polynomial-Time Approximation Scheme (FPTAS) (Garey and Johnson (1979)) that produces a solution which exceeds the optimal total value by a factor of at most $(1+\varepsilon)$, for any constant $\varepsilon > 0$.

Algorithm 8 FPTAS for TYPED KNAPSACK problem

Input: $K, M_k, w_{k,m}, v_{k,m}, W, \varepsilon$
Output: minimum total value $\hat{V}^*$ or Infeasible
1: $\hat{v}_{k,m} = \lceil v_{k,m}/\theta \rceil$ for all k, m
2: $\hat{V}^* = \infty$
3: **for** $k' = 1 : K$ **do**
4: **for** $m' = 1 : M_{k'}$ **do**
5: $\hat{v}_{\max} = \hat{v}_{k',m'}$
6: $\hat{v}_{k,m} = \infty$ for all k, m with $\hat{v}_{k,m} > \hat{v}_{\max}$
7: Run Algorithm 7 using $\hat{v}_{k,m}$ for all k, m, which returns V^*
8: $\hat{V}^* = \min\{\hat{V}^*, V^*\}$
9: **end for**
10: **end for**
11: **if** $\hat{V}^* = \infty$ **then**
12: **return** Infeasible
13: **else**
14: **return** $\hat{V}^*$
15: **end if**

The FPTAS algorithm for solving the TYPED KNAPSACK problem is given in Algorithm 8, where $\theta = \varepsilon v_{\max}/K$ is defined as the scaling factor and $\varepsilon \in (0, 1]$ is the precision parameter. In line 1, the approximation algorithm rounds all item values up into integers lying in a finite range $[0, \lceil K/\varepsilon \rceil]$, then it runs the DP algorithm on the rounded instance, and finally returns the optimal solution of the rounded instance, which is the near optimal solution to the original instance. We assume that we have "guessed" the largest value $v_{\max}$ used by the optimal solution. The "guessing" is performed by exhaustive enumeration of all items m for all types k as the "maximum-valued" item (given in the double for-loop in the algorithm), and running Algorithm 7 for each choice (line 7), excluding all items of

value greater than the guessed maximum item value (line 6); then we keep the solution of minimum total item value. This exhaustive enumeration increases the running time of our algorithm by a factor of $O(\sum_k M_k) = O(KM_{\max})$. The obtained optimal solution to the rounded instance is the near optimal solution to the original instance.

Theorem 6. *For any given constant $\varepsilon > 0$, the algorithm returns a feasible solution (when one exists) of total value at most $(1+\varepsilon)OPT$, where OPT is the optimal solution value, and runs in polynomial time $O(K^4 M_{\max}^2/\varepsilon)$.*

Proof. Let O be an optimal solution to TYPED KNAPSACK when we use the original values, and $v_{\max}$ the maximum item value used in O (which we have guessed). Let $\hat{O}$ be the optimal solution obtained by the DP algorithm when run on the rounded-up values instance. Let $m_k, \hat{m}_k$ be the items of type k picked by $O, \hat{O}$, respectively. Let $|O|, |\hat{O}|$ be the values achieved by these solutions when the original item values are used, i.e., $|O| = \sum_k v_{m_k}$ and $|\hat{O}| = \sum_k v_{\hat{m}_k}$. Note that $|O| \geq v_{max}$. Then we have:

$$|\hat{O}| \leq \theta \sum_k \hat{v}_{\hat{m}_k} \tag{5.4.2}$$

$$\leq \theta \sum_k \hat{v}_{m_k} \tag{5.4.3}$$

$$\leq \sum_k v_{m_k} + K\theta \tag{5.4.4}$$

$$\leq |O| + \varepsilon v_{max} \leq (1+\varepsilon)|O|, \tag{5.4.5}$$

where (5.4.3) is due to the fact that $\hat{O}$ is an optimal solution for the rounded values. Although $v_{\max}$ is unknown, the algorithm exhaustively tries all possibilities for $v_{\max}$ and returns the solution that results in the smallest total value. Therefore, the returned solution also satisfies (5.4.5).

For the running time of the algorithm, first note that all rounded values are integers in the range $[0, \lceil K/\varepsilon \rceil]$. Therefore $0 \leq V_{\max} \leq K^2/\varepsilon$, and the running time of the DP algorithm for a single 'guess' of $v_{\max}$ is $O(K^3 M_{\max}/\varepsilon)$. Adding over all possible choices for the $v_{\max}$ item, we get a total running time of $O(K^4 M_{\max}^2/\varepsilon)$, which is polynomial on the size of the TYPED KNAPSACK input. □

5.4.2 Mapping of subproblem (OIP$_i$) to TYPED KNAPSACK

The mapping of subproblem (OIP$_i$) for a given PS-DT pair i to TYPED KNAPSACK is straight-forward: Each feature k defines a type, and the models in $\mathcal{M}_k$ are the items of type k. Item m of type k has weight $w_{k,m} = \frac{s_{i,k,m}}{R_i T_{i,k}}$ and value $v_{k,m} = \frac{f_{i,k,m}}{F T_{i,k}}$. Finally, we set $W = 1$. After the mapping, we run the FPTAS algorithm on the resulting TYPED KNAPSACK instance, and that provides us with an approximate solution of subproblem (OIP$_i$), according to Theorem 6. Algorithm 6 is modified, so that line 8 uses the approximation algorithm for (OIP$_i$), and the condition $\sum_{i=1}^{N} opt(\text{OIP}_i) \leq 1$ in line 9 is modified to $\sum_{i=1}^{N} opt(\text{OIP}_i) \leq 1 + \varepsilon$.

5.5 Generalization with model upper & lower bounds

In this section, we will give a generalization of the model selection problem for DTs formulated in section 5.3 and propose an approximation algorithm to solve it efficiently with guaranteed small violation of constraints.

We introduce limitation constraints on the utilization of the models for the demanded features of DTs, i.e., integer lower and upper bounds $L_m^{\min}, L_m^{\max}$ on the times a model m for

feature k can be used by different DTs, i.e.,

$$L_m^{\min} \leq \sum_{i=1}^{N} x_{i,k,m} \leq L_m^{\max}. \tag{5.5.1}$$

As special cases, when $L_m^{\min} = 0$, model m may not be selected by any features; and when $L_m^{\max} = \infty$, model m may be selected by all features. These extra constraints capture model accuracy requirements, such as "the high-accuracy model $m \in \mathcal{M}_k$ must be used by at least two sensors (i.e., $L_m^{\min} = 2$)", or "computationally-heavy model $m \in \mathcal{M}_k$ can be used by at most one camera (i.e., $L_m^{\max} = 1$)".

Introducing the new constraints to the original problem (IP) results in the following more general one:

$$\max_{\mathbf{x}_\beta} \min_{i,k:\beta_{i,k}=1} \Psi_{i,k} \text{ s.t.} \tag{GIP}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k : \beta_{i,k} = 1 \tag{5.5.2}$$

$$\sum_{i=1}^{N} x_{i,k,m} \geq L_m^{\min}, \qquad \forall k, \forall m \in \mathcal{M}_k \tag{5.5.3}$$

$$\sum_{i=1}^{N} x_{i,k,m} \leq L_m^{\max}, \qquad \forall k, \forall m \in \mathcal{M}_k \tag{5.5.4}$$

$$\sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{s_{i,k,m}}{T_{i,k}} x_{i,k,m} \leq R_i, \qquad \forall i \tag{5.5.5}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{f_{i,k,m}}{T_{i,k}} x_{i,k,m} \leq F \tag{5.5.6}$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m \tag{5.5.7}$$

Using the same procedure of problem transformation for (IP) in section 5.3, with given

$\hat{\tau}$, the problem can be transformed to the following feasibility problem:

$$\max_{\mathbf{x}_{\beta,\hat{\tau}}} 0 \text{ s.t.} \tag{GFIP}$$

$$\sum_{m=1}^{M_k} x_{i,k,m} = 1, \qquad \forall i,k : \beta_{i,k} = 1 \tag{5.5.8}$$

$$\sum_{i=1}^{N} x_{i,k,m} \geq L_m^{\min}, \qquad \forall k, \forall m \in \mathcal{M}_k \tag{5.5.9}$$

$$\sum_{i=1}^{N} x_{i,k,m} \leq L_m^{\max}, \qquad \forall k, \forall m \in \mathcal{M}_k \tag{5.5.10}$$

$$\sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{s_{i,k,m}}{R_i T_{i,k}} x_{i,k,m} \leq 1, \qquad \forall i \tag{5.5.11}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{f_{i,k,m}}{FT_{i,k}} x_{i,k,m} \leq 1 \tag{5.5.12}$$

$$x_{i,k,m} \in \{0,1\}, \qquad \forall i,k,m \tag{5.5.13}$$

Note that the added constraints are *hard*, i.e., we cannot violate them. In addition, the equivalence to TYPED KNAPSACK does not hold anymore, and, therefore, the approximation algorithm of section 5.4 does not work in this case. Hence, we will propose a different approximation algorithm in order to solve (GFIP) approximately in polynomial time, but with a worse approximation guarantee, which is not surprising since we are solving a more constrained problem.

5.5.1 An $O(\frac{\ln N}{\ln \ln N})$-approximation Algorithm

In this subsection, we present a polynomial-time approximation algorithm, that will guarantee a solution τ_s with $\tau_s \geq \tau_{opt}$, where τ_{opt} is the optimal solution, by respecting *exactly* constraints (5.5.8) (5.5.9) (5.5.10), but by violating constraints (5.5.11), (5.5.12) by a factor

of at most $O(\frac{\ln N}{\ln \ln N})$.

We note that by relaxing constraints (5.5.13) in (GFIP) to $x_{i,k,m} \geq 0,\ \forall i, k, m$, problem (GFIP) becomes a linear programming (LP) problem. This LP-relaxed problem is referred to as Relaxed-(GFIP). When lines 8-11 in Algorithm 6 are applied to Relaxed-(GFIP) instead of (OIP$_i$), the algorithm returns solution $\tau_f \geq \tau_{opt}$, unless the relaxed problem is infeasible (and therefore, the original problem is also infeasible).

Let x_f be the optimal fractional solution of Relaxed-(GFIP) achieving τ_f, computed in polynomial time by an LP solver. Our approximation algorithm will use the *dependent rounding* of (Gandhi *et al.* (2006)) to round the fractional components of x_f to values 0 or 1 with the required guarantees. The rounded solution cannot have a τ value smaller than τ_f, since Algorithm 6 guarantees that $x_{i,k,m} = 0$ when $\Phi_{i,k,m} < \tau_f$ (lines 3-7), and the rounding does not change this fact.

First, we give a high-level description of the dependent rounding procedure of (Gandhi *et al.* (2006)). Assume we are given a bipartite graph (V_1, V_2, E) with bipartition (V_1, V_2) and a value $x_{i,j} \in [0, 1]$ for each edge $(i, j) \in E$. Initialize $y_{i,j} = x_{i,j}$ for each $(i, j) \in E$. Values $y_{i,j}$ will be probabilistically modified in several (at most $|E|$) iterations such that $y_{i,j} \in \{0, 1\}$ at the end, at which point we will set $X_{i,j} := y_{i,j}$ for all $(i, j) \in E$, where $X_{i,j}$ are the (randomly) rounded final values for edges $(i, j) \in E$.

The iterations that modify y proceed as follows: We call an edge (i, j) *floating* if its value $y_{i,j}$ is not integral (i.e., $y_{i,j} \in (0, 1)$). Let $\tilde{E} \subseteq E$ be the current set of floating edges. If $\tilde{E} = \emptyset$, the process terminates by setting $X_{i,j} := y_{i,j}$ for all $(i, j) \in E$. Otherwise, find a simple cycle or maximal path S in the subgraph $(V_1, V_2, \tilde{E})$ in $O(|V_1| + |V_2|)$ time running depth-first-search (DFS). Partition the edge-set of S into two alternating matchings A and

B. We define

$$\alpha := \min\{\gamma > 0 : (\exists (i,j) \in A : y_{i,j} + \gamma = 1) \ \vee (\exists (i,j) \in B : y_{i,j} - \gamma = 0)\} \quad (5.5.14)$$

$$\beta := \min\{\gamma > 0 : (\exists (i,j) \in A : y_{i,j} - \gamma = 0) \ \vee (\exists (i,j) \in B : y_{i,j} + \gamma = 1)\}. \quad (5.5.15)$$

Then we execute the following randomized step:

- With probability $\beta/(\alpha+\beta)$ set $y_{i,j} := y_{i,j} + \alpha \ \forall (i,j) \in A, y_{i,j} := y_{i,j} - \alpha \ \forall (i,j) \in B$, and
- with probability $\alpha/(\alpha+\beta)$ set $y_{i,j} := y_{i,j} - \beta \ \forall (i,j) \in A, y_{i,j} := y_{i,j} + \beta \ \forall (i,j) \in B$.

In either case, at least one edge $(i,j) \in \tilde{E}$ will stop being floating, i.e., $y_{i,j} \in \{0,1\}$, and, therefore, after at most $|E|$ iterations or $O(|E|(|V_1| + |V_2|))$ time, all values y will become integral and the rounding process terminates. Algorithm 9 codifies the dependent rounding procedure. Note that dependent rounding is a randomized algorithm, and the final rounded solution X are random variables.

We apply this framework to the fractional solution of Relaxed-(GFIP), that can be obtained in polynomial time. The fractional solution corresponds to a bipartite graph $G = (V_1, V_2, E)$, with $V_1 = \{(i,k) : \beta_{i,k} = 1\}$, $V_2 = \{m \in \mathcal{M}_k, \ \forall k\}$, and $E = \{((i,k), m) : 0 < x_{i,k,m} < 1\}$.

Property (P2) of Dependent Rounding in (Gandhi *et al.* (2006)) states the following:

Theorem 7 (Degree-preservation)**.** *For a vertex $u \in V_1 \cup V_2$, let $d_u = \sum_{v:(u,v)\in E} x_{u,v}$ be the fractional degree of u. Then, if $X_{u,v}$ is the rounded value of $x_{u,v}$, $X_{u,v} \in \{\lfloor d_v \rfloor, \lceil d_v \rceil\}$.*

Theorem 7 ensures that the rounded solution will satisfy constraints (5.5.8), since these are constraints on the fractional degree $d_{i,k}$ of the vertices $(i,k) \in V_1$ and $d_{i,k} = 1$. Also,

Algorithm 9 Dependent rounding

Input: Bipartite graph (V_1, V_2, E), $x_{i,j} \in [0,1]$ for each edge $(i,j) \in E$
Output: $X_{i,j} \in \{0,1\}$ for each edge $(i,j) \in E$

$y_{i,j} = x_{i,j},\ \forall (i,j) \in E$; $\tilde{E} = E$
for all $(i,j) \in \tilde{E}$ **do**
  if $y_{i,j} \in \{0,1\}$ **then**
    $\tilde{E} := \tilde{E} \setminus \{(i,j)\}$
  end if
end for
while $\tilde{E} \neq \emptyset$ **do**
  Simple cycle or maximal path $S = DFS(V_1, V_2, \tilde{E})$; $S = A \cup B$ for alternating matchings A, B
  Obtain α and β from (5.5.14) and (5.5.15)
  With probability $\beta/(\alpha+\beta)$, $y_{i,j} := y_{i,j} + \alpha,\ \forall (i,j) \in A$, and $y_{i,j} := y_{i,j} - \alpha,\ \forall (i,j) \in B$
  With probability $\alpha/(\alpha+\beta)$, $y_{i,j} := y_{i,j} - \beta,\ \forall (i,j) \in A$, and $y_{i,j} := y_{i,j} + \beta,\ \forall (i,j) \in B$
  for all $(i,j) \in \tilde{E}$ **do**
    if $y_{i,j} \in \{0,1\}$ **then**
      $\tilde{E} := \tilde{E} \setminus \{(i,j)\}$
    end if
  end for
end while
$X_{i,j} = y_{i,j},\ \forall (i,j) \in E$
return $X_{i,j}$

it ensures the satisfaction of constraints (5.5.9) and (5.5.10), since these constraints imply $L_m^{min} \leq d_m \leq L_m^{max}$ and $L_m^{\min}, L_m^{\max}$ are integers.

It remains to study by how much constraints (5.5.11), (5.5.12) are violated. If $[x_{i,k,m}, \forall i,k,m]$ is the fractional solution of Relaxed-(GFIP), let

$$D^{\mathrm{LP}} := \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{f_{i,k,m}}{FT_{i,k}} x_{i,k,m},$$

$$R_i^{\mathrm{LP}} := \sum_{k=1}^{K} \sum_{m=1}^{M_k} \frac{s_{i,k,m}}{R_i T_{i,k}} x_{i,k,m},\ \forall i.$$

The application of part (i) of Theorem 3.1 of (Gandhi *et al.* (2006)) (proved by (Panconesi and Srinivasan (1997))) on the (randomly) rounded values $X_{i,k,m}$ implies the following Chernoff-type bounds:

Theorem 8.

$$\Pr\left[\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{f_{i,k,m}}{FT_{i,k}}X_{i,k,m} \geq (1+\varepsilon)\right] \leq \frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}, \tag{5.5.16}$$

$$\Pr\left[\exists i : \sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{s_{i,k,m}}{R_iT_{i,k}}X_{i,k,m} \geq (1+\varepsilon)\right] \leq \frac{Ne^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}. \tag{5.5.17}$$

Proof. Note that $0 \leq \frac{f_{i,k,m}}{FT_{i,k}}, \frac{s_{i,k,m}}{R_iT_{i,k}} \leq 1,\ \forall i,k,m$, since, otherwise, the optimal LP solution sets $x_{i,k,m} = 0$. Also, assuming the solution of (GFIP) does not result in infeasibility, i.e., $D^{\mathrm{LP}} \leq 1$ and $R_i^{\mathrm{LP}} \leq 1\ \forall i$ for some $\hat{\tau}$, the properties of dependent rounding imply that $E[\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{f_{i,k,m}}{FT_{i,k}}X_{i,k,m}] = D^{\mathrm{LP}} \leq 1$ and $E[\sum_{k=1}^{K}\sum_{m=1}^{M_k}\frac{s_{i,k,m}}{R_iT_{i,k}}X_{i,k,m}] = R_i^{\mathrm{LP}} \leq 1,\ \forall i$. Hence, given that properties (P1), (P3) of (Gandhi *et al.* (2006)) (extended by Theorem 4.4 of (Saha and Srinivasan (2018))) of Dependent Rounding hold, the conditions of Theorem 3.1 of (Gandhi *et al.* (2006)) are satisfied, and we have the inequalities of the theorem statement. □

As in Theorem 3.2 of (Gandhi *et al.* (2006)), setting $\varepsilon = O(\frac{\ln N}{\ln\ln N})$ guarantees a violation of (5.5.11), (5.5.12) by at most a factor $O(\frac{\ln N}{\ln\ln N})$ with probability larger than a constant. By repeating the experiment, i.e., dependent rounding, a constant number of times, the probability of success can be made bigger than any constant, e.g., 0.75.

As a result, with high probability the available resources need to be augmented only by small amounts in order to render our solutions feasible for the system. Also, note that although the upper bound of $O(\frac{\ln N}{\ln\ln N})$ for resource augmentation is guaranteed with high

probability, the randomized nature of our algorithm suggests that we can have better results if we run it a few times and keep the best solution.

5.6 Simulation Results

In this section, we present simulation results to demonstrate the performance of the proposed solutions for both the original problem and the extended problem. For comparison, we also obtain the optimum solutions using exhaustive search. We consider all the PSs are uniformly distributed in the circular coverage area of a BS, which has the maximum coverage of 150 m. The ES is located at the BS and hosts a DT for each of the PSs. The transmission rate between PS i and the BS is $R_i = w\log_2(1 + \frac{P_i^{\mathrm{T}} g_i}{\sigma^2})$, where w is the wireless channel bandwidth, P_i^{T} and g_i are the wireless transmission power and the link gain from PS i to the BS, respectively, and σ^2 denotes the noise power at the BS receiver input. Distance-based path loss is used for the link gains and the path loss exponent is 3. Default parameters used in the simulation are summarized in Table 5.1, where $U[a, b]$ denotes the uniform distribution between a and b. These parameter values are similar to those used in (Lu *et al.* (2021); Dai and Zhang (2022); Zhou *et al.* (2022)), and varied during the simulation. We intentionally use a wide range of parameter values based on the referenced ranges so that we can make conclusions that apply in general settings.

5.6.1 FPTAS for the Original Problem

In the first set of simulation, there are 6 PSs, the number of features (K) for each PS is varied during the simulation, and each feature has 6 models. The simulations are performed on a server with Ubuntu 18.04.6 LTS, Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz

Table 5.1: Default Parameters

Parameter	Value
$T_{i,k}$	$U[1, 5]$ s
$s_{i,k,m}$	$U[30, 150]$ M bits
$f_{i,k,m}$	$U[50, 500]$ M CPU cycles
$\Phi_{i,k,m}$	$U[0.7, 1]$
F	2 GHz
P_i^{T}	0.1 W
w	10 MHz
σ^2	-120 dBm

and 196 GB memory. Table 5.2 shows the running time of the proposed approximation solution and the optimum solution. As K increases, the size of the problem increases, and the running time of both solutions increases. For the optimum solution, the running time increases exponentially and quickly becomes prohibitively long, e.g., more than 24 days when $K = 10$. Although the running time of our proposed solution is sometimes longer than the optimum solution (but still short) when K is small (e.g., $K = 4$), it increases much slower and is significantly shorter than the running time of the optimum solution when $K = 10$. This demonstrates that our proposed solution is highly efficient when the system size is large.

Table 5.2: Comparison of running time

Number of features (K)	2	4	6	8	10
Proposed solution	0.05s	0.15s	2.18s	5.12s	13.34s
Optimum solution	0.06s	0.09s	32.5s	16236s	24 days

In the second set of simulation, we compare the minimum achieved accuracies using the proposed FPTAS solution and the optimum, respectively. Due to the long running time of the optimum solution, the comparison can only be performed for small size systems. In the simulation, there are 3 PSs, each PS requires 5 features, and each feature can be

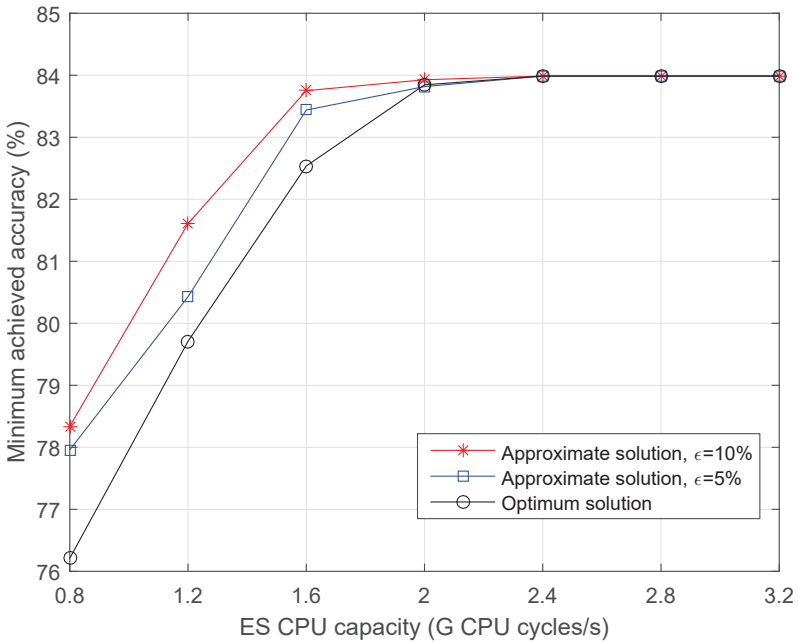


Figure 5.2: Minimum achieved accuracy versus ES computation capacity (original case)

implemented by using one of the 4 models (with different accuracy). The simulation results are averaged over 100 independent experiments, each of which is for one set of randomly generated PS locations and feature and model parameters.

Figure 5.2 shows the minimum achieved accuracy of features versus the ES CPU capacity F. First of all, the minimum achieved feature accuracy increases with the ES CPU capacity in general. The increase is more significant when F is relatively small and gradually becomes saturated as the wireless transmission rates eventually become the performance bottleneck. It is also seen that the minimum achieved accuracy using the proposed approximate solution can be higher than the optimum one, and the approximate solution achieves this at the price of violating constraint (5.3.7), i.e., ES CPU capacity constraint, by a factor of at most $(1 + \varepsilon)$. Besides, we can see that when the ES CPU capacity is relatively small, the minimum achieved feature accuracy with $\varepsilon = 10\%$ is higher than that

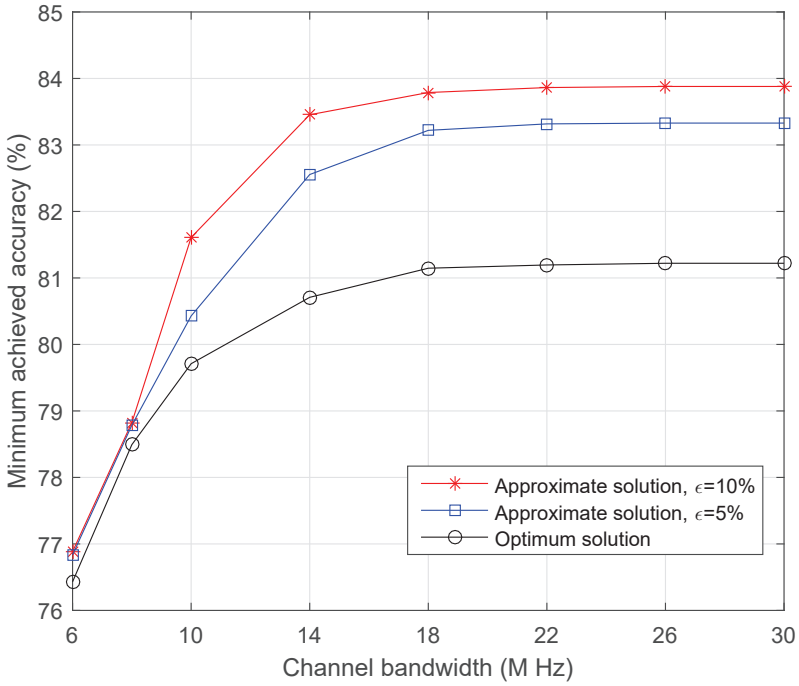


Figure 5.3: Minimum achieved accuracy versus wireless channel bandwidth (original case)

with $\varepsilon = 5\%$, since the approximate solution can achieve a better minimum accuracy by violating the ES capacity constraint more. However, as the ES CPU capacity becomes large enough, the minimum achieved accuracies using the proposed solution is the same as the optimum solution when the proposed solution does not violate the constraints.

Figure 5.3 shows the minimum achieved accuracy of features versus the wireless channel bandwidth w. The observations are similar as in Figure 5.2. The minimum achieved feature accuracy increases with the wireless channel bandwidth in general and gradually becomes a constant as the bandwidth is sufficiently large and the ES computation capacity eventually becomes the performance bottleneck. It can be seen that the minimum achieved accuracy using the proposed approximate solutions is always higher than the optimum one. It is because the ES computation capacity used in this simulation is relatively small, i.e.,

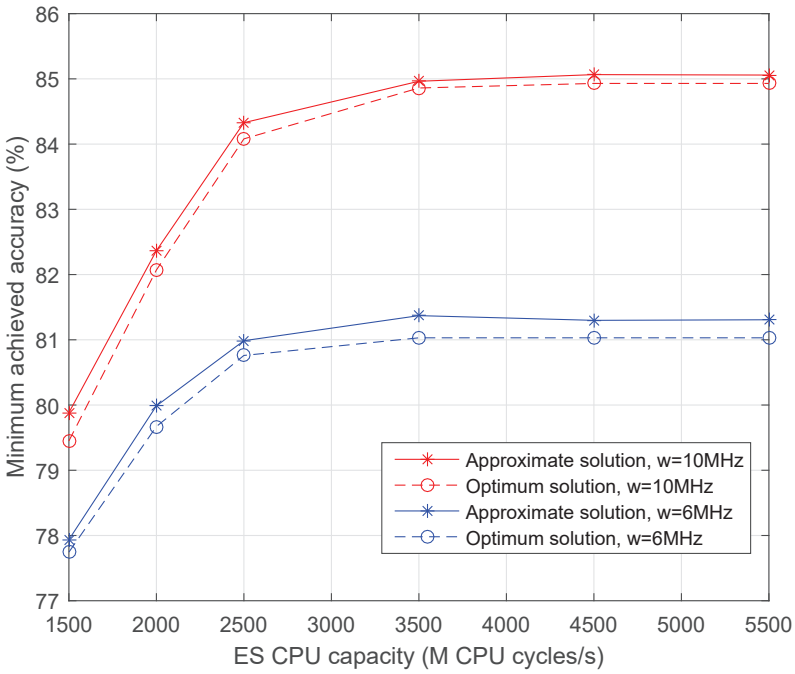


Figure 5.4: Minimum achieved accuracy versus ES computation capacity (generalized case)

1.2 G CPU cycles/s, the ES CPU capacity constraint is always violated by a factor of at most $(1+\varepsilon)$ in the approximate solutions.

5.6.2 Approximate Solution for the Generalization

In this subsection, we consider there are 5 PSs in the coverage of the BS, each PS requires 5 features, and each feature can be implemented by using one of the 4 models (with different accuracy). The input data $s_{i,k,m}$'s are randomly generated from $U[2, 200]$ M bits. The needed CPU cycles $f_{i,k,m}$'s are randomly generated from $U[5, 500]$ M CPU cycles. In this case, we assume the model with the highest accuracy of the first feature has to be used at least once. The models with the lowest accuracy of the first and the third features

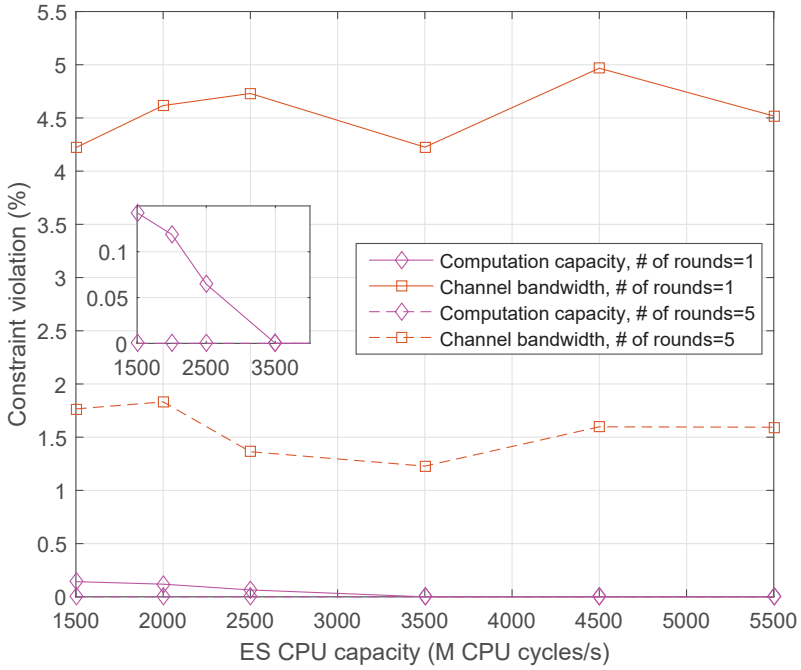


Figure 5.5: Constraint violation versus ES computation capacity

have to be used at most 4 times. The link gains include both the path loss and small-scale fading given as $g_i = 10^{-3}\rho_i{}^2 d_i^{-3}$, where d_i denotes the distance between PS i and the BS and ρ_i represents the additional channel small-scale fading which is assumed to be Rayleigh distributed (Ju and Zhang (2014); Chen *et al.* (2019a)). Thus, $\rho_i{}^2$ is an exponentially distributed random variable with unit mean. Note that a 30 dB average signal power attenuation is assumed at a reference distance of 1 m. The simulation results are averaged over 100 independent experiments, each of which is for one set of randomly generated PS locations and feature and model parameters.

Figure 5.4 shows the minimum achieved accuracy of features versus the ES CPU capacity F, where the proposed approximate method is applied by running one round of dependent rounding. The figure shows that the minimum achieved accuracy using the proposed approximate solution is very much close to the optimum. The gap is slightly

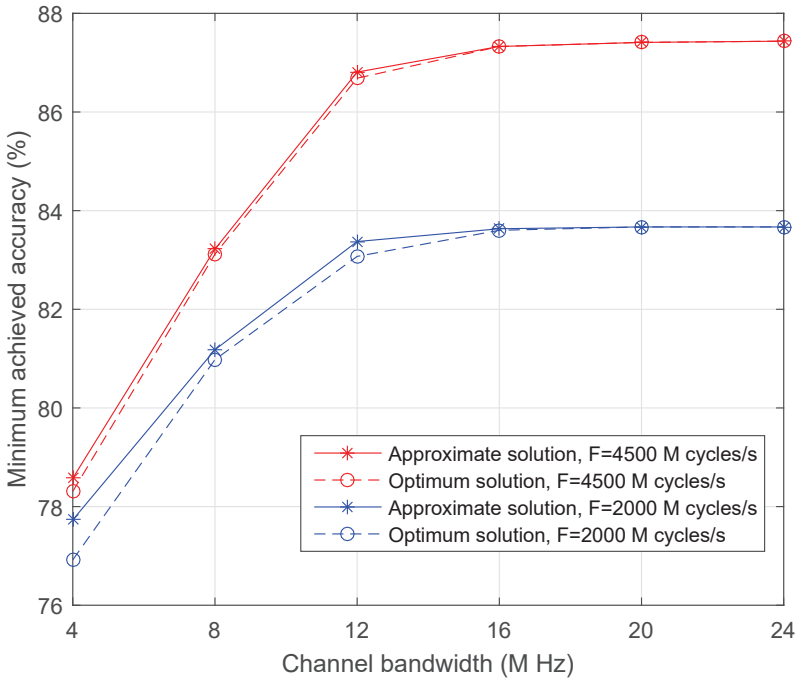


Figure 5.6: Minimum achieved accuracy versus wireless channel bandwidth (generalized case)

higher when the wireless channel bandwidth is smaller. Meanwhile, Figure 5.5 shows the corresponding constraint violation resulted from dependent rounding in the proposed approximate solution. Both the wireless channel bandwidth constraints (i.e., (5.5.5)) and the computation capacity constraint (i.e., (5.5.6)) are considered. For the computing constraint, the violation is calculated as percentage changes to the right-hand value after running the algorithm. For the bandwidth constraints, the percentage change to the right-hand value is calculated for each of the N constraints and then the maximum is taken. The figure shows that as F increases, the computation constraint violation decreases in general. When F is sufficiently large, there is no violation in the computation constraint. Note that the results of the dependent rounding are random, and therefore, the change of the constraint violation is not monotonic. The changes in bandwidth constraint violation are mainly due to

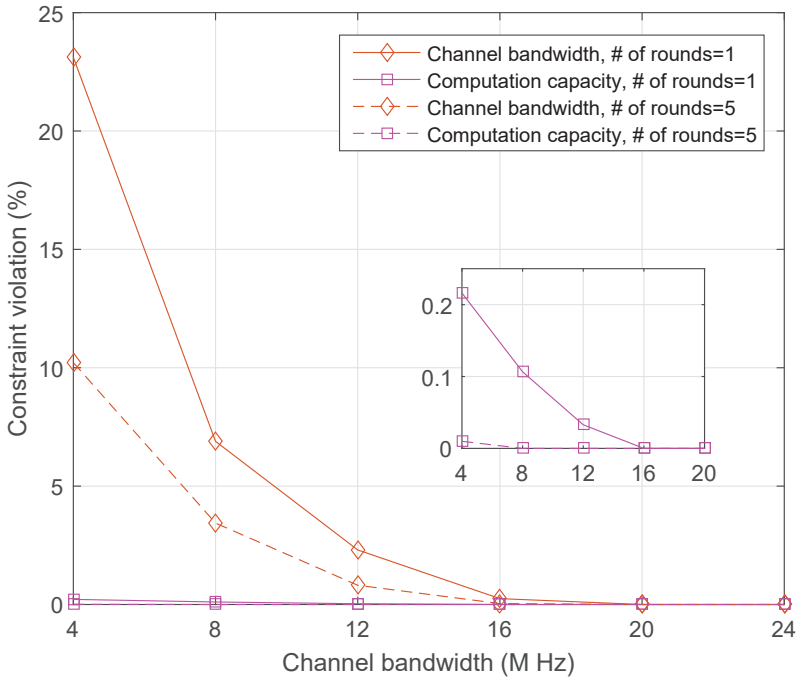


Figure 5.7: Constraint violation versus wireless channel bandwidth

the random effect of the dependent rounding. Figure 5.5 shows for both the bandwidth and computation constraints, running additional rounds of the dependent rounding helps reduce the constraint violation. Running additional rounds of the dependent rounding also helps bring the approximate solution closer to the optimum. However, the results after running five rounds of dependent rounding are not shown in Figure 5.4 because the approximate solution after running the dependent rounding for one round is already very close to the optimum.

Figure 5.6 shows the minimum achieved accuracy of features versus the wireless channel bandwidth w, when one round of dependent rounding is run in the proposed approximate solution. It is seen that the minimum achieved feature accuracy using the proposed approximate solution is very much close to the optimum, although the gap is slightly larger when the computation capacity is smaller. Figure 5.7 shows the constraint violation as the

wireless channel bandwidth changes. As the channel bandwidth increases, the bandwidth constraint violation decreases in general. When w is sufficiently large, there is no violation in the bandwidth constraint. Increasing the channel bandwidth, the computation constraint violation decreases in general, which is significantly smaller compared to the bandwidth constraint violation. However, increasing the number of rounds of running the dependent rounding helps reduce the violation in both types of constraints.

5.7 Summary

We have studied the model selection problem for optimizing feature accuracy of DTs in an edge computing network. A different feature model requires a different amount of data from the PS to the ES and a different amount of computation load at the ES. The objective of the feature model selection is to optimize the feature accuracy, subject to the periodic updates of the features and the network resource availability. Together with the previous two chapters, we have seen that joint communication and computation resource management results in some complicated optimization problems that require creative methods in order to solve them efficiently.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis investigated resource management for mobile computation offloading (MCO) and digital twins (DTs) in wireless edge networks under limited communication and computation resource availability. The work consisted of three parts.

In the first part of the thesis, we studied joint wireless network and task service allocation for mobile computation offloading. The objective is to minimize the average power consumption of mobile devices (MDs) while satisfying the delay constraints of tasks and the cost budget for network resources. The formulations presented included both soft and hard task completion time deadlines. The designs were formulated as mixed integer nonlinear programs (MINLPs) and approximate solutions were obtained by decomposing the formulations into convex subproblems. Simulation results were presented that characterize the performance of the system and show various tradeoffs between task deadline violation, average mobile device power consumption and the cost budget. Results were presented that

demonstrate the quality of the proposed solutions, which can achieve close-to-optimum performance over a wide range of system parameters. The optimum allocation was obtained by doing exhaustive search-based discrete event simulations for assigning the wireless channels from each base station (BS) and edge server (ES) capacity.

In the second part of the thesis, we introduced algorithms for static task class partitioning in MCO. The algorithms are given a set of task classes that must be partitioned into two sets, each containing task classes that are either executed locally by the MD or those classes that can contend for remote ES execution. The objective is to find the task class partition that gives the minimum mean mobile device power consumption subject to task completion time constraints. Algorithms for both soft and hard task completion time deadlines were presented. The algorithms consider two different variations of the problem. In the first, the wireless and computational resource capacities are given beforehand and the algorithms determine the task class partitioning. In the second variation, the resource capacities are not given, and the algorithms generate both capacity assignments and the partitioning subject to a resource cost budget constraint. Two basic algorithms were introduced, the first one uses latency based task class ordering, and the second orders the task classes in an hierarchical way by first grouping task classes with similar mean power consumption and then ordering the classes within the same group based on task completion time. A variety of simulation results were presented that demonstrate the excellent performance of the proposed class partitioning algorithms for both given and optimized network resource assignments.

In the third part of the thesis, considered DTs, that provide features that represent the real behavior of their associated physical systems (PSs). This is done using models that

yield differing levels of system accuracy. In this thesis, we considered the DT model selection problem where the DTs of multiple PSs are hosted at the same ES in a wireless edge network. The objective is to maximize the minimum achieved accuracy among the requested features by making appropriate model selections subject to communication channel and ES resource availability. This problem was first formulated as an NP-complete integer program. The thesis then reduced it to a feasibility problem and decomposed it into multiple subproblems that each consists of a modified Knapsack problem. A polynomial-time approximation algorithm was proposed using dynamic programming fully polynomial-time approximation scheme (FPTAS) to solve it efficiently by violating the constraint by at most a given factor. A generalization of the model selection problem for DTs was then introduced. A polynomial-time approximation algorithm using relaxation and dependent rounding was proposed that finds approximate problem solutions that provide an asymptotic bound on constraint violation. A variety of simulation results were presented that demonstrate the excellent performance of the proposed algorithms. It was verified that our proposed FPTAS is highly efficient when the system size is large. The approximate solution can be obtained by violating its constraints no more than a given factor. The proposed solution to the generalization achieved close-to-optimum feature accuracy and its violation of constraints can be reduced by running additional rounds of the dependent rounding.

6.2 Future directions

There are several possible directions for future extensions of our work. Unlike assigning aggregate channel resources from the network operator to each BS to support its associated mobile device population in the first part of the thesis, a possible future research direction is to allocate specific amount of wireless channels and computation resources of each BS

and ES to individual MDs based on the task requirements, which may improve the MCO performance.

An obvious future research direction is the study of dynamic task class partitioning and the development of online task class partitioning algorithms. In addition to ideas presented in this thesis, machine learning and deep learning techniques could also be used, in response to quickly changing environments.

Another future research direction is the combination of task class partitioning with partial task offloading. This adds the extra complication of dividing application tasks into sub-tasks and the assignment of sub-tasks to a set of sub-classes that will have to be partitioned according to task class partitioning, possibly with the requirement that sub-task precedence constraints are also satisfied.

For the DT work, a potential future research direction is the joint optimization of the DT model selection and network resource allocations. Instead of sharing the entire ES computation capacity and using fixed transmit power and equal channel bandwidth for PSs, allocating a specific amount of network resources to each pair of PS and DT could improve the DT performance, e.g., feature accuracy, based on its quality of service (QoS) requirements.

Bibliography

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., and Buyya, R. (2014). Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. *IEEE Communications Surveys Tutorials*, **16**(1), 337–368.

Alameddine, H. A., Sharafeddine, S., Sebbah, S., Ayoubi, S., and Assi, C. (2019). Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. *IEEE Journal on Selected Areas in Communications*, **37**(3), 668–682.

Apostolopoulos, P. A., Fragkos, G., Tsiropoulou, E. E., and Papavassiliou, S. (2023). Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty. *IEEE Transactions on Mobile Computing*, **22**(1), 175–190.

Ba, H., Heinzelman, W., Janssen, C.-A., and Shi, J. (2013). Mobile computing-a green computing resource. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4451–4456. IEEE.

Barricelli, B. R., Casiraghi, E., and Fogli, D. (2019). A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE access*, **7**, 167653–167671.

Berezner, S. A., Krzesinski, A. E., and Taylor, P. G. (1998). On the inverse of Erlang's function. *Journal of Applied Probability*, **35**(1), 246–252.

Blazek, T. and Mecklenbräuker, C. F. (2018). Measurement-based burst-error performance modeling for cooperative intelligent transport systems. *IEEE Transactions on Intelligent Transportation Systems*, (99), 1–10.

Boyd, S. P. and Vandenberghe, L. (2014). *Convex Optimization*. Cambridge University Press.

Burman, D. Y. (1981). Insensitivity in queueing systems. *Advances in Applied Probability*, **13**(4), 846–859.

Cao, X., Wang, F., Xu, J., Zhang, R., and Cui, S. (2019). Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet of Things Journal*, **6**(3), 4188–4200.

Cascone, L., Nappi, M., Narducci, F., and Passero, I. (2021). DTPAAL: Digital twinning pepper and ambient assisted living. *IEEE Transactions on Industrial Informatics*, **18**(2), 1397–1404.

Chakrabarty, S., Engels, D. W., and Wood, L. (2020). Consumer frameworks for smart environments. In *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 1–5. IEEE.

Chen, H., Zhao, D., Chen, Q., and Chai, R. (2020a). Joint computation offloading and radio resource allocations in small-cell wireless cellular networks. *IEEE Transactions on Green Communications and Networking*, **4**(3), 745–758.

Chen, H., Todd, T. D., Zhao, D., and Karakostas, G. (2023). Wireless and service allocation for mobile computation offloading with task deadlines. *arXiv preprint arXiv:2301.12088*.

Chen, J., Zhang, L., Liang, Y.-C., Kang, X., and Zhang, R. (2019a). Resource allocation for wireless-powered IoT networks with short packet communication. *IEEE Transactions on Wireless Communications*, **18**(2), 1447–1461.

Chen, M.-H., Liang, B., and Dong, M. (2018). Multi-user multi-task offloading and resource allocation in mobile cloud systems. *IEEE Transactions on Wireless Communications*, **17**(10), 6790–6805.

Chen, X., Liu, Z., Chen, Y., and Li, Z. (2019b). Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks. *IEEE Access*, **7**, 184172–184182.

Chen, X., Cai, Y., Shi, Q., Zhao, M., Champagne, B., and Hanzo, L. (2020b). Efficient resource allocation for relay-assisted computation offloading in mobile-edge computing. *IEEE Internet of Things Journal*, **7**(3), 2452–2468.

Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., and Shen, X. (2021). Energy efficient dynamic offloading in mobile edge computing for internet of things. *IEEE Transactions on Cloud Computing*, **9**(3), 1050–1060.

Dab, B., Aitsaadi, N., and Langar, R. (2019). Joint optimization of offloading and resource allocation scheme for mobile edge computing. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7.

Dai, Y. and Zhang, Y. (2022). Adaptive digital twin for vehicular edge computing and networks. *Journal of Communications and Information Networks*, **7**(1), 48–59.

Daley, D. J. and Servi, L. D. (1998). Idle and busy periods in stable M/M/k queues. *Journal of Applied Probability*, **35**(4), 950–962.

Deng, Y., Chen, Z., and Chen, X. (2020). Resource allocation for multi-user mobile-edge computing systems with delay constraints. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6.

Du, J., Zhao, L., Feng, J., and Chu, X. (2018). Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, **66**(4), 1594–1608.

El Marai, O., Taleb, T., and Song, J. (2020). Roads infrastructure digital twin: A step toward smarter cities realization. *IEEE Network*, **35**(2), 136–143.

Ericsson (2020). Ericsson mobility report 2020. `https://www.ericsson.com/en/mobility-report`.

Erol, T., Mendi, A. F., and Doğan, D. (2020). The digital twin revolution in healthcare. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–7. IEEE.

Fang, T., Yuan, F., Ao, L., and Chen, J. (2022). Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach. *IEEE Internet of Things Journal*, **9**(5), 3226–3237.

Feng, W., Tang, J., Zhao, N., Zhang, X., Wang, X., Wong, K.-K., and Chambers, J. A. (2021). Hybrid beamforming design and resource allocation for UAV-aided wireless-powered mobile edge computing networks with NOMA. *IEEE Journal on Selected Areas in Communications*, **39**(11), 3271–3286.

Franx, G. J. (2001). A simple solution for the M/D/1 waiting time distribution. *Operations Research Letters*, **29**(5), 221–229.

Gandhi, R., Khuller, S., Parthasarathy, S., and Srinivasan, A. (2006). Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, **53**(3), 324–360.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Geng, Y., Yang, Y., and Cao, G. (2018). Energy-efficient computation offloading for multicore-based mobile devices. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 46–54.

Gilbert, E. N. (1960). Capacity of a burst-noise channel. *The Bell System Technical Journal*, **39**(5), 1253–1265.

Glaessgen, E. and Stargel, D. (2012). The digital twin paradigm for future NASA and US air force vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA*, page 1818.

Gu, Z., Takahashi, R., and Fukazawa, Y. (2019). Real-time resources allocation framework for multi-task offloading in mobile cloud computing. In *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE.

Guo, H., Liu, J., and Zhang, J. (2018). Computation offloading for multi-access mobile edge computing in ultra-dense networks. *IEEE Communications Magazine*, **56**(8), 14–19.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, **3**(Mar), 1157–1182.

Han, Y., Niyato, D., Leung, C., Kim, D. I., Zhu, K., Feng, S., Shen, S. X., and Miao, C. (2022). A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse. *IEEE Internet of Things Journal*, pages 1–1.

Hekmati, A., Teymoori, P., Todd, T. D., Zhao, D., and Karakostas, G. (2020). Optimal mobile computation offloading with hard deadline constraints. *IEEE Transactions on Mobile Computing*, **19**(9), 2160–2173.

Huawei Inc. (2016). 5G network architecture - a high-level perspective. `https://www.huawei.com/en/technology-insights/industry-insights/outlook/mobile-broadband/insights-reports/5g-network-architecture`.

Jiang, K., Zhou, H., Chen, X., and Zhang, H. (2021a). Mobile edge computing for ultra-reliable and low-latency communications. *IEEE Communications Standards Magazine*, **5**(2), 68–75.

Jiang, Y., Yin, S., Li, K., Luo, H., and Kaynak, O. (2021b). Industrial applications of digital twins. *Philosophical Transactions of the Royal Society A*, **379**.

Ju, H. and Zhang, R. (2014). Throughput maximization in wireless powered communication networks. *IEEE Transactions on Wireless Communications*, **13**(1), 418–428.

Khintchine, A. Y. (1932). Mathematical theory of a stationary queue. *Matematicheskii Sbornik*, **39**(4), 73–84.

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

Kwon, Y., Yi, H., Kwon, D., Yang, S., Cho, Y., and Paek, Y. (2016). Precise execution offloading for applications with dynamic behavior in mobile cloud computing. *Pervasive and Mobile Computing*, **27**, 58–74.

Lee, A., Kim, J., and Jang, I. (2020). Movable dynamic data detection and visualization for digital twin city. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–2. IEEE.

Li, H., Xu, H., Zhou, C., Lü, X., and Han, Z. (2020). Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. *IEEE Transactions on Vehicular Technology*, **69**(9), 10214–10226.

Li, M., Gao, J., Zhou, C., Shen, X., and Zhuang, W. (2022a). Digital twin-driven computing resource management for vehicular networks. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 5735–5740.

Li, Q., Wang, S., Zhou, A., Ma, X., Yang, F., and Liu, A. X. (2022b). QoS driven task offloading with statistical guarantee in mobile edge computing. *IEEE Transactions on Mobile Computing*, **21**(1), 278–290.

Li, Y., Yang, B., Wu, H., Han, Q., Chen, C., and Guan, X. (2022c). Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach. *IEEE Internet of Things Journal*, **9**(17), 15969–15982.

Liu, D., Hafid, A., and Khoukhi, L. (2020). Multi-item auction based mechanism for mobile data offloading: A robust optimization approach. *IEEE Transactions on Vehicular Technology*, **69**(4), 4155–4168.

Liu, J., Ren, J., Zhang, Y., Peng, X., Zhang, Y., and Yang, Y. (2021). Efficient dependent task offloading for multiple applications in MEC-cloud system. *IEEE Transactions on Mobile Computing*, pages 1–1.

Lu, R., Rausch, C., Bolpagni, M., Brilakis, I., and Haas, C. T. (2020). Geometric accuracy of digital twins for structural health monitoring. In *Structural Integrity and Failure*. IntechOpen.

Lu, Y., Maharjan, S., and Zhang, Y. (2021). Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet of Things Journal*, **8**(22), 16219–16230.

Mach, P. and Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, **19**(3), 1628–1656.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, **19**(4), 2322–2358.

Masoudi, M. and Cavdar, C. (2021). Device vs edge computing for mobile services: Delay-aware decision making to minimize power consumption. *IEEE Transactions on Mobile Computing*, **20**(12), 3324–3337.

Messerli, E. J. (1972). B.S.T.J. brief: Proof of a convexity property of the erlang B formula. *The Bell System Technical Journal*, **51**(4), 951–953.

Mu, S., Zhong, Z., Zhao, D., and Ni, M. (2019). Joint job partitioning and collaborative computation offloading for internet of things. *IEEE Internet of Things Journal*, **6**(1), 1046–1059.

Mu, S., Zhong, Z., and Zhao, D. (2020). Energy-efficient and delay-fair mobile computation offloading. *IEEE Transactions on Vehicular Technology*, **69**(12), 15746–15759.

Muñoz, O., Pascual-Iserte, A., and Vidal, J. (2015). Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. *IEEE Transactions on Vehicular Technology*, **64**(10), 4738–4755.

Nath, S. and Wu, J. (2020). Dynamic computation offloading and resource allocation for multi-user mobile edge computing. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6.

Nath, S., Li, Y., Wu, J., and Fan, P. (2020). Multi-user multi-channel computation offloading and resource allocation for mobile edge computing. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6.

Noor, T. H., Zeadally, S., Alfazi, A., and Sheng, Q. Z. (2018). Mobile cloud computing: Challenges and future research directions. *Journal of Network and Computer Applications*, **115**, 70–85.

Paldino, G. M., De Caro, F., De Stefani, J., Vaccaro, A., Villacci, D., and Bontempi, G. (2022). A digital twin approach for improving estimation accuracy in dynamic thermal rating of transmission lines. *Energies*, **15**(6).

Panconesi, A. and Srinivasan, A. (1997). Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, **26**(2), 350–368.

Park, H., Jin, Y., Yoon, J., and Yi, Y. (2016). On the economic effects of user-oriented delayed wi-fi offloading. *IEEE Transactions on Wireless Communications*, **15**(4), 2684–2697.

Peng, J., Qiu, H., Cai, J., Xu, W., and Wang, J. (2021). D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC. *IEEE Transactions on Wireless Communications*, **20**(8), 4858–4873.

Pham, X.-Q., Nguyen, T.-D., Nguyen, V., and Huh, E.-N. (2021). Joint service caching and task offloading in multi-access edge computing: A QoE-based utility optimization approach. *IEEE Communications Letters*, **25**(3), 965–969.

Porambage, P., Okwuibe, J., Liyanage, M., Ylianttila, M., and Taleb, T. (2018). Survey on multi-access edge computing for internet of things realization. *IEEE Communications Surveys & Tutorials*, **20**(4), 2961–2991.

Qu, G., Zhou, J., Sheng, Z., Yu, H., and Ren, Y. (2021). Reliability-guaranteed admission control for mobile computation offloading under nakagami fading channel. *IEEE Wireless Communications Letters*, **10**(10), 2195–2199.

Raes, L., Michiels, P., Adolphi, T., Tampere, C., Dalianis, T., Mcaleer, S., and Kogut, P. (2021). DUET: a framework for building secure and trusted digital twins of smart cities. *IEEE Internet Computing*, pages 1–9.

Ren, J. and Xu, S. (2021). DDPG based computation offloading and resource allocation for MEC systems with energy harvesting. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5.

Ren, J., Yu, G., Cai, Y., and He, Y. (2018). Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, **17**(8), 5506–5519.

Ricci, A., Croatti, A., and Montagna, S. (2021). Pervasive and connected digital twins–a vision for digital health. *IEEE Internet Computing*.

Saha, B. and Srinivasan, A. (2018). A new approximation technique for resource-allocation problems. *Random Struct. Algorithms*, **52**(4), 680–715.

Shanbhag, D. N. and Tambouratzis, D. G. (1973). Erlang's formula and some results on the departure process for a loss system. *Journal of Applied Probability*, **10**(1), 233–240.

Shen, X., Gao, J., Wu, W., Li, M., Zhou, C., and Zhuang, W. (2021). Holistic network virtualization and pervasive network intelligence for 6G. *IEEE Communications Surveys & Tutorials*, **24**(1), 1–30.

Shen, X., Liao, W., and Yin, Q. (2022). A novel wireless resource management for the 6G-enabled high-density internet of things. *IEEE Wireless Communications*, **29**(1), 32–39.

Sheng, M., Wang, Y., Wang, X., and Li, J. (2020). Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server. *IEEE Transactions on Communications*, **68**(3), 1524–1537.

Shi, Y., Chen, S., and Xu, X. (2018). MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing. *IEEE Internet of Things Journal*, **5**(1), 164–174.

Stegmaier, V., Ghasemi, G., Jazdi, N., and Weyrich, M. (2022). An approach enabling accuracy-as-a-service for resistance-based sensors using intelligent digital twins. *Procedia CIRP*, **107**, 833–838. Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.

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 Communications Surveys & Tutorials*, **19**(3), 1657–1681.

Talkhestani, B. A. and Weyrich, M. (2020). Digital twin of manufacturing systems: a case study on increasing the efficiency of reconfiguration. *at-Automatisierungstechnik*, **68**(6), 435–444.

Tan, L., Kuang, Z., Zhao, L., and Liu, A. (2022). Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing. *IEEE Transactions on Wireless Communications*, **21**(3), 1960–1972.

Tang, F., Chen, X., Rodrigues, T. K., Zhao, M., and Kato, N. (2022). Survey on digital twin edge networks (DITEN) toward 6G. *IEEE Open Journal of the Communications Society*, **3**, 1360–1381.

Tran, T. X., Hajisami, A., Pandey, P., and Pompili, D. (2017). Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, **55**(4), 54–61.

Vaezi, M., Noroozi, K., Todd, T. D., Zhao, D., Karakostas, G., Wu, H., and Shen, X. (2022). Digital twins from a networking perspective. *IEEE Internet of Things Journal*, **9**(23), 23525–23544.

Vaezi, M., Noroozi, K., Todd, T. D., Zhao, D., and Karakostas, G. (2023). Digital twin placement for minimum application request delay with data age targets. *IEEE Internet of Things Journal*, pages 1–1.

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

Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., and Leung, K. K. (2015). Dynamic service migration in mobile edge-clouds. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9.

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

Wang, S., Wu, Y.-C., Xia, M., Wang, R., and Poor, H. V. (2020). Machine intelligence at the edge with learning centric power allocation. *IEEE Transactions on Wireless Communications*, **19**(11), 7293–7308.

Wang, Z., Zhong, Z., Zhao, D., and Ni, M. (2018). Vehicle-based cloudlet relaying for mobile computation offloading. *IEEE Transactions on Vehicular Technology*, **67**(11), 11181–11191.

Wang, Z., Zhao, D., Ni, M., Li, L., and Li, C. (2021). Collaborative mobile computation offloading to vehicle-based cloudlets. *IEEE Transactions on Vehicular Technology*, **70**(1), 768–781.

Wickramasinghe, N., Jayaraman, P. P., Forkan, A. R. M., Ulapane, N., Kaul, R., Vaughan, S., and Zelcer, J. (2021). A vision for leveraging the concept of digital twins to support the provision of personalized cancer care. *IEEE Internet Computing*, **26**(5), 17–24.

Wolff, R. W. (1982). Poisson arrivals see time averages. *Operations Research*, **30**(2), 223–414.

Wu, H. and Wolter, K. (2018). Stochastic analysis of delayed mobile offloading in heterogeneous networks. *IEEE Transactions on Mobile Computing*, **17**(2), 461–474.

Wu, H., Knottenbelt, W. J., and Wolter, K. (2019). An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, **30**(7), 1464–1480.

Wu, H., Sun, Y., and Wolter, K. (2020). Energy-efficient decision making for mobile cloud offloading. *IEEE Transactions on Cloud Computing*, **8**(2), 570–584.

Wu, Y., Ni, K., Zhang, C., Qian, L. P., and Tsang, D. H. K. (2018). NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation. *IEEE Transactions on Vehicular Technology*, **67**(12), 12244–12258.

Yang, X., Yu, X., Huang, H., and Zhu, H. (2019). Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems. *IEEE Access*, **7**, 117054–117062.

Yi, C., Huang, S., and Cai, J. (2021). Joint resource allocation for device-to-device communication assisted fog computing. *IEEE Transactions on Mobile Computing*, **20**(3), 1076–1091.

Yiping, G., Xinyu, L., and Gao, L. (2021). A deep lifelong learning method for digital twin-driven defect recognition with novel classes. *Journal of Computing and Information Science in Engineering*, **21**(3).

Yue, S., Ren, J., Qiao, N., Zhang, Y., Jiang, H., Zhang, Y., and Yang, Y. (2022). TODG: Distributed task offloading with delay guarantees for edge computing. *IEEE Transactions on Parallel and Distributed Systems*, **33**(7), 1650–1665.

Zaw, C. W., Tran, N. H., Han, Z., and Hong, C. S. (2021). Radio and computing resource allocation in co-located edge computing: A generalized nash equilibrium model. *IEEE Transactions on Mobile Computing*, pages 1–1.

Zeng, L., Wen, W., and Dong, C. (2022). QoS-aware task offloading with NOMA-based resource allocation for mobile edge computing. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1242–1247.

Zhang, J., Hu, X., Ning, Z., Ngai, E. C.-H., Zhou, L., Wei, J., Cheng, J., and Hu, B. (2017). Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet of Things Journal*, **5**(4), 2633–2645.

Zhang, J., Xia, W., Yan, F., and Shen, L. (2018). Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. *IEEE Access*, **6**, 19324–19337.

Zhou, S., Sun, Y., Jiang, Z., and Niu, Z. (2019). Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks. *IEEE Communications Magazine*, **57**(5), 49–55.

Zhou, Z., Jia, Z., Liao, H., Lu, W., Mumtaz, S., Guizani, M., and Tariq, M. (2022). Secure and latency-aware digital twin assisted resource scheduling for 5G edge computing-empowered distribution grids. *IEEE Transactions on Industrial Informatics*, **18**(7), 4933–4943.

Šlapak, E., Gazda, J., Guo, W., Maksymyuk, T., and Dohler, M. (2021). Cost-effective resource allocation for multitier mobile edge computing in 5G mobile networks. *IEEE Access*, **9**, 28658–28672.