OPTIMAL SERVER PLACEMENT, SERVICE DEPLOYMENT, AND RESOURCE ALLOCATION IN NEXT-GENERATION COMPUTER NETWORKS

by

Betül Aktel

B.S., Industrial Engineering, Boğaziçi University, 2013

M.S., Industrial Engineering, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to thank my supervisors Necati Aras and Kuban Altınel for their support and patience throughout my entire graduate study. I feel very happy to have the opportunity of benefiting from their vast knowledge in all areas. This research would have been impossible without their guidance and encouragement.

I also want to thank Cem Ersoy for his invaluable and constructive contributions throughout the years, which remarkably made my thesis to reach this quality. I am also very grateful to Refik Güllü, Metin Türkay, and Hande Küçükaydın for taking part in my thesis committee and for their valuable suggestions.

I would like to express my sincere thanks to all IE members for always having their support, understanding and friendship.

I would like to thank my parents and my brother for their support and love. I am thankful for trusting and encouraging me throughout my life.

I would like to thank my loving husband with all my heart. I am sure that things would be much harder without his support and love.

I also thank TUBITAK for their financial support during my graduate study.

This thesis is partially supported by Boğaziçi University Research Fund Grant Number 14522.

ABSTRACT

OPTIMAL SERVER PLACEMENT, SERVICE DEPLOYMENT, AND RESOURCE ALLOCATION IN NEXT-GENERATION COMPUTER NETWORKS

With the expansion of mobile devices and new trends in mobile communication technologies, there is an increasing demand for diversified services. To accommodate a large number of services on a common network, it becomes crucial for an operator to optimize resource allocation decisions to satisfy the service requirements in an economical way. In this thesis, the computation architecture design problem is considered first where server placement, service deployment, and task assignment decisions are optimized to maximize the revenue of the operator. The problem is modeled as a mixed-integer linear programming (MILP) formulation and a Lagrangian relaxation-based heuristic algorithm is proposed. Then, the concept of network slicing, which partitions a single physical network into multiple isolated slices, is examined. In the deterministic network slicing problem, the capacities of the computational resources are partitioned into slices each of which is customized for a particular service type. An MILP formulation is presented that takes the delay requirements of services into account. Additionally, two algorithms based on Benders decomposition are devised along with some valid inequalities and cut generation techniques. The problem definition is also extended to consider the stochastic behavior of the service requests. A two-stage stochastic integer programming model is constructed which is then converted into a large-scale MILP model by defining a set of scenarios for the random parameters. A similar decomposition approach is also applied to the stochastic network slicing problem. In our computational study on randomly generated test instances, the validity of our models is assessed and the effectiveness of the proposed solution approaches is demonstrated.

ÖZET

YENİ NESİL BİLGİSAYAR AĞLARINDA SUNUCU YERLEŞTİRME, SERVİS DAĞITIMI VE KAYNAK TAHSİSİ ENİYİLEMESİ

Taşınabilir aygıtların yaygınlaşması ve taşınabilir iletişim teknolojilerindeki yeni eğilimler nedeniyle, çeşitlendirilmiş hizmetlere yönelik artan bir istem bulunmaktadır. Çok sayıda hizmeti ortak bir ağ üzerinde barındırmak için, bir işletmenin hizmet gereksinimlerini ekonomik bir şekilde karşılamaya yönelik kaynak ayrılması kararlarını eniyilemesi çok önemli hale gelmiştir. Bu tezde ilk olarak, işletmen gelirini enbüyüklemek için sunucu yerleştirme, hizmet dağıtımı ve istem atama kararlarının eniyilendiği hesaplama mimarisi tasarım problemi tanıtılmaktadır. Problem, karma tamsayılı doğrusal programlama (KTDP) olarak modellenmiştir. Ek olarak, Lagrange gevşetmesi tabanlı sezgisel algoritma önerilmiştir. Ardından, tek bir fiziksel ağı birden çok ayrık dilime ayıran ağ dilimleme kavramı incelenmiştir. Deterministik ağ dilimleme problemi, hesaplama kaynaklarının sığalarının belirli bir hizmet tipi için özelleştirilmiş dilimlere bölünmesi olarak tanımlanır. Hizmetlerin gecikme gereksinimlerini dikkate alan bir KTDP formülasyonu sunulmaktadır. Ek olarak, bazı geçerli eşitsizlikler ve kesi oluşturma teknikleri ile birlikte Benders ayrıştırmasına dayalı iki algoritma geliştirilmiştir. Problem tanımı, hizmet istemlerinin rassal davranışını da dikkate alacak şekilde genişletilmiştir. İki aşamalı bir rassal tamsayılı programlama modeli oluşturulmuştur. Bu model, rassal parametreler için bir dizi senaryo tanımlanarak büyük ölçekli bir KTDP modeline dönüştürülmüştür. Benzer ayrıştırma yaklaşımı rassal ağ dilimleme problemine de uygulanmıştır. Rasgele oluşturulmuş sınam örnekleri üzerindeki hesaplamaya dayalı çalışmamızda, modellerimizin geçerliliği incelenmiş ve önerilen çözüm yaklaşımlarının etkinliği gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_l	Capital cost of a server at capacity level l
b	Total budget of the operator that can be spent for server placement decisions
B	Processing capacity of a single core
C_{qs}	Capacity per second of type q service deployed on potential server location s
d_{uq}	Total number of type q service requests per second generated at the end-user location u
E	Links
E_{us}	Links on the min-hop route between end-user location u and potential server location s
F_{qs}	The total load per second on server location s generated by type q service requests
F_s	The total load per second on server location s generated by all service requests
G_i	Subgradient vectors
g_l	The maximum number of service deployments on a server at capacity level l
h_q^{req}	The network load for a type q service request
h_q^{res}	The network load for a type q service response
K	Scenarios
L	Capacity levels of servers
M	SAA parameter that defines the number of independent samples to obtain the SAA problem in the first phase
m_q	Computation load for a type q service request
N	SAA parameter representing the sample size in the first phase

N'	SAA parameter representing the sample size in the second phase
N''	SAA parameter that defines the number of independent samples of second-stage problems used in the third phase
n	Number of instances for which the corresponding method can find a feasible solution within the allowed time limit
n_l	The processing capacity of server at capacity level l
o_q	Unit penalty cost incurred by unsatisfied service request of type q
p_k	Probability of scenario k
Q	The set of services
r_q	Unit revenue obtained by satisfying a service request of type q
S	Potential server locations
s	Standard deviation of the percent optimality gap
T	Step size of the Lagrangian heuristic
U	End-user locations
$U(a, b)$	Uniform distribution over interval (a, b)
V	Vertices
V_{us}	Vertices on the min-hop route between end-user location u and potential server location s
W_{iqs}	1 if i cores are allocated to service type q on server location s, and 0 otherwise
X_{sl}	1 if a server of level l is placed at server location s; 0 otherwise
Y_{qs}	1 if type q service is deployed on a server at location s; 0 otherwise
Z_{Lag}	The optimal objective value of LUBP
Z_{LB}	The best lower bound of the corresponding method
Z_{LP}	The objective value of the LP relaxation
Z_{LR}	The objective value of the Lagrangian relaxation solution

Z_{MILP}	The best objective value of the MILP model
Z_{UB}	The best upper bound of the corresponding method
Z_{uqs}	1 if type q service requests at end-user location u are ever assigned to a server at location s; 0 otherwise
α_q	The maximum allowed delay limit for service type q
β_{uqs}	Overall transmission delay for a type q service request generated at the end-user location u and assigned to a server at location s
γ	Scaling factor
δ_{uqs}	Lagrange multipliers
ϵ_s	Lagrange multipliers
ζ_{uqs}	Lagrange multipliers
η_u	Dual multipliers
θ_{uqs}	Fraction of type q service requests generated at end-user location u and assigned to a server at location s
κ_s	Dual multipliers
$\boldsymbol{\lambda}$	Lagrange multiplier vector
μ_{uq}	Dual multipliers
ν_{qs}	Dual multipliers
$\boldsymbol{\xi}$	Random data vector representing the set of stochastic parameters
ξ	Actual realizations of $\boldsymbol{\xi}$
π	Subgradient algorithm parameter
ρ	Maximum allowed utilization for computational resources
σ	Maximum allowed utilization for networking resources
τ_u	Dual multipliers
υ_s	Dual multipliers
ϕ_s	Dual multipliers
χ_l	Number of cores on a server at capacity level l

ψ_v	Networking capacity of vertex v
ω_e	Networking capacity of link e

LIST OF ACRONYMS/ABBREVIATIONS

5G	Fifth generation
AP	Access point
BD1	First decomposition approach for deterministic network slicing problem
BD2	Second decomposition approach for deterministic network slicing problem
CDN	Content delivery network
EB	Exabytes
ETSI	European telecommunications standards institute
IoT	Internet of things
IP	Integer programming
KTDP	Karma tamsayılı doğrusal programlama
LB Dev	The average percent deviation between Z_{MILP} and Z_{LB}
LB	Lower bound
LDP	Lagrangian dual program
LH	Lagrangian heuristic
LP Dev	The average percent deviation between Z_{MILP} and Z_{LP}
LP	Linear programming
LR Dev	The average percent deviation between Z_{MILP} and Z_{LR}
LR	Lagrangian relaxation
LUBP	Lagrangian upper bound program
MAN	Metropolitan area network
Mbits	Megabits
Mbps	Megabits per second
MI	Millions of instructions
MILP	Mixed-integer linear programming
MINLP	Mixed-integer nonlinear programming
NFV	Network function virtualization

QoE	Quality of experience
QoS	Quality of service
RAN	Radio access network
SAA	Sample average approximation
SDN	Software-defined networking
SLA	Service level agreement
SP_BD	Decomposition approach for stochastic network slicing problem
SP_BD_int	Decomposition approach for stochastic network slicing problem with integer capacity levels
SP_MILP	Deterministic equivalent MILP model for stochastic network slicing problem
SP_MILP_int	Deterministic equivalent MILP model for stochastic network slicing problem with integer capacity levels
UB Dev	The average percent deviation between Z_{MILP} and Z_{UB}
VM	Virtual machine
VNF	Virtual network function
WAN	Wide area network
WLAN	Wireless local area network

1. INTRODUCTION

The mobile communication industry has recently been subject to rapid evolution and innovations on mobile networking technologies. During the last decade, we have witnessed an exponential growth in mobile computing. The use of smartphones have spread to all aspects of our daily lives. In addition to smartphones, sensors and wearable gadgets such as smart glasses, watches, and bracelets are now commercially available in the market. This change has considerably enhanced our day-to-day experiences and dramatically transformed a wide-range of industries.

With recent innovations and emerging technologies in mobile communication industry, the number of smart devices and gadgets requiring connectivity to the internet has increased significantly. According to Cisco, global data traffic is expected to reach 396 exabytes (EB) per month by 2023. The number of connected devices, such as sensors and wearable gadgets will increase to 29.3 billion in 2023 and 45% of those will be mobile. Nearly 66% of the global population will have an access to the internet and there will be 5.3 billion active internet users worldwide [1]. According to world-leading operators, a growth on a tremendous scale is expected in the coming decade, i.e. 1,000 times higher traffic volume and 100 times more throughput [2].

This growth in mobile data traffic may push the network bandwidth requirements to the limit, cause a burden on the underlying network infrastructure, which can result in increased load and congestion, and significant latency throughout the network. Hence, the owner of the physical telecommunication infrastructure, called the operator, may face increased capital and operational expenses. Because of this massive growth in data volume, it is vital for the operators to optimize their resources in an economically sustainable way.

Meanwhile, the emerging fifth generation (5G) networks are designed to connect virtually everyone and everything together including machines, objects, and devices.

They are expected to provide higher performance and improved efficiency by delivering higher data speeds, massive machine-type communications, improved network capacity, ultra-low latency, higher reliability, and increased availability. With these improvements it is able to create never-before-seen opportunities for people and businesses [3]. According to Qualcomm, the overall global economic output of 5G is estimated to reach $13.1 trillion and it will support up to 22.8 million jobs worldwide through a thriving value chain by 2035 [4].

Recently, the term Internet of Things (IoT) has received increasing attention from researchers and practitioners. It refers to the physical objects, such as home appliances, machines, or transportation vehicles equipped with sensors, processing ability, software, and other technologies. These objects are usually interconnected, can exchange data with other devices over the internet, and act based on the information they get from one another without any human assistance. This brings a new form of communication between people and devices. It enables smarter environments and real-time control over devices. High speed, low latency, and massive communication capacity of 5G networks enable the IoT on a truly massive scale [5, 6].

The higher connectivity provided by 5G networks along with IoT devices are expected to allow more efficient businesses and provide consumers access to more information in a faster way than ever before. Hence, they are envisioned to open up new innovation opportunities and offer a wide range of services with a diversified set of performance and service requirements from a diverse set of verticals to achieve substantial improvement on the quality of service (QoS). These verticals include manufacturing, automotive, healthcare, energy, and entertainment. In addition, mobile gadgets are now capable of providing novel user-centric services that can enhance the quality of life such as autonomous driving, drone-based delivery systems, smart cities, smart transportation, cloud-connected traffic control, smart grid, augmented reality, and mobile gaming [7].

In contrast to the traditional mobile broadband services, the novel services may have differentiated and potentially conflicting QoS requirements in terms of reliability, latency, mobility, data traffic volume, power efficiency, security, and privacy protection. For example, in a scenario of autonomous cars, latency and reliability are very critical and real-time interaction has to be ensured. On the other hand, IoT applications may require massive connectivity among lots of devices [8]. Applications such as face recognition, augmented reality, and smart home generate high amounts of data, require massive connectivity among devices, ultra-low end-to-end latency, real-time communication, high reliability, and more energy efficient networking [2, 9]. These services are expected to be active concurrently and share the same underlying physical network. In addition, most of these services require real-time interaction with the end-users. This requires a high degree of flexibility and scalability of the network to support different use-cases as each of them has its own set of performance and computational requirements defined in their service level agreements (SLAs).

The proliferation of novel services have led to the development of a new network design concept. The growth in mobile data traffic and stringent requirements of diversified services necessitates the optimization of network operations and resource utilization to maintain quality of experience (QoE) and generate revenue for the operators [8]. With billions of smart devices demanding a wide range of services in a heterogeneous network environment, users' expectations are getting more diverse and they differ in usage patterns of data-intensive mobile applications. Moreover, networks become increasingly dynamic as they support ever-growing demand for diversified services with stringent QoS requirements. The demand for these services can be time varying and may change over time due to user mobility. Hence, the existing networks that work on "one-size-fits-all" basis are not economically feasible to handle the challenges of the deployment of services with a broad range of requirements as a result of increased cost of energy and higher capital investment requirement [10]. Management of a next-generation network requires having a scalable and adaptive models suitable for large-scale problem instances and dynamic fluctuations. The network needs to be well designed to meet the requirements of differentiated services. To provide cost

and energy-efficient solutions, the network architecture and technologies need to be revisited, a flexible and scalable mobile network has to be designed. The decisions on the architecture design, network deployment, and network management are highly challenging for the operators [11, 12].

An operator who wants to invest in the next-generation mobile networks has to design the computing architecture to optimize its operations in an efficient manner to ensure the satisfaction of the end-to-end delay restrictions of service instances defined in their SLAs. In this thesis, we consider a computing architecture where discrete capacity levels of servers can be synchronized with numerous services having diversified characteristics as shown in Figure 1. We assume that the networking resources are already deployed and functional, and their capacities are known. In this achitecture, various capacity levels of servers can co-exist in the same network to meet the user expectations. In this way, the service requests requiring very low end-to-end delay can be processed in the vicinity of the end-users while resource-hungry, but latency-tolerant services can be offloaded to a remote and powerful server whenever needed [13].

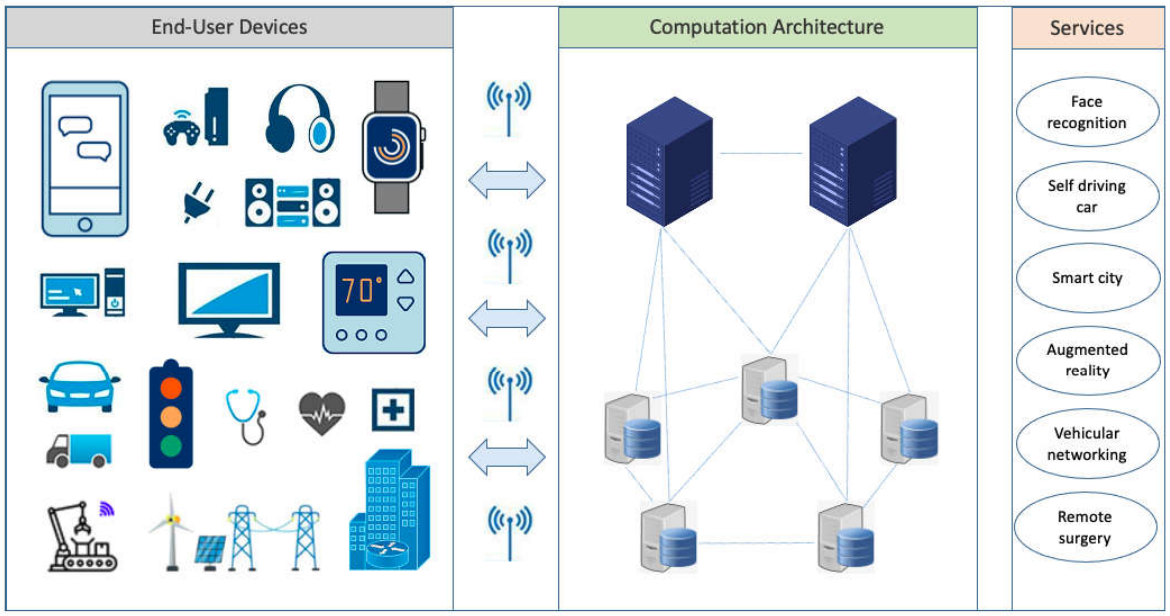

Figure 1.1. An example network architecture with diversified services.

At the initial investment phase of designing a next-generation mobile computing architecture, the following decisions are among key factors having an effect on the profitability of an operator:

- Server placement: To identify the locations and capacity levels of the computational resources among a given set of potential sites
- Service deployment: To decide the set of services that will be deployed on each server
- Resource allocation: To determine the partitioning of the capacities of computational resources among different service types
- Task offloading: To identify which server will execute the service request generated at an end-user location.

These decisions are interrelated, and thus should be considered in an integrated manner. In this thesis, we study three problem variants combining these decisions under different scenarios; namely the computation architecture design problem, the deterministic network slicing problem, and the stochastic network slicing problem. The main goal of the thesis is to create a decision making tool and optimize the network design process for an operator by taking service requirements and investment restrictions into account.

In the computation architecture design problem, the objective is to maximize the revenue of the operator that will be collected through successful task assignments. The main focus is to determine the server placement decisions as they are the most crucial components of the planning activities for the efficient design and operation of the network infrastructure that can prevent potential revenue loss in the future due to unsatisfied service requests or under-utilization of servers. Although these decisions are at a strategic level, they have a long-term impact on the profitability of the operator since they affect the tactical and operational level decisions, namely service deployment and task assignment operations. This implies that the problem has to be considered in an integrated manner. With a limited initial investment budget, the operator should op-

timize the server placement and service deployment decisions so that task assignments can be handled to meet the service requirements and user expectations.

While designing the computation architecture, the actual demand for service requests is usually unknown at the initial investment phase. In this problem, we do not assume any prior distribution over service demand. Even though the service offload requests arrive in a stochastic manner, the operator can maximize its expected revenue in the future by considering the expected demand. Hence, we use a static approach based on steady-state demand rates rather than considering transient behavior. To optimize server placement, service deployment, and task assignment decisions in a comprehensive way, an MILP model is developed in which numerous service types having diversified characteristics can be accommodated. Additionally, this model is capable of satisfying additional QoS specifications such as the maximum allowed delay limit. In addition to the MILP formulation, a Lagrangian relaxation-based heuristic algorithm is designed to find near-optimal solutions for large instances where the MILP model becomes insufficient to provide a feasible solution. Then, we extend the problem and assume that service demand can fluctuate over time and networking and computational resource requirements of different services can be random. To formulate the extended problem, we propose a two-stage stochastic integer programming model to optimize the server placement, service deployment, and task assignment decisions given an initial investment budget restriction. We apply the Sample Average Approximation (SAA) method described by Kleywegt et al. [14] to find good-quality solutions for different network topologies.

In the network slicing problem, service instances do not share the common capacity of the computational resources. Instead, the capacity of a server is partitioned among different service types, called as slices. In this architecture, each slice is customized and provides a dedicated capacity for a particular service type. With the help of isolation between slices, any changes or failure in a service has no impact on the performance of other services. Within this structure, the operator needs to optimize the deployment of the computational resources, the capacity allocation of service

instances, and task assignment operations while satisfying the stringent delay requirements of services with diversified characteristics.

In the deterministic network slicing problem, the aim is to introduce an optimal network slicing design algorithm for a capacitated environment where numerous service types having various characteristics and requirements can be accommodated by meeting the QoS specifications. We focus on the optimization of server placement, resource allocation, and task assignment decisions to maximize the revenue of the operator. We bring all these subproblems together and address them within a unified framework in an integrated MILP model. As in the case of computation architecture design problem, steady-state demand rates are used to formulate a mathematical optimization problem. In addition, we assume that the operator has a limited investment budget that can be spent to place computational resources. Our model also takes into account the upper bound for the service end-to-end delay restriction. In addition to the MILP formulation, two exact algorithms which are based on Benders decomposition and exploiting the special structure of the proposed formulation are presented for solving large instances where the MILP model becomes insufficient to obtain even a feasible solution within an acceptable time limit. We also introduce valid inequalities and problem-specific cut generation techniques to improve the efficiency of the suggested solution approach.

In the stochastic network slicing problem, we extend the scope of the previous problem to cover the stochasticity in the service request demands and propose an integrated solution approach under time-varying data traffic. We focus on the optimal placement and capacity allocation of computational resources by operators where dynamic changes in the service request patterns are also taken into consideration. To formulate the problem, we let the number of service requests be stochastic, but assume that it can be described by a finite random vector whose probability distribution is known in advance based on some historical data. By allowing changes in the service request patterns, the highly dynamic environment in the next-generation networks is taken into account.

In this problem, we aim to provide a network design scheme with delay-sensitive services to determine server placement and capacity allocation decisions in an optimal way for a new entrant operator. For this reason, we construct a two-stage stochastic integer programming model. The main objective of the problem is to minimize the capital cost of server placement decisions and the expected cost incurred by unsatisfied service requests within the specified delay limit. In addition, by defining a set of scenarios with known probability of occurrence to capture the uncertainty in the number of service requests, we develop a deterministic equivalent MILP model of the associated stochastic programming model. However, since the number of decision variables and constraints in this model is highly affected by the number of scenarios, it becomes unable to provide good-quality solutions in a reasonable amount of time when the number of scenarios is increased. Thus, we propose a Benders decomposition algorithm to solve larger instances where the MILP model becomes insufficient to obtain a feasible solution within an acceptable time limit. Moreover, we derive valid inequalities and simplify the subproblem to enhance the original formulations that can significantly improve the performance of our solution method. We also study a more realistic counterpart of the same problem where the capacity allocation of a server to different service instances can only occur at discrete levels. We apply a similar decomposition architecture to this problem and derive additional cuts to strengthen the formulation.

In order to validate the applicability of the proposed solution approaches and observe their performances, computational experiments are conducted using randomly generated topologies and test instances. While designing the experiments, some real-life aspects of computational architectures and the characteristics of next-generation services are incorporated in the data generation process as much as possible. Experimental results show that the proposed formulations are valid, and the developed solution methods significantly improve the solution quality and yield near-optimal results for even very large instances within the allocated time limit. Hence, we can conclude that the suggested algorithms can successfully address all the key components in a comprehensive way.

The content of this thesis is organized as follows: In Chapter 2, some terminology that is used throughout the thesis is introduced. Chapter 3 presents related works in the literature and summarizes the contribution of the thesis. The input parameters and decision variables along with their definitions are given in Chapter 4. In Chapter 5, we introduce the computation architecture design problem and present the MILP formulation and Lagrangian-relaxation based heuristic algorithm. In Chapter 6, we describe the deterministic network slicing problem and propose two solution algorithms based on Benders decomposition. Then, we continue with the stochastic version of the problem and present the stochastic network slicing problem along with a decomposition algorithm to efficiently solve the problem in Chapter 7. The efficacy of these approaches is tested on a large suite of randomly generated test instances. The results of computational experiments are summarized in Chapter 8. Finally, several open issues and challenges are discussed along with potential future research directions in Chapter 9.

2. BACKGROUND

2.1. Cloud and Edge Computing

With the emergence of novel applications demanding high computational resources, end-user devices are facing many challenges in terms of computational power, storage, and battery life since they have limited storage capacity and rather limited processing capability. Although current devices appearing in the market have higher resource capacity than before, they may not be able to handle the applications requiring considerable processing power in a short time. In addition, high battery consumption also restricts the users to run highly demanding applications on their own devices. Hence, it is not a straightforward solution to execute these services on the end-user devices with a high performance [15].

One potential solution to these challenges is to shift data processing and storage operations from the mobile device to powerful computing resources located in the cloud to enhance the computational capabilities required for these operations. Cloud data centers are capable of handling storage and processing of large scales of data. Thus, cloud computing can provide extended battery life, improved data storage, processing power, reliability, and scalability [16]. It allows users to utilize infrastructure, platforms and software provided by cloud providers (e.g., Google, Amazon, Facebook, Apple, and Microsoft) at low cost on a pay-as-you-go basis. It supports elasticity of computing, storage, and networking resources [17].

Nevertheless, cloud servers are often remotely located and far from the end-users. Although computing power and data processing capabilities are higher, the bandwidth of the network has been the bottleneck for the cloud-based network architecture because of the growing quantity of data generated by mobile devices. The increased data traffic destined to the cloud servers may impose huge additional load and create a burden on the already-congested network. Hence, it results in a significant Wide Area

Network (WAN) delay while accessing the cloud computing infrastructures. It makes the real-time interaction challenging to be achieved for the latency-intolerant services and weakens the user experience. For example, IoT applications may produce large amount of data that can create a heavy load on the network, may require very short response time and data privacy. Cloud computing is not solely sufficient enough to address these challenges and support next-generation services [18].

To overcome the aforementioned delay problem, an emerging concept called edge computing is proposed in recent years. It is considered as an enabling technology that allows computation to be performed at the edge of the network at the proximity of the end-user. The term "edge" is defined as any resource along the path between data sources and cloud data centers. While cloud computing is a centralized approach with highly powerful resources located at a single or few locations, edge computing is deployed in a fully distributed manner. It has the ability to extend cloud capabilities at the edge of the network by performing computationally-intensive tasks and storing massive amount of data at close proximity to end-user devices. With this approach, computing operations can be handled at the proximity of resources and large amount of data can be processed before sending to the cloud. Compared to the traditional cloud-based computation architecture, edge computing has the potential to improve the overall end-to-end latency, enable real-time interaction, and reduce energy consumption [3, 18, 19].

Edge computing is considered as a promising solution that can address the latency requirements, battery life limitations, high bandwidth demand, data security, and privacy issues [18]. By bringing the computational resources closer to the end-user, it creates a convenient environment to leverage the practical use of novel services. By keeping the network traffic at the edge and executing the demanded tasks without contributing to the congestion, WAN delay can be eliminated and user experience and the overall performance can be improved. It is more efficient to process the data at the edge of the network where the data is produced. Next-generation applications, such as augmented reality, autonomous driving, or healthcare applications, require real-time

interaction with the end-users. Moreover, these highly interactive applications are resource-hungry and have stringent QoS requirements like ultra-low latency and high reliability. Due to the limited capabilities of the end-user devices, edge computing appears as a solution to process and store massive amount of data without causing additional load on the network [3].

Even though the edge technology can deal with the excessive delay problem and is considered a remedy for real-time interaction, cloud servers may still be favorable in some cases to meet the diversified requirements. To combine the benefits of both approaches, edge and cloud servers may also co-exist on the same network to meet the user expectations and QoS requirements of differentiated services. By complementing the centralized cloud data centers with a pool of resources at the edge of the network, it can be possible to improve the user experience for computationally-intensive and delay-sensitive applications. This architecture can be further improved by integrating intermediary layers of computational resources with various capacity levels. With a smooth integration of multiple layers with different levels of computational capabilities, stringent requirements of services can be addressed in a single multi-level computing structure [13].

2.2. Network Slicing

In addition to traditional voice and broadband communication, 5G systems are expected to enable new innovation opportunities in different vertical industries. They have three main characteristics that are not provided in previous generation networks: high speed data transmission, lower latency, and the ability to connect a lot more devices. They are anticipated to be the cornerstone for numerous novel services, ranging from remote surgery to smart cities [11]. Most of these applications and services require high reliability, ultra-low latency, and real-time user interaction. These novel services with stringent requirements and IoT applications lead to tremendous increase in data traffic volume and computation demands. Traditional networks designed to provide voice and broadband services cannot cope with the exponential growth in the demand

of services that generates massive amount of data and require large computation power [20].

The one-size-fits-all type of architecture in the past telecommunication networks (2G, 3G, and 4G), where the same pipeline is utilized with almost no service customization, is no longer suitable to accommodate so many services and address their diverging performance requirements. The satisfaction of diversified service requirements can be achieved by partitioning a single physical network infrastructure into a set of isolated logical networks on a per-service basis. Recent advances in communication environment, such as Software-Defined Networking (SDN) and Network Function Virtualization (NFV), enable multiple services to coexist on top of a common underlying physical infrastructure [10]. SDN plays the role of cloud servers on the network by providing a centralized control mechanism to manage traffic flow and orchestrate network resource allocation to achieve high performance. On the other hand, NFV decouples the network functions such as firewall or load balancing from the underlying hardware and runs them as virtual network functions (VNFs) on virtual machines (VMs). With these technologies, isolated logical networks can be deployed over a single physical infrastructure [11, 20].

The network slicing concept has emerged as an enabling technology in which virtualized and independent logical networks can be deployed on the same physical network infrastructure. It is considered a promising technology for 5G networks to simultaneously accommodate on-demand vertical-specific services by sharing the same physical network resources in a sustainable way. The fundamental idea of network slicing is to divide the common physical network architecture into multiple logical and isolated networks that are configured to fulfill the needs of a particular use-case. Each slice can offer a customized service for a specific application scenario. Hence, each network slice is an isolated, self-contained end-to-end network designed to satisfy the requirements of a specific service [2, 21].

Network slicing plays a crucial role to meet the differentiated service requirements. Instead of a dedicated end-to-end network for each service, various services can share the networking and computational resources on a common physical infrastructure. Numerous service types having various characteristics and requirements can be accommodated simultaneously. By slicing a single physical network into multiple isolated logical networks, the networking resources can be efficiently allocated to customized services according to their particular QoS requirements. It enables to offer tailored solutions to different verticals and to use the network resources in a more efficient manner by creating specialized slices for each service type. Hence, capital and operational expenses can be reduced while providing customized service through limited resources [21].

The advantages of network slicing are multifold. Network slicing can improve the flexibility of network resource allocation [21]. It also assures isolation between slices which requires that the performance of a slice has no impact on the performance of another slice. In this way, the performance of a slice is not affected by the adjacent slice in case of a network failure, overload, or security attacks. In addition, privacy between slices is ensured since no private data is shared among different slices [2]. Each slice can be customized and dedicated to satisfy the requirements of the particular service instance. Hence, service differentiation can be achieved and SLA requirements can be guaranteed [22]. It allows service-specific resource allocation, which makes the network management more flexible and efficient. This service-oriented approach enables to design the network to meet the requirements of diversified services in a simplified and cost-efficient way [10].

Network slicing enables the network-as-a-service framework so that computing, storage, and networking resources are shared based on various service demands. In this architecture, it is known that optimal allocation of resources is crucial for the owner of the physical telecommunication infrastructure in terms of resource utilization and networking performance [10]. Operators can utilize network slicing to create customized virtual networks on top of a common physical infrastructure for a wide range

of use-cases demanding various and often conflicting requirements in terms of mobility, latency, and reliability [9]. An operator who wants to invest in this business needs to optimize its long-term strategic investment planning decisions arising in deployment of network slicing concept such that capital and operational expenses are minimized while the SLA-specified service requirements are fulfilled.

3. RELATED WORKS

In this chapter, the related works in the literature are discussed. The scope of the problems and proposed solution methodologies in these studies are summarized. Finally, the original contributions of the thesis are highlighted.

3.1. Studies on Computation Architecture Design Problem

Several studies that address the server placement, service deployment, and task assignment decisions with various objectives under different scenarios exist in the literature. Nevertheless, they still appear as popular research topics due to the expansion of novel use-cases and the development of computation methodologies [23–26]. In this section, recently published studies focusing on the computation architecture design problem are presented to reveal their scope and methodology and to highlight the original contributions of this study. The summary of a comprehensive literature review that briefly presents the objectives, content and methodology of related works is shown in Table 3.1.

Most of the existing studies formulate a mathematical optimization model and use operations research tools for the solution methodology. Saavedra et al. [27] study optimization of edge computing architectures and Radio Access Networks (RANs) to minimize RAN costs while maximizing the edge computing performance under network capacity constraints. They propose a mathematical optimization model that determines the function splits, the deployment of edge servers, and the routing of data between radio units and central unit. They use a weighted objective function that combines network expenditures and the end-to-end delay. They apply Benders decomposition to their proposed formulation to separate the problem into two subproblems: the routing problem and the network configuration problem. They show that their proposed method is reasonable as it provides good-quality solution for large problem instances.

Due to the high complexity of the problems under consideration, instead of exact solution methods, heuristic algorithms are also popular as an alternative approach to provide near-optimal solutions. Ceselli et al. [28] study edge cloud network design problem for mobile access networks. This problem requires to decide where to install cloudlets among potential sites and assignment of access points to those cloudlets in an optimal manner while satisfying the SLA definitions. A link-path MILP formulation with an exponential number of routing variables is introduced. A heuristic algorithm that combines local search, iterative rounding, and column generation techniques is utilized to achieve high-quality solutions in a reasonable amount of time. They show that the developed solution method also supports virtual machine orchestration with partial user mobility information.

Mondal et al. [29] focus on the static cloudlet network planning problem that aims to optimize the placements of cloudlets over existing optical access networks. The objective of this problem is to minimize the overall installation cost while satisfying the resource capacity restrictions and latency requirements. A mixed-integer nonlinear programming (MINLP) model is proposed to minimize the total installation expenditures. An open-source solver Couenne [30] is used to evaluate the performance of the proposed cloudlet placement framework over urban, suburban, and rural deployment scenarios.

Wang et al. [31] study the edge server placement problem in a large-scale environment to determine the placement of servers and task offload operations. They assume that servers are identical and have limited computational capacity. The objective of the problem is to minimize the access latency between end-users and edge servers while ensuring balance in terms of the workload among servers. They formulate an MILP model and use IBM CPLEX [32] as the solver. They perform computational experiments on a real dataset provided by Shanghai Telecom. They compare the solution quality obtained by solving the suggested formulation against several heuristic algorithms.

Li et al. [33] study the VNF placement problem considering service function chains in NFV and edge computing enabled networks. They formulate an MILP model that combines hierarchical structure and heterogeneous latency constraints to minimize the total resource consumption. This problem is a combination of facility location and multi-commodity flow problems, hence it is an NP-hard problem. Due to the high complexity of the problem, they also present a priority-based heuristic algorithm to solve it in a polynomial time. They show that this algorithm is able to find near-optimal solutions.

Zeng et al. [34] investigate the deployment of edge servers effectively and economically in wireless metropolitan area networks (MANs). They aim to minimize the number and the overall installation cost of edge servers while satisfying QoS requirements such as maximum acceptable delay limit. The experimental results obtained by simulation show that the proposed greedy heuristic and simulated annealing based algorithm provide promising solutions.

In a study by Li et al. [35], a dynamic resource management strategy is proposed to minimize the cost of the nodes rented from cloud providers while satisfying balance among the workloads of the edge cloud. The authors also investigate a dynamic replica allocation strategy that satisfies the user experience while reducing the storage overheads. A heuristic algorithm based on tabu-search is devised. The experimental results show that their proposed algorithm provide more effective solutions than the benchmark algorithms.

Farhadi et al. [36] optimize service deployment and request scheduling decisions in an integrated manner to serve data-intensive applications from the edge. They construct an MILP formulation that maximizes the expected number of requests served per slot subject to storage, communication, computation, and budget constraints. A polynomial-time heuristic algorithm that provides a constant approximation ratio under certain conditions is devised and its performance is evaluated through extensive simulations.

Sun et al. [37] study the replica server placement problem in an NFV environment which is modeled as a clustering problem. They aim to minimize the total traffic cost by reducing the response delay and bandwidth consumption by bringing the replica servers closer to the users while guaranteeing QoS requirements of services. An efficient heuristic algorithm based on spectral clustering called SC_CDN is designed and its performance is tested through simulation experiments having different traffic demand conditions.

Santoyo and Cervello [38] examine the fog node placement problem in an NFV environment. They formulate the problem as a capacitated facility location problem augmented with coverage constraints and present an MILP model by taking resource capacity and service latency restrictions into account. They also propose a hybrid simulated annealing algorithm. By computational experiments, they show that the suggested heuristic algorithm is able to provide promising results compared to the traditional simulated annealing algorithm.

Xu et al. [39] combine replica server placement, content caching, and request load assignment problems in content delivery networks (CDNs). They assume that computational and networking resources have a limited capacity. They develop an MILP model to minimize the ratio of unserved content request load. They decompose this model into three subproblems and suggest heuristic algorithms to solve each subproblem. They examine the effect of number of replica servers, link capacities, server processing capacities, and server storage capacities on the performance of the CDN. Their proposed algorithm is both advantageous in terms of time-complexity and gives near-optimum results.

In Baktır et al. [13], the computing infrastructure and the deployments of the services are assumed to be known in advance. They aim to introduce a task assignment scheme that guarantees SLA requirements of services in a multi-level computing architecture. They also consider fairness issues among different service types by imposing a minimum satisfaction ratio. The proposed formulations address the various

requirements of a service-oriented approach through an optimal task offloading scheme by considering the SLA of the service types. A heuristic implementation based on the nearest-fit algorithm is developed to obtain quick results.

To sum up, many studies in the literature investigate the server placement problem with the objective of minimizing the installation costs [28, 29, 37, 38, 40, 41], as can be seen in Table 3.1. Others try to minimize the service latency and the total energy consumption of the system and balance the workload among the servers [31, 42–46]. However, these studies do not consider an architecture where different capacity levels of servers work in harmony. Moreover, the requirements of the differentiated services are not taken into account. While each phase of the computation architecture design problem is dealt in a separate problem in the literature, an operator should consider them together to achieve revenue maximization.

Table 3.1. Properties of related works on the computation architecture design problem.

Study	Objective	Multi-level	Server Placement	Service Deployment	Task Assignment	Methodology
[27]	Minimizing the virtualized RAN costs and maximizing the edge cloud performance	✗	✗	✓	✓	Benders decomposition
[28]	Minimizing the installation cost of all network facilities, except networking links	✗	✓	✗	✓	Heuristic algorithm, column generation
[29]	Minimizing the overall cloudlet installation expenditures	✗	✓	✗	✓	MINLP formulation
[31]	Minimizing the access delay between user and edge server, balancing the workload among edge servers	✗	✓	✗	✓	MILP formulation
[33]	Minimizing the total resource consumption	✓	✗	✓	✓	MILP formulation, heuristic algorithm
[34]	Minimizing the number of edge servers	✗	✓	✗	✓	Greedy heuristic, simulated annealing based heuristic algorithm
[35]	Minimizing the cost of the nodes rented from the remote cloud	✓	✗	✓	✓	Tabu search based heuristic algorithm
[36]	Maximizing the number of served requests	✗	✗	✓	✓	Heuristic algorithm
[37]	Minimizing the total traffic cost of provisioning	✗	✓	✗	✓	Heuristic algorithm
[38]	Minimizing the number of fog nodes	✗	✓	✗	✓	MILP formulation, simulated annealing heuristic
[47]	Minimizing the total cost of edge system	✗	✗	✓	✓	Heuristic algorithm

Table 3.1. Properties of related works on the computation architecture design problem. (cont.)

Study	Objective	Multi-level	Server Placement	Service Deployment	Task Assignment	Methodology
[39]	Minimizing the ratio of unserved content request load	✗	✓	✓	✓	MILP formulation, heuristic algorithm
[42]	Minimizing the average service latency under migration cost budget constraints	✗	✗	✓	✓	Online algorithm
[43]	Minimizing the total energy cost and service time	✗	✗	✓	✓	Heuristic algorithm
[48]	Maximizing the number of concurrent requests that are satisfied within the system	✓	✗	✓	✓	Heuristic algorithm
[49]	Minimizing the overall latency of mobile devices	✗	✗	✓	✓	Heuristic algorithm based on Lyapunov optimization
[44]	Minimizing the computing and networking energy	✓	✗	✗	✓	Heuristic based on genetic algorithm, simulation with VirtFogSim [50]
[30]	Maximizing the number of primary servers and minimizing the number of backup servers	✗	✓	✗	✗	MILP formulation
[51]	Maximizing the total number of satisfied users and minimizing the overall cost	✗	✓	✗	✓	User clustering, MILP formulation
[45]	Minimizing the total energy consumption	✗	✓	✗	✗	Heuristic algorithm based on particle swarm optimization
[41]	Minimizing the number of edge servers while ensuring QoS constraints	✗	✓	✗	✓	Heuristic algorithm

Table 3.1. Properties of related works on the computation architecture design problem. (cont.)

Study	**Objective**	**Multi-level**	**Server Placement**	**Service Deployment**	**Task Assignment**	**Methodology**
[52]	Minimizing the total data access cost	✓	✗	✓	✓	Heuristic algorithm
[46]	Minimizing the disequilibrium of load within and among servers	✗	✗	✓	✗	Heuristic algorithm, simulation with CloudSim [53]
Our study	Maximizing the revenue of the operators while satisfying latency requirements	✓	✓	✓	✓	MILP formulation, heuristic algorithm based on Lagrangian relaxation

3.2. Studies on Network Slicing

The network slicing concept has been recently gaining momentum among researchers and attracted a lot of interest from both academia and industry, especially from the communities of operations research, networking, and computer science. In recent years, the network slicing design problem has extensively been studied under different objective functions and sets of constraints. An extensive literature survey on mathematical optimization models for network slicing in 5G networks can be found in [10]. In this section, we briefly mention some relevant studies in the literature with an emphasis on mathematical optimization formulations and techniques available in operations research.

Although there exist many studies that formulate different aspects of the network slicing problem using mathematical optimization techniques, these models can be hard to solve in polynomial time. Thus, instead of exact solution methods, heuristic approaches are commonly used to provide near-optimal solutions in a short amount of time due to high complexity of the problem. Vassilaras et al. [22] present the algorithmic challenges that one may encounter in efficient network slicing and state that these challenges can be handled and practical solutions can be obtained by operations research tools. They present an MILP model to choose the locations of network functions among a set of candidate locations and to decide the capacities of the connections between them to minimize the overall resource utilization cost, which is shown to be an NP-hard problem. They also introduce additional constraints to deal with some extensions of the problem such as survivability constraints, QoS constraints, optical network constraints, etc. However, their model requires immense computational capabilities, hence solving the model using a standard MILP solver is impractical for real-life network scenarios.

Destounis et al. [54] suggest an integer programming (IP) formulation to maximize the number of accepted slices and minimize the cost of embedding virtual networks. This model also takes QoS requirements and high reliability constraints into account.

Due to large number of binary variables in the model, in large problem sizes, the runtime grows exponentially if it is solved as a pure IP problem. Hence, as an alternative solution methodology, they relax the integrality constraints and propose a solution algorithm based on column generation that starts with a heuristic solution. At the end of the algorithm, the fractional solution is converted into an integral solution using rounding techniques.

Leconte et al. [55] provide an efficient and robust resource provisioning and autoscaling scheme for network slices with respect to different stakeholders, namely the slice owners, cloud and network providers. They formulate the resource allocation problem where network bandwidth and cloud processing power capacities are limited. Their model allocates network resources to slices while the satisfaction of delay constraints is guaranteed. They also propose an iterative heuristic algorithm based on the alternating direction method of multipliers that provides quick and efficient solutions. They perform extensive numerical simulations to show the effectiveness and the flexibility of their approach.

De Domenico et al. [56] focus on the deployment of VNFs and computational resource allocation in a hybrid environment with multiple edge clouds and one central cloud. They formulate an MILP model where the heterogeneous characteristics and latency requirements of differentiated services are taken into account to achieve high resource utilization efficiency. The objective function of this model is to minimize the total computational resources required to run VNF chains. As the resulting model is similar to the bin-packing problem, the computational complexity of the model with large number of variables and constraints is handled by defining a simple, low-complexity heuristic algorithm using the best-fit-decreasing strategy that generates near-optimal solutions to find feasible VNF deployments with a limited number of functional splits.

Zhang et al. [21] study the network slicing problem in an environment where a number of service requests can be processed and routed through the network simulta-

neously. They formulate an MILP model by allowing the traffic flows to be transferred on multiple paths and introduced some practical constraints. They also show that the problem is NP-hard in general. As it is computationally expensive to solve it to optimality, they suggest a heuristic algorithm called penalty successive upper bound minimization by relaxing the binary variables of the model and adding penalty terms into the objective function to obtain feasible solutions. They test the performance of their proposed approach through a set of computational experiments on simulated instances.

Xiang et al. [57] examine joint slicing of computational resources in a mobile network architecture with multiple service types having a maximum end-to-end latency limit. In this problem, the capacities of the networking nodes, links, and servers are assumed to be limited and it is ensured that all service requests are satisfied within the boundaries of their delay requirements. They suggest an MINLP formulation to minimize the overall latency that consists of transmitting, outsourcing, and processing of service requests subject to QoS requirements. Then, they equivalently reformulate it into a mixed-integer quadratically constrained programming problem. To deal with the complexity of the model, they use two heuristic algorithms: sequential fixing and a greedy heuristic to achieve good-quality solutions for large-scale scenarios in a short amount of time.

Fossati et al. [58] address multi-resource allocation problem by assuming that all service requests within the system do not have to be satisfied. They study how to share the network resources between slices in a fair manner on a network with limited computational capability to avoid excessive capacity allocation of resources. Their objective is to maximize the overall system efficiency while ensuring fairness and user satisfaction. They suggest a resource allocation framework by defining a set of resource allocation rules and ordered weighted average utility function to ensure fairness between slices. They test the efficiency and fairness of their solution approach by extensive simulations.

Lee et al. [59] propose a two-level dynamic resource allocation scheme including admission control, user association, baseband resource allocation, and transmission power allocation decisions to allocate network resources to different tenants. The objective of their proposed model is to maximize the weighted network throughput across all network slices considering priority, baseband resources, fronthaul and backhaul capacities, QoS requirements, and interference. They utilize dynamic programming, greedy heuristic algorithm, and Lagrangian dual method to solve different components of the problem. They test the performance of their approach in terms of throughput, fairness, and QoS performance against the baseline schemes.

Jiang et al. [60] aim to maximize the QoE while the satisfaction of the service requirements is guaranteed in 5G networks. They introduce a heuristic algorithm which takes inter-slice and intra-slice priority order into consideration and allocates network resources to slices dynamically. The performance of the proposed algorithm is tested through simulations and they show that the algorithm provides increased user experience and better utilization of network resources.

Bega et al. [61] introduce a mathematical optimization model for the allocation of network slices in order to maximize the infrastructure provider's revenue while satisfying the service requirements imposed by their SLAs. They design an adaptive algorithm based on Q-learning designed as a decision support mechanism for the infrastructure provider to decide whether to admit or reject a new network slice request. They demonstrate that this approach provides practical solutions with a near-optimal performance.

Sattar and Matrawy [62] focus on intra-slice virtual function isolation problem in core networks while satisfying end-to-end latency limits of differentiated service types. The goal of their study is to assign the incoming slice request to the least utilized server and to find a a path with minimum delay between the slice components. They formulate an MILP model that provides isolation between different components of a slice to ensure reliability. They evaluate the solution method by simulating a virtualized

mobile core.

Chen et al. [63] study the network slicing problem in terms of system energy efficiency. They aim to minimize the total power consumption of the whole network by minimizing the total number of activated cloud nodes while considering the resource budget, functional instantiation, flow routing, and end-to-end latency requirements of services. Their proposed MILP model allows the traffic flows to be transmitted through multiple paths. Service function chain constraints and the capacities of nodes and links in the network architecture are also taken into account in this model. They demonstrate the advantages of their formulation over the existing studies by conducting computational experiments.

As can be observed, there has already been a huge effort spent on the network slicing problem in recent years. Many authors also investigate the resource allocation problem for network slicing ([64–67]). In addition, different aspects of network slicing such as mobility, security, robustness, flexibility, and security issues have been extensively taken into account ([68–76]). A more detailed survey on the existing studies in network slicing can be found in [2, 10, 20, 21].

While optimization of the server placement and resource allocation decisions in deterministic network slicing problem has attracted a lot of interest in the literature, there are a limited number of research studies on its stochastic counterpart. In fact, the stochastic approach is able to design a more flexible network in terms of resource management, can provide better utilization of resources, and can deal with time-varying changes within the network.

Baumgartner et al. [77] propose a mathematical optimization model for cost-effective deterministic network slice design problem where the objective is to minimize the sum of capacity installation and consumption costs. Then, they extend their formulation to deal with the uncertainties in the service demand and consider the stochastic nature of the demand in a dynamic environment. They assume that the distribu-

tion of the service demand is symmetric and bounded, hence they can model chance constraints by employing an appropriate robust uncertainty set. They also provide a protection mechanism for network slices against single network element failure, such as node or link failures, to obtain a robust and survivable network slice design model. They use Gurobi [78] optimizer to solve the integrated model. However, due to large complexity of their proposed model, they are only able to test their approach on very small networks containing 12, 14, and 17 vertices in their computational experiments. In addition, they do not consider any SLA-specified service requirements such as maximum delay limit.

Sharma et al. [79] address the dynamic network slicing problem considering growth, provisioning, capacity sizing, and deletion operations of network slices based on a utility model. They introduce an algorithm that gives a framework for virtual topology of VNFs in data-centers to share and better utilize network resources among different slices. They also present a two-stage stochastic optimization model to handle uncertainty in the number of service requests by assuming that the random demand vector can be defined by a finite number scenarios based on some historical information. The objective of this model is to minimize the number of active slice instances within the network. They compare the performances of both approaches on a test instance with a network topology having 11 vertices and 34 links. In addition, they define 6 scenarios to obtain the second-stage decisions by assuming that the service demand has a known probability distribution.

Zhang and Wong [80] examine two-timescale resource management problem in network slicing and assume that the service demand can vary over time with known probability distribution. They formulate it as a two-stage stochastic integer programming model that aims to maximize the total profit of a tenant while satisfying the QoS requirements of services. In the first stage, before the actual realization of service demand, long-term resource reservation decisions are optimized. Then, the second-stage model determines the short-term intra-slice resource allocation decisions given the decisions of the previous stage. They transform the proposed stochastic programming

model into a deterministic equivalent MILP model by introducing a maximum interference threshold and applying semi-definite relaxation. To obtain sub-optimal solutions, they also utilize branch-and-bound and primal-relaxed dual techniques.

3.3. Studies in the Field of Operations Research

Recent advances in mobile communication industry have also led to novel research perspectives where operations research tools are utilized to obtain practical solutions to challenging problems arising in this domain. Bektaş et al. [81] focus on the joint object placement and request routing problem for a CDN from an operational point of view by assuming that commercial content providers already have established a number of proxy servers. They formulate a nonlinear integer programming model to minimize the total distribution cost by replicating the content on the proxy servers and assigning the client requests to an appropriate server in a CDN architecture. They also set an upper bound on end-to-end object transfer time to guarantee the QoS requirements. They linearize the proposed model and design two algorithms based on Benders decomposition and Lagrangian relaxation by exploiting the structural property of the model to solve the integrated problem. They generate random internet topologies to demonstrate the effectiveness of their proposed approaches compared to the state-of-the-art integer programming solver.

Similarly, Sen et al. [82] deal with the static data segment allocation problem in an information network where access patterns do not change over time. The goal of the study is to locate very large database of files in a cost-efficient manner to optimize the locations of servers, file placement, and user assignment. Instead of presenting a large-scale integrated model, they construct a two-phased approach: in the first phase, files are clustered into a pre-specified number of segments and then in the second phase, these segments are located and the requests are assigned to segments. They suggest a Benders decomposition algorithm to solve large instances where the subproblems are equivalent to the p-median problem. They also utilize various performance-tuning strategies to improve the efficacy of their proposed algorithm.

Gendron et al. [83] study the location design problem of green wireless local area networks (WLANs) that aims to reduce the overall power consumption cost of the access points when the load is scarce. They integrate the decisions on powering-on a set of access points, power level assignment and user terminal assignment into a single model. To formulate the problem, they consider a bipartite network architecture containing a set of access points and user terminals. They try to satisfy the user demand by taking the capacity restrictions of the connections between these two sets. For each access point, discrete power levels are defined that can be used when it is powered-on. They develop an exact algorithm inspired by Benders decomposition and use branch-and-Benders cut method that can provide high-quality and robust solutions even on large instances.

Li and Aneja [84] study fault management in optical networks where a failure can cause large data loss or interrupt communication services. They examine the fault tolerant regenerator location problem to guarantee signal transmission and communication under link failures. The goal is to minimize the number of regenerator deployments such that each node pair can still communicate in case of a single-link failure. They also suggest a branch-and-Benders-cut framework where the subproblem is used to check the feasibility of the master problem variables. Instead of the classical linear programming (LP) duality-based Benders cuts, they derive combinatorial Benders cuts in each iteration. They demonstrate that their proposed approach can find high-quality solutions even for the large instances through computational experiments.

3.4. Studies on the Capacitated Facility Location Problem

A significant amount of research has been carried out on various extensions of the capacitated facility location problem assuming that the demand is known. However, the real demand is often random in nature. Hence, although the facility can deal with the average demand, it may not cope in case of a peak demand. Then, this facility is said to be congested [85]. In fact, the network facility location problem, where congestion arises and delay functions are used to approximate the queuing process at

facilities, has been studied extensively by the operations research community. This problem has applications ranging from emergency service systems (fire, ambulance, police) to networks of public and private facilities.

Berman and Mandowsky [86] address the location-allocation decisions on congested networks where some of the arriving demand cannot be served immediately due to server unavailability when a call for service arrives. Hence, such a demand must either wait in the queue or be lost. The authors propose an algorithm to simultaneously optimize the locations of facilities and the partitioning policy. Descrochers et al. [87] model the congested facility location problem in which the objective is to minimize the sum of customers' transportation and waiting times, and facilities' fixed and variable costs. They propose a nonlinear, convex integer programming formulation and use column generation technique within a branch-and-bound scheme to solve their proposed model.

A comprehensive review of congestion models in facility location problems is discussed in [85]. In addition, Berman and Krass [88] describe the main components, present the terminology, and discuss main streams of research, solution approaches, and challenges in facility location problems on networks in case of stochastic customer demand and potential congestion at the facilities.

3.5. Thesis Contributions

In this thesis, we focus on optimizing the strategic level decisions having long-term effect on the profitability of an operator. We assume that networking nodes and links are already deployed. We aim to suggest a decision making tool for investment planning of a new entrant operator in an environment with numerous service types having diversified characteristics. Most of these services are latency-intolerant, so their SLA impose a maximum limit on the end-to-end delay for successful task offload operations. In addition, they are differentiated in terms of their networking and computational resource requirements. The operator needs to determine the placement

and discrete capacity levels of computational resources to meet the user expectations in an economical way.

The server placement decisions also affect tactical and operational level decisions such as service deployment, resource allocation, and task assignment operations. Therefore, optimization of these decisions has to be considered in an integrated manner. In this thesis, three different problem instances that combines strategic, tactical, and operational level decisions are introduced under different assumptions:

- Computation architecture design problem,
- Deterministic network slicing problem,
- Stochastic network slicing problem.

In the computation architecture design problem, the primary aim is to optimize server placement, service deployment, and task assignment decisions to maximize the revenue of the operator. We assume that the distribution of future service demand is unknown, hence the expected service demand is utilized. Given that the operator has a limited investment budget, we formulate an MILP model that takes the computational requirements and end-to-end delay restrictions of services into account. Then, we design a Lagrangian relaxation-based heuristic algorithm to find good-quality solutions for large instances.

As can be observed through the literature review given in Table 3.1, our work differs from the existing studies in terms of the following original contributions: In our problem, the multi-level computation system with potential server placement sites and service deployments play a vital role in optimal computation environment design. Based on the deployment schemes, task assignment operations are optimized so that the operators can maximize their revenue while satisfying the latency restrictions. Therefore, three different subproblems, each of which turns out to be complex on its own, are considered together. We provide an efficient and comprehensive solution methodology to address all the phases in an integrated fashion.

Then, we examine the network slicing concept which is proposed as a key enabling technology in the design of next-generation wireless networks. In this system, the overall capacity of the computational resources is partitioned into slices. These slices are then offer dedicated capacity for distinct service types. It allows service-specific resource allocation, enables to accommodate various service types simultaneously by sharing a common physical network in a sustainable way, and ensures service isolation among slices.

The optimal network slicing problem in next-generation mobile networks requires optimization of server placement, resource allocation, and task assignment decisions while satisfying the QoS specifications of services. We study two variants of this problem. In the deterministic network slicing problem, we assume that the distribution of service demand is unknown. Hence, steady-state demand rates are utilized. An MILP model is formulated where the objective is to maximize the revenue of the operator while satisfying the delay requirements. On the other hand, in the stochastic network slicing problem, the scope of the problem is extended to take the stochastic behavior of service demand into account. In this problem, the probability distribution of the number of service requests is assumed to be given in advance. It is formulated as a two-stage stochastic integer programming model that aims to minimize the cost of server placement decisions and the expected cost of unsatisfied service requests. For both problems, exact solution algorithms based on Benders decomposition is developed to improve the solution quality.

Our study differs from the existing studies due to the fact that it integrates server placement, resource allocation, and task assignment decisions in a single model and provides an efficient solution methodology for large instances. To the best of our knowledge, these components of the optimal network slicing problem have not been sufficiently studied and considered simultaneously in the existing literature. In addition, as shown in Section 3.2, most of the existing studies suggest heuristic algorithms to find near-optimal solutions instead of exact solution methods because of the high complexity of the problem. Thus, they are unable provide an efficient and integrated so-

lution methodology for the decisions arising in deployment of computational resources to serve differentiated services on the same network architecture.

4. NOTATION

While formulating the problems considered in this thesis, it is assumed that a network topology (V, E), where V is the set of vertices and E is the set of links connecting vertex pairs, is given in advance. Each vertex $v \in V$ and each link $e \in E$ have a networking capacity represented by ψ_v and ω_e, respectively, in terms of Megabits per second (Mbps). It is assumed that these networking devices are already deployed on the network.

A number of end-users exist throughout the network requesting different services. These end-users with smart devices continuously generate service requests at each physical location. We assume that each end-user is connected to and access the service through the closest vertex on the network. Instead of dealing with each service request individually, the requests triggered by the end-users connecting to the same access point (AP) on a vertex are aggregated as a single point of demand. These aggregated demand points are referred to end-user locations and represented by the set U. Hence, it is assumed that there are multiple smart devices generating service requests at each end-user location.

The set of services with different characteristics and requirements is denoted by Q. Each service $q \in Q$ has a unit revenue of r_q that the operator gains for each successfully handled task assignment operation. Similarly, o_q represents the unit penalty cost that the operator needs to pay for an unsatisfied service request of type $q \in Q$. The service requests triggered by the end-users generate load on both computational and networking resources depending on the service type. The parameter m_q represents the expected load on servers for a type q service request in terms of millions of instructions (MI) to be executed. For a type q service request and its corresponding response, the expected networking load on vertices and links of the network are denoted by h_q^{req} and h_q^{res}, respectively, in megabits (Mbits). As most of the services, such as augmented reality and healthcare applications, provided by the operators in the next-generation

wireless networks require real-time interaction, they are latency-intolerant. This is assured by imposing a maximum acceptable delay for a successful task assignment in their SLAs. Thus, for a type q service request, an upper limit for the end-to-end delay, which consists of transmitting the request and the response between end-user and server locations and executing the code on servers, is denoted by α_q, in seconds. The total number of service requests of type q generated per second at the end-user location u is denoted as d_{uq}.

A set of potential locations where servers can be placed is assumed to be given in advance and denoted by the set S. A server can be at different capacity levels. The set L represents the discrete capacity levels of the servers. The processing capacity of a level l server is represented by n_l in millions of instructions per second (MIPS) and the capital cost for placing a level l server is denoted by a_l. In addition, the number of cores on a server at capacity level l is denoted by χ_l and the processing capacity of a single core is shown by B. The capacity of a server can be allocated to different services to satisfy their SLA requirements. To proliferate efficient resource utilization of a server, the maximum number of service instances that can be hosted on a server is restricted depending on the server capacity level. For a server at capacity level l, at most g_l different service instances can be deployed. It is assumed that the operator has a limited budget of b that can be spent for the total capital cost of server placement decisions. Note that all revenue/cost components are assumed to be on an annualized basis. A scaling factor γ is used to standardize different revenue/cost components.

In this thesis, we also let some parameters to be stochastic, but we assume that their joint probability distribution can be expressed using historical data or predictions. The random data vector $\boldsymbol{\xi}$ represents the set of stochastic parameters with a finite probability distribution and ξ represents the actual realizations of the random data. Let K denote the set of possible realizations of $\boldsymbol{\xi}$, also called scenarios. In addition, let p_k represent the corresponding probabilities of each scenario $k \in K$ such that $\sum_{k \in K} p_k = 1$.

When a task assignment process is inspected from the beginning to the end, it can be observed that the following operations contribute to the overall end-to-end delay:

- Routing the service request from the end-user location to the destination server through networking resources, i.e. vertices and links,
- Execution of the service code on the destination server,
- Routing the service response from the server location back to the end-user location.

For each task assignment operation, all of these three components are taken into account while calculating the end-to-end delay. To eliminate the possibility of excessive delay on computational and networking resources, a maximum utilization limit is set as ρ and σ, respectively.

In computer networks, to transmit a service request and its corresponding response between the end-user and server locations, the conventional approach is to transmit them via the shortest path with minimum hops [89]. Therefore, we assume that while routing the service request as well as the response between the end-user and server locations, they follow the shortest path in terms of the number of hops. In our models, a shortest path between each end-user and potential server location pair is found by breaking ties arbitrarily in the preprocessing step. The set of vertices and links on the min-hop route from end-user location u to potential server location s are represented by V_{us} and E_{us}, respectively.

Each service generates a different amount of load on the networking resources. It is assumed that the amount of time required to transmit the request and the corresponding response between end-user and server locations through the min-hop route, also called the transmission delay, depends on the expected networking load requirement of that particular service type and the networking capacities of the vertices and the links. The effect of congestion on the networking resources is ignored [90]. Then, the overall transmission delay in seconds for a type q service request generated at the

end-user location u and assigned to a server at location s, denoted as β_{uqs}, can be calculated using the equality

$$\beta_{uqs} = \sum_{v \in V_{us}} \frac{h_q^{req} + h_q^{res}}{\psi_v} + \sum_{e \in E_{us}} \frac{h_q^{req} + h_q^{res}}{\omega_e}. \tag{4.1}$$

For convenience, all index sets and parameters used in the formulations are summarized below:

Sets:

V	:	Vertices
E	:	Links
U	:	End-user locations
Q	:	Services
S	:	Potential server locations
L	:	Capacity levels of servers
K	:	Scenarios
V_{us}	:	Vertices on the min-hop route between end-user location u and potential server location s
E_{us}	:	Links on the min-hop route between end-user location u and potential server location s

Parameters:

ψ_v	:	Networking capacity of vertex v (Mbps)
ω_e	:	Networking capacity of link e (Mbps)
r_q	:	Unit revenue obtained by satisfying a service request of type q
o_q	:	Unit penalty cost incurred by unsatisfied service request of type q
m_q	:	Computation load for a type q service request (MI)
h_q^{req}	:	Network load for a type q service request (Mbits)

h_q^{res} : Network load for a type q service response (Mbits)

α_q : Maximum allowed delay limit for service type q (s)

d_{uq} : Total number of type q service requests per second generated at the end-user location u

a_l : Capital cost of a server at capacity level l

n_l : Processing capacity of server at capacity level l (MIPS)

χ_l : Number of cores on a server at capacity level l

g_l : Maximum number of service deployments on a server at capacity level l

b : Total budget of the operator for server placement decisions

B : Processing capacity of a single core (MIPS)

γ : Scaling factor

p_k : Probability of scenario k

$\boldsymbol{\xi}$: Random data vector representing the set of stochastic parameters

ξ : Actual realizations of $\boldsymbol{\xi}$

ρ : Maximum allowed utilization for computational resources

σ : Maximum allowed utilization for networking resources

β_{uqs} : Overall transmission delay for a type q service request generated at the end-user location u and assigned to a server at location s (s)

The problems examined in this thesis focus on optimizing the strategic and tactical level decisions including server placement, service deployment, and capacity allocation decisions as well as operational level decisions on task offload operations of an operator that will enter the market. To formulate a mathematical model for these problems, the following decision variables are defined:

Decision Variables:

X_{sl} : 1 if a server of capacity level l is placed at server location s; 0 otherwise

Y_{qs} : 1 if type q service is deployed on a server at location s; 0 otherwise

C_{qs} : Capacity per second of type q service deployed on potential server location s (MIPS)

θ_{uqs} : Fraction of type q service requests that are generated at end-user location u and assigned to a server at location s ($0 \leq \theta_{uqs} \leq 1$)

Z_{uqs} : 1 if type q service requests at end-user location u are ever assigned to a server at location s; 0 otherwise

5. COMPUTATION ARCHITECTURE DESIGN PROBLEM

5.1. Problem Definition

In an environment where different capacity levels of servers can potentially co-operate, an operator can maximize its revenue in the future by optimally placing the computational resources, distributing the services within the network, and assigning the tasks generated by the end-users in an optimal way with a limited initial investment budget while meeting the end-users' expectations. The problem considered in this chapter focuses on finding an optimal server placement and service deployment scheme within the given budget limit of the operator. The overall aim is to maximize the revenue of the operator obtained by successfully handling the offloaded tasks while strict latency constraints are satisfied.

Our primary motivation is to optimize the computational resource deployments, the distribution of the service instances, and task assignment operations by addressing all these issues with a single and comprehensive solution methodology. Since this is a long-term strategic investment planning problem rather than being a real-time operational one, changes in the user behavior or operations within the network are not taken into account. In fact, at the initial investment phase, the actual distribution of the service requests is usually unknown. Hence, the operator needs to take action based on the average demand to maximize its expected revenue through the optimal deployment of computational resources. Although the service offload requests are time-varying, the operator can maximize its expected revenue by designing the system considering the steady-state demand rates.

5.2. Problem Formulation

Let F_s denote the total load per second on server location s generated by all service requests in terms of MIPS. Then, using the notation given in Chapter 4, an MILP formulation of the problem can be written as

$$\max \sum_u \sum_q \sum_s r_q d_{uq} \theta_{uqs} \tag{5.1}$$

$$\text{s.t.} \sum_l X_{sl} \leq 1 \qquad s \in S \tag{5.2}$$

$$\sum_s \sum_l a_l X_{sl} \leq b \tag{5.3}$$

$$\sum_q Y_{qs} \leq \sum_l g_l X_{sl} \qquad s \in S \tag{5.4}$$

$$\sum_s \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{5.5}$$

$$\theta_{uqs} \leq Y_{qs} \qquad u \in U, q \in Q, s \in S \tag{5.6}$$

$$F_s = \sum_u \sum_q m_q d_{uq} \theta_{uqs} \qquad s \in S \tag{5.7}$$

$$F_s \leq \rho \sum_l n_l X_{sl} \qquad s \in S \tag{5.8}$$

$$\sum_q \sum_{(u,s):v \in V_{us}} (h_q^{req} + h_q^{res}) d_{uq} \theta_{uqs} \leq \sigma \psi_v \qquad v \in V \tag{5.9}$$

$$\sum_l n_l X_{sl} - F_s \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{5.10}$$

$$\theta_{uqs} \leq Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{5.11}$$

$$\theta_{uqs}, F_s \geq 0 \qquad u \in U, q \in Q, s \in S \tag{5.12}$$

$$X_{sl}, Y_{qs}, Z_{uqs} \in \{0, 1\} \qquad u \in U, q \in Q, s \in S, l \in L. \tag{5.13}$$

Constraints (5.2) enforce that at most one capacity level can be placed at each potential server location. In constraint (5.3), the total expenditure is restricted so that the total capital cost of server placement decisions cannot exceed the given budget of the operator. On the other hand, to operate effectively, the maximum number of service deployments on a server is restricted based on the capacity level of the server.

Constraints (5.4) state that the total number of services hosted at each potential server location cannot exceed the maximum allowed number of service instances depending on the server capacity level decisions.

Any solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ satisfying constraints (5.2)–(5.4) provides a feasible server placement and service deployment scheme. However, the quality of these solutions can only be evaluated based on the total revenue obtained when the task assignments are realized. Therefore, the task assignment decisions are also integrated into this model, and the output of this part is used to assess the quality of the server placement and service deployment decisions.

The objective function can be defined as to maximize the overall revenue of the operator collected by the successfully handled service executions, as formulated in (5.1). Note that the revenue of the operator is affected directly by the task assignment decisions and indirectly by the server placement and service deployment decisions. Constraints (5.5) specify that for each service request generated at each end-user location, the sum of all task assignment fractions considering all potential destinations within the structure should be at most one. A service request can be assigned to a server if and only if there exists an instance of the corresponding service type on that server, which is guaranteed by constraints (5.6). The total load per second on each server location can be equivalently expressed as in constraints (5.7) in terms of MIPS. In order to avoid an indefinite amount of delay on computational resources, a maximum utilization bound is set for each potential server location, which is ensured by constraints (5.8). Similarly, a maximum utilization limit is guaranteed for each vertex of the network in constraints (5.9). Here, we assume that the service requests and the corresponding responses are always propagated through the shortest path in terms of the number of hops between each end-user and server location pairs. Then, the left hand side of the constraints (5.9) represents the total load per second on each vertex generated by all service requests and their corresponding responses on the network in terms of Mbps, and the utilization of each vertex is restricted by the parameter σ.

Enforcing a maximum utilization limit on the computational and networking resources to avoid excessive delay is not solely enough to provide real-time interaction and satisfy the maximum latency values imposed by the SLA definitions of the services. Besides, the overall service delay, from the instant of request generation until the reception of the response, should not exceed the maximum acceptable delay of the corresponding service type. To guarantee the satisfaction of the delay requirements of the services, the overall transmission delay between every end-user and server location pair is calculated in the preprocessing step using the equality (4.1) for each service type.

After calculating the time spent on the networking resources while routing the request and its response between the end-user and the target server locations, the remaining part of the end-to-end delay is the time required to execute the service code on the server. Thus, the maximum end-to-end delay requirement for a successful task assignment can be reduced to a maximum code execution delay requirement at the server location. As suggested by Jia et al. [91], we use the analytical formula for the expected time spent in an $M/M/1$ queuing system to represent the expected time for the code execution on a server by taking the congestion effect into account. Then, the maximum end-to-end delay requirement for a task assignment operation can be equivalently written as

$$\frac{m_q}{\sum_l n_l X_{sl} - F_s} \leq \alpha_q - \beta_{uqs}. \tag{5.14}$$

Please notice that both sides of the inequality are in seconds. This inequality must be satisfied to have a successful task assignment, i.e. if $\theta_{uqs} > 0$. As the transmission delay is relatively smaller than the time required to execute the service code on the server, we assume that the transmission delay between any end-user and server location for each service type is strictly less than the maximum allowed delay limit [92]. Hence, we have $\alpha_q - \beta_{uqs} > 0$ for $u \in U, q \in Q, s \in S$. By introducing an indicator binary variable Z_{uqs}, which takes value 1 if type q service requests from end-user location u are ever assigned to a server at location s, and 0 otherwise, the maximum end-to-end

delay requirement given in (5.14) can be converted into a set of linear constraints as in (5.10) and (5.11).

The proposed model (5.1)–(5.13) is capable of finding an optimal server placement and service deployment scheme in a service-oriented environment. It is achieved by evaluating the revenue gain of the operator collected through the successfully handled task offload operations satisfying the maximum delay requirements of the services. However, the number of decision variables and the number of constraints in the MILP model depend on the cardinalities of the index sets, which can be very large for a real-size architecture. On the other hand, our model extends the edge server placement problem, which is already proven to be an NP-hard problem [31], by accommodating the additional service deployment and task assignment decisions along with different capacity level options for servers. Moreover, the problem considered in the chapter has the p-median problem, a well-known NP-hard problem in discrete location theory [93], as a special case. Hence, the optimal solution to the overall problem cannot be obtained efficiently in polynomial time and it may require a prohibitively long amount of time to obtain an exact solution especially when the problem size gets larger. Therefore, a novel heuristic algorithm based on the Lagrangian relaxation of the proposed model is presented in the following section.

5.3. Lagrangian Relaxation-Based Heuristic Algorithm

As the numbers of end-user locations, service types, and potential server locations increase, the MILP model (5.1)–(5.13) may become incapable of finding a good-quality feasible solution within the allowed time limit. In this section, we present a Lagrangian heuristic algorithm as an alternative solution method to obtain a high-quality solution in a reasonable amount of time.

Lagrangian relaxation (LR) is a technique commonly used to find an upper bound for an MILP model with a maximization objective function. The main rationale of LR is based on the idea that in an MILP, there are some constraints that make the

problem difficult to solve, and the removal of these "complicating" constraints from the constraint set results in a problem which can be solved relatively easier compared to the original problem.

When the original problem is a pure IP problem P given as $\left\{ \max \mathbf{cx} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{Dx} \leq \mathbf{d}, \mathbf{x} \in \{0,1\} \right\}$ with optimal objective value z^*, then the Lagrangian relaxation of problem P with respect to the constraint set $\mathbf{Dx} \leq \mathbf{d}$ is defined by introducing a non-negative Lagrange multiplier vector $\boldsymbol{\lambda} \geq \mathbf{0}$, attaching each component of this vector to one of these constraints and bringing them into the objective function to obtain the Lagrangian upper bound program (LUBP), which can be expressed as $\left\{ z_{LR}(\boldsymbol{\lambda}) = \max \mathbf{cx} + \boldsymbol{\lambda}(\mathbf{d} - \mathbf{Dx}) : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \{0,1\} \right\}$. Note that for any $\boldsymbol{\lambda} \geq \mathbf{0}$, $z_{LR}(\boldsymbol{\lambda})$ becomes an upper bound for z^*, i.e., $z_{LR}(\boldsymbol{\lambda}) \geq z^*$. It is also worthwhile to mention that solving the LUBP turns out to be easier than the original problem P. In some cases, it can be even solved by inspection.

In order to obtain the best (smallest) upper bound (UB) which is as close as possible to z^*, one has to solve the Lagrangian Dual Program (LDP) given as $z_{LD} = \min_{\boldsymbol{\lambda} \geq \mathbf{0}} z_{LR}(\boldsymbol{\lambda})$, which requires finding the optimal values of Lagrange multipliers. This can be achieved by different techniques such as subgradient optimization and multiplier adjustment. Subgradient optimization is by far the dominant method used in the literature. It can be shown that for any IP or MILP formulation, the best UB that can be obtained by LR is at least as good as z_{LP} which is provided by the LP relaxation of P where the binary restriction of each component of the decision variable vector $\mathbf{x}$ is relaxed to the interval $[0,1]$. Namely, $z_{LD} \leq z_{LP}$. A formal proof can be found in [94, 95].

A nice property of LR is that it can also be used to develop a heuristic algorithm, known as Lagrangian heuristic (LH), to generate feasible solutions for problem P. The objective value corresponding to each of these feasible solutions provides a lower bound (LB) on z^*, and when the algorithm terminates, the best lower bound becomes the output of the LH. At each iteration of this heuristic, the LUBP is solved to generate

an upper bound. In order to produce the smallest (best) UB, the LDP is solved by subgradient optimization, which requires the update of the Lagrange multiplier vector $\boldsymbol{\lambda}$ according to the violations in the relaxed constraints using a step size parameter. This is achieved by generating a feasible solution to the original problem P from the optimal solution of the LUBP. The objective value of the feasible solution becomes a lower bound on z^*, and it is utilized at the same time to update $\boldsymbol{\lambda}$. This process is repeated until a stopping criterion is reached. The most widely used criteria are the gap between the best LB and best UB, the maximum number of iterations, and the allowed time limit. When the LH terminates, the best feasible solution found so far is a heuristic solution, and the objective value obtained is a lower bound on z^*. In fact, this lower bound is usually quite tight, even though the upper bound may be weak.

As mentioned earlier, the solution of the LUBP turns out to be much easier since one or more sets of constraints from the original problem P are relaxed. Sometimes, the LUBP can even be solved by inspection, which means that there is no need to solve a mathematical programming model, and it suffices to use a greedy mechanism to determine the optimal values of the decision variables in LUBP. In other cases like ours, as will be explained shortly, it becomes possible to decompose the LUBP into two or more subproblems so that each subproblem can be solved independently by spending a smaller computational effort. This significantly reduces the time complexity of solving the LUBP compared to the computation time required to solve P. Unfortunately, as is the case with metaheuristic algorithms, no performance guarantee can be given based on a theoretical analysis for the LH. The only way to show its effectiveness is to use it on benchmark instances for which optimal or best-known objective values are known or to apply it on randomly generated instances and compare the performance with other methods.

5.3.1. Lagrangian Relaxation

When our formulation given in the previous section is examined, it can be observed that the overall process of designing a computation infrastructure can be de-

composed into two separate but interrelated subproblems: (1) server placement and service deployment problem (2) task assignment problem. The constraints (5.6), (5.8) and (5.10) combine the decision variables of both subproblems, so they are considered the complicating constraints that increase the complexity of the problem. To obtain a Lagrangian relaxation of the proposed MILP model (5.1)–(5.13), these constraints are relaxed in a Lagrangian fashion by associating non-negative Lagrange multipliers δ_{uqs}, ϵ_s, and ζ_{uqs}, respectively. Then, the resulting Lagrangian relaxation problem can be expressed as

$$\begin{aligned}\max \sum_u \sum_q \sum_s \Bigg[& r_q d_{uq} \theta_{uqs} + \delta_{uqs}\Big(Y_{qs} - \theta_{uqs}\Big) + \epsilon_s\Big(\rho \sum_l n_l X_{sl} - F_s\Big) \\ & + \zeta_{uqs}\Big[\sum_l n_l X_{sl} - F_s - \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs}\Big]\Bigg] \\ \text{s.t. } & (5.2) - (5.5), (5.7), (5.9), (5.11) - (5.13).\end{aligned}$$

Note that for given values of multipliers δ_{uqs}, ϵ_s, and ζ_{uqs}, this formulation can be decomposed into two easier-to-solve subproblems. The first subproblem includes only binary $\boldsymbol{X}$ and $\boldsymbol{Y}$ decision variables, and the solution for this problem provides a feasible server placement and service deployment scheme. The second problem is expressed in terms of the remaining task assignment-related decision variables. However, the capacity levels of the deployed servers and the service distribution decisions obtained by the first subproblem are ignored while solving the task assignment part due to the relaxed constraints.

5.3.2. Subgradient Optimization

One of the challenging aspects of applying the LR approach is to determine the appropriate values for the Lagrange multipliers. In order to achieve this objective, the traditional subgradient optimization scheme suggested by Fisher [94] is utilized where Lagrange multipliers are updated at each iteration of the procedure until a termination criterion is satisfied.

In this iterative process, initially, the parameter π is set to a value within the set $(0, 2]$ and Lagrangian multipliers δ_{uqs}, ϵ_s, and ζ_{uqs} are initialized as zero. At each iteration of the algorithm, the LUBP is solved to optimality with the optimal objective value denoted as Z_{Lag} and the best UB, namely UB^*, is updated if necessary. The solution of the LUBP is then used to find a feasible solution for the original problem and the best LB, LB^*, is updated. Subgradient vectors G_i are evaluated at the current solution for each relaxed constraint. A step size T is defined, which depends on the parameter π, the gap between the objective value of the best known solution (LB^*) and that of the current solution (Z_{Lag}), and the squared norm of the subgradient vectors. Finally, the multipliers are updated using the step size and the subgradient vectors. The step size is a critical factor for the convergence of the solution. To speed up the convergence, if UB^* has failed to decrease for a specified number of iterations, π is halved. After the termination condition is satisfied, LB^* is reported as the best feasible objective value for the original problem. All steps of the solution algorithm are summarized in Figure 5.1.

At any step of the algorithm, the optimal solution of the relaxed problem does not necessarily satisfy the constraints (5.6), (5.8), and (5.10). Therefore, this solution needs to be converted into a feasible solution, denoted as Z_{Feas}, with respect to the original problem to obtain a lower bound. Let $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}})$ be the optimal server placement and service deployment decisions obtained by solving the relaxed problem. Then, the following problem is solved to obtain the optimal task assignments for the current solution:

$$\max \sum_{u} \sum_{q} \sum_{s} r_q d_{uq} \theta_{uqs} \tag{5.15}$$

$$\text{s.t.} \sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{5.16}$$

$$\theta_{uqs} \leq \tilde{Y}_{qs} \qquad u \in U, q \in Q, s \in S \tag{5.17}$$

$$F_s = \sum_{u} \sum_{q} m_q d_{uq} \theta_{uqs} \qquad s \in S \tag{5.18}$$

$$F_s \leq \rho \sum_l n_l \tilde{X}_{sl} \qquad s \in S \qquad (5.19)$$

$$\sum_q \sum_{(u,s):v \in V_{us}} (h_q^{req} + h_q^{res}) d_{uq} \theta_{uqs} \leq \sigma \psi_v \qquad v \in V \qquad (5.20)$$

$$F_s + \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \leq \sum_l n_l \tilde{X}_{sl} \qquad u \in U, q \in Q, s \in S \qquad (5.21)$$

$$\theta_{uqs} \leq Z_{uqs} \qquad u \in U, q \in Q, s \in S \qquad (5.22)$$

$$\theta_{uqs}, F_s \geq 0 \qquad u \in U, q \in Q, s \in S \qquad (5.23)$$

$$Z_{uqs} \in \{0, 1\} \qquad u \in U, q \in Q, s \in S. \qquad (5.24)$$

In this way, the feasibility of the resulting solution with respect to the original problem is ensured. Although this approach requires higher computational effort than using a simple, fast heuristic algorithm to obtain feasibility for the original problem, it is shown to be useful in obtaining good quality solutions.

5.4. Stochastic Variant of the Problem

In this section, we extend the problem definition and investigate computation architecture design where dynamic changes in the number of service requests are taken into account and the resource consumption of different services is uncertain. The aim is to maximize the expected revenue of successfully handled service requests by optimally allocating computational resources within a limited budget. We assume that the number of service requests from an end-user location may vary in time. Similarly, two different requests from the same service type may require different load on the network and computational resources. So, we let the number of service requests and service load requirements be stochastic, but assume that their probability distribution is known.

5.4.1. Problem Formulation

Let $\boldsymbol{\xi} = (\boldsymbol{d}, \boldsymbol{m}, \boldsymbol{h})$ represent the random data vector corresponding to the number of service requests, computation and network load requirements with known distribu-

tion. Also let the parameters $\xi = (d, m, h)$ be actual realizations of the random data. Using this notation, two-stage stochastic integer programming formulation of the problem can be written as follows:

$$\max \mathbb{E}[Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\xi})] \tag{5.25}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{5.26}$$

$$\sum_{s} \sum_{l} a_l X_{sl} \leq b \tag{5.27}$$

$$\sum_{q} Y_{qs} \leq \sum_{l} g_l X_{sl} \qquad s \in S \tag{5.28}$$

$$X_{sl}, Y_{qs} \in \{0, 1\} \qquad q \in Q, s \in S, l \in L \tag{5.29}$$

where $Q(\boldsymbol{X}, \boldsymbol{Y}, \xi)$ is the optimal value of the second-stage problem

$$\max \sum_{u} \sum_{q} \sum_{s} r_q d_{uq} \theta_{uqs} \tag{5.30}$$

$$\text{s.t.} \sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{5.31}$$

$$\theta_{uqs} \leq Y_{qs} \qquad u \in U, q \in Q, s \in S \tag{5.32}$$

$$F_s = \sum_{u} \sum_{q} m_q d_{uq} \theta_{uqs} \qquad s \in S \tag{5.33}$$

$$F_s \leq \rho \sum_{l} n_l X_{sl} \qquad s \in S \tag{5.34}$$

$$\sum_{q} \sum_{(u,s): v \in V_{us}} (h_q^{req} + h_q^{res}) d_{uq} \theta_{uqs} \leq \sigma \psi_v \qquad v \in V \tag{5.35}$$

$$\sum_{l} n_l X_{sl} - F_s \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{5.36}$$

$$\theta_{uqs} \leq Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{5.37}$$

$$\theta_{uqs}, F_s \geq 0 \qquad u \in U, q \in Q, s \in S \tag{5.38}$$

$$Z_{uqs} \in \{0, 1\} \qquad u \in U, q \in Q, s \in S. \tag{5.39}$$

Note that $Q(\boldsymbol{X}, \boldsymbol{Y}, \xi)$ is a function of the first-stage decision variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, and a realization $\xi = (d, m, h)$ of the random parameters. $\mathbb{E}[Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\xi})]$ denotes the expected revenue obtained by the satisfaction of the service requests. In the first stage, we determine the server placement and service deployment decisions before the realization of the uncertain data. In the second stage, after a realization of ξ becomes available, we optimize the task assignments for the given server placement and service deployment decisions.

The objective function of the first stage problem (5.25) tries to maximize the expected revenue collected by successfully offloaded service requests. Constraints (5.26) enforce that at most one capacity level of server can be placed at every potential server location. Constraint (5.27) guarantees that the total capital expenditures for server placement decisions cannot exceed the given budget limit of the operator. Constraints (5.28) state that the total number of service deployments cannot exceed the maximum number depending on the server level decisions at each potential server location.

In the second stage, when the server placement and service deployment decisions are made and the uncertain data is revealed, the model optimizes the task assignment decisions to maximize the revenue of the operator while satisfying the delay requirements. The objective function (5.30) aims to maximize the total revenue obtained by successfully handled task assignments. Constraints (5.31) state that a service request from an end-user location can be assigned to at most one server. A task assignment is valid only if the corresponding service instance is deployed on that particular server location. This is guaranteed by constraints (5.32). The total flow on computational and network resources generated by all service requests are calculated in constraints (5.33) and (5.35), respectively. To prevent excessive delay on both resources, the maximum utilization is bounded, which is ensured by constraints (5.34) and (5.35). Finally, as most of the services in such an environment are latency-intolerant and their SLA definitions may impose maximum latency values to enhance the user experience, the end-to-end delay for each successful service request should not exceed the maximum delay limit of that service. The maximum allowed delay requirement is expressed in

constraints (5.36) and (5.37).

5.4.2. Sample Average Approximation (SAA) Method

It is difficult to solve the stochastic program (5.25)–(5.39) since $\mathbb{E}[Q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\xi})]$ cannot be written in a closed-form expression. However, the function $Q(\boldsymbol{X}, \boldsymbol{Y}, \xi)$ can be computed for the given first-stage decision variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Therefore, we implement SAA scheme described by Kleywegt et al. [14]. In this method, a random sample of realizations (ξ) are generated and the expected value for the objective function of the stochastic program is approximated by the sample average function. After that, deterministic optimization techniques are used to solve the sample average approximating problem. This procedure is repeated until a stopping criterion is satisfied. At the end of the algorithm, we obtain an estimate for the optimality gap.

The SAA algorithm consists of three phases: In the first phase, we generate M independent samples of size N and solve the SAA problem. Then, we calculate an upper statistical bound for the optimal value of the true problem. In the second phase, by fixing each optimal solution obtained in the first phase of the SAA method, we solve the same problem with a sample size of N'. The solution providing the largest estimated objective value is used to obtain an estimate of a lower bound for the true optimal value by solving N'' independent second-stage problems in the third phase. Finally, we compute an estimate of the optimality gap and its estimated standard deviation.

1: Initialize multipliers δ, ϵ, and ζ as zero vectors.
2: $k = 1$, $LB^* = -\infty$, $UB^* = \infty$
3: Initialize $\pi \in (0, 2]$.
4: **while** Time limit has not been reached **do**
5: Solve the relaxed problem. Let the objective value of the relaxed problem be Z_{Lag}.
6: **if** $Z_{Lag} < UB^*$ **then**
7: $UB^* = Z_{Lag}$
8: **end if**
9: Modify the solution of the relaxed problem into a feasible solution. Let Z_{Feas} denote the objective value of this solution.
10: **if** $Z_{Feas} > LB^*$ **then**
11: $LB^* = Z_{Feas}$
12: **end if**
13: **if** UB^* has not improved for ω iterations **then**
14: $\pi = \pi/2$
15: **end if**
16: Define subgradients vectors:
17: $(G_1)_{uqs} = Y_{qs} - \theta_{uqs}$
18: $(G_2)_s = \rho \sum_l n_l X_{sl} - F_s$
19: $(G_3)_{uqs} = \sum_l n_l X_{sl} - F_s - \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs}$
20: Define a step size $T = \dfrac{\pi(LB^* - Z_{Lag})}{\sum_i ||G_i||^2}$
21: Update the multipliers:
22: $\delta^{k+1} = max\{0, \delta^k + TG_1\}$
23: $\epsilon^{k+1} = max\{0, \epsilon^k + TG_2\}$
24: $\zeta^{k+1} = max\{0, \zeta^k + TG_3\}$
25: $k = k + 1$
26: **end while**
27: Output LB^* as the best feasible objective

Figure 5.1. Lagrangian relaxation-based heuristic algorithm.

6. DETERMINISTIC NETWORK SLICING PROBLEM

6.1. Problem Definition

In this chapter, we consider a computation architecture design where numerous service types having various characteristics and requirements can be accommodated by meeting the QoS specifications. We focus on the network slicing problem in which the operator determines the deployment of the computational resources, the capacity allocation of service instances, and task assignment operations. The capacity of the computational resources is not shared among different services as in the case of computation architecture design problem. Alternatively, the overall capacity of a server is divided into slices that correspond to a particular service instance. These slices are customized and dedicated to satisfy the requirements defined in the SLA of that particular service type.

The deterministic network slicing problem focuses on optimizing server placement and capacity allocation decisions so that the revenue obtained by successful task offloading operations is maximized. We combine all these subproblems together in a single integrated MILP model. We use steady-state demand rates to formulate the problem. Our model also takes into account the upper bound for the overall end-to-end delay restriction. We assume that the operator has a limited initial investment budget that can be spent on capital cost expenditures of computational resources.

6.2. Problem Formulation

To formulate the problem, let F_{qs} denote the total load per second on server location s generated by type q service requests in terms of MIPS. Then, using the notation given in Chapter 4, a mathematical model for the deterministic network slicing

problem can be written as

$$\max \sum_{u}\sum_{q}\sum_{s} r_q d_{uq} \theta_{uqs} \tag{6.1}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{6.2}$$

$$\sum_{s}\sum_{l} a_l X_{sl} \leq b \tag{6.3}$$

$$\sum_{q} C_{qs} \leq \sum_{l} n_l X_{sl} \qquad s \in S \tag{6.4}$$

$$\sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{6.5}$$

$$F_{qs} = \sum_{u} m_q d_{uq} \theta_{uqs} \qquad q \in Q, s \in S \tag{6.6}$$

$$C_{qs} - F_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} - M(1 - Z_{uqs}) \qquad u \in U, q \in Q, s \in S \tag{6.7}$$

$$\theta_{uqs} \leq Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{6.8}$$

$$C_{qs}, F_{qs}, \theta_{uqs} \geq 0 \qquad u \in U, q \in Q, s \in S \tag{6.9}$$

$$X_{sl}, Z_{uqs} \in \{0, 1\} \qquad s \in S, l \in L, u \in U, q \in Q \tag{6.10}$$

where M is a sufficiently large constant. The objective function (6.1) maximizes the overall revenue obtained by the successfully handled task assignment decisions. Constraints (6.2) ensure that at most one capacity level of server can be placed at any potential server location. The overall capital cost for server placement decisions is restricted by the budget of the operator, which is guaranteed by constraint (6.3). Constraints (6.4) imply that the total capacity allocated for all services should not exceed the capacity level of the deployed server for each server location in terms of MIPS. Constraints (6.5) state that the sum of all task assignment fractions originating at each end-user location for each service type should be at most one. Using the notation given above, the total load per second on server location s generated by type q service requests can be expressed as in equality (6.6) in terms of MIPS.

To satisfy the SLA requirements of differentiated services, we need to ensure that the overall end-to-end delay do not exceed the maximum allowed delay limit of

that particular service. As explained in Chapter 4, the potential transmission delay, denoted as β_{uqs}, indicating the time required to transmit the service request from the originating end-user location to the destination server location and to transmit its response back in the reverse direction through the vertices and the links on the network, is calculated using the equality (4.1) for each service type in the preprocessing step. Then, the overall end-to-end delay restriction of a service can be equivalently stated as a maximum execution delay on the server. Here, we also use the formula for the average time spent in an $M/M/1$ queuing system to represent the expected time for the code execution on a server. Then, the maximum delay requirement of a type q service request generated at end-user location u and assigned to a server at location s can be expressed as

$$\frac{m_q}{C_{qs} - F_{qs}} \leq \alpha_q - \beta_{uqs} \tag{6.11}$$

where F_{qs} and C_{qs} denote the total load and the capacity of type q service on server location s in terms of MIPS, respectively. Note that both sides of the inequality (6.11) are in seconds. In addition, similar to the computation architecture design problem, we assume that $\alpha_q - \beta_{uqs} > 0$ for $u \in U, q \in Q, s \in S$.

To represent the latency requirement given in (6.11) as a set of linear constraints, we define an indicator binary variable Z_{uqs} that takes value 1 if type q service requests at user location u are ever assigned to server location s (i.e. $\theta_{uqs} > 0$) and 0 otherwise. This is ensured by constraints (6.8). Then, the maximum delay requirement given in (6.11) can be expressed as constraints (6.7).

Proposition 6.1. *A tighter version of constraints* (6.7) *can be obtained by replacing* $M = \frac{m_q}{\alpha_q - \beta_{uqs}}$.

Proof. By replacing $M = \frac{m_q}{\alpha_q - \beta_{uqs}}$, constraints (6.7) become

$$C_{qs} - F_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S. \tag{6.12}$$

If $\theta_{uqs} > 0$, it implies $Z_{uqs} = 1$. In that case, both constraints (6.7) and (6.12) state that $C_{qs} - F_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}}$, which is equivalent to (6.11), and the delay requirement is satisfied. On the other hand, if $Z_{uqs} = 0$, it implies $\theta_{uqs} = 0$ and constraints (6.7) become redundant. Meanwhile, constraints (6.12) become $C_{qs} - F_{qs} \geq 0$, which are still valid since the total load per second on each server for each service type cannot exceed the allocated capacity. □

By replacing constraints (6.7) with (6.12), the big-M structure in the MILP model can be eliminated. In this way, numerical problems that may arise with the use of big-M can be prevented, and a better linear programming (LP) relaxation bound can be obtained. Therefore, constraints (6.7) are replaced with (6.12) in the sequel of the thesis.

6.3. Decomposition Approaches

In the MILP model given in Section 6.2, the number of decision variables and the number of constraints are $O(|U||Q||S||L|)$. Therefore, it may be computationally difficult to solve this problem for real-sized networks with a large number of end-user and potential server locations. In this section, we present two exact solution algorithms based on Benders decomposition to efficiently solve the problem.

6.3.1. Decomposition based on Server Placement and Binary Task Assignment Decisions

Our first decomposition approach determines the server placement and binary task assignment decisions by solving a master problem, which contains decision variables $\boldsymbol{X}$ and $\boldsymbol{Z}$ only. In the subproblem, given these decisions, the optimal capacity allocation and fractional task assignment decisions are determined, and the revenue of the corresponding solution is obtained by solving an LP problem. To decompose the MILP model given in Section 6.2, we first observe that it can be reformulated in terms of only binary variables and an additional continuous variable t, which estimates

the maximum revenue of the operator corresponding to the current solution, given as follows:

Master Problem 1:

$$\max\ t \tag{6.13}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{6.14}$$

$$\sum_{s} \sum_{l} a_l X_{sl} \leq b \tag{6.15}$$

$$0 \leq t \leq t_{UB} \tag{6.16}$$

$$X_{sl}, Z_{uqs} \in \{0,1\} \qquad u \in U, q \in Q, s \in S, l \in L \tag{6.17}$$

where t_{UB} denotes an upper bound for the revenue obtained by the task assignment decisions. As the variable t predicts the maximum revenue of the operator, we can set $t_{UB} = \sum_{u} \sum_{q} r_q d_{uq}$ by assuming that all service requests are assigned. Note that this model contains significantly fewer decision variables and constraints than the original MILP model, which is advantageous from a computational point of view.

By solving this master problem, we obtain server placement decisions, denoted as $\tilde{\mathbf{X}}$, satisfying the initial investment budget. In addition, the model identifies a set of service requests from each end-user location that can be ever assigned to a server location through the binary task assignment decisions, denoted as $\tilde{\mathbf{Z}}$. Given these decisions, the optimal capacity allocation and fractional task assignment decisions, and the revenue of the operator gained by successfully handled service requests can be found by solving the following subproblem:

Subproblem 1 $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$:

$$\max \sum_{u} \sum_{q} \sum_{s} r_q d_{uq} \theta_{uqs} \tag{6.18}$$

$$\text{s.t.} \sum_{q} C_{qs} \leq \sum_{l} n_l \tilde{X}_{sl} \qquad s \in S \tag{6.19}$$

$$\sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{6.20}$$

$$F_{qs} - \sum_{u} m_q d_{uq} \theta_{uqs} = 0 \qquad q \in Q, s \in S \tag{6.21}$$

$$C_{qs} - F_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \qquad u \in U, q \in Q, s \in S \tag{6.22}$$

$$\theta_{uqs} \leq \tilde{Z}_{uqs} \qquad u \in U, q \in Q, s \in S \tag{6.23}$$

$$\theta_{uqs}, C_{qs}, F_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S. \tag{6.24}$$

This formulation contains only continuous decision variables, hence it is an LP problem. Thus, it is easy to solve the subproblem and we can use its dual formulation to generate Benders cuts based on duality theory.

It can be noted that this formulation for the subproblem can be further simplified. First, let us assume that we are given a master problem solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$. Then, in the subproblem, for each service type $q \in Q$ and potential server location $s \in S$, it can be observed that one of the constraints (6.22) corresponding to $u = \underset{u \in U}{\operatorname{argmax}} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \right\}$ is always tighter than the others since it has the largest right-hand side. To remove the remaining constraints and have a simpler formulation for the subproblem, we can replace (6.22) with

$$C_{qs} - F_{qs} \geq \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad q \in Q, s \in S. \tag{6.25}$$

Another key observation is that if $\tilde{Z}_{uqs} = 1$ for some $u \in U, q \in Q, s \in S$, constraints (6.23) are redundant since constraints (6.20) are tighter. Otherwise, they imply $\theta_{uqs} = 0$. In that case, we can also remove the θ_{uqs} variable from the formulation. Hence, constraints (6.23) can be removed from the formulation by fixing $\theta_{uqs} = 0$ if $\tilde{Z}_{uqs} = 0$. Then, the subproblem can be equivalently written as

Subproblem 1 $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$:

$$\max \sum_{u} \sum_{q} \sum_{s} r_q d_{uq} \theta_{uqs} \tag{6.26}$$

$$\text{s.t.} \sum_{q} C_{qs} \leq \sum_{l} n_l \tilde{X}_{sl} \qquad s \in S \tag{6.27}$$

$$\sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{6.28}$$

$$F_{qs} - \sum_{u} m_q d_{uq} \theta_{uqs} = 0 \qquad q \in Q, s \in S \tag{6.29}$$

$$C_{qs} - F_{qs} \geq \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad q \in Q, s \in S \tag{6.30}$$

$$\theta_{uqs} = 0 \qquad u \in U, q \in Q, s \in S : \tilde{Z}_{uqs} = 0 \tag{6.31}$$

$$\theta_{uqs}, C_{qs}, F_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S. \tag{6.32}$$

With this transformation, the number of decision variables and constraints in the subproblem can be reduced significantly.

In each iteration of the standard Benders decomposition procedure, the master problem is solved to optimality. Then, given the master problem solution, the dual of the subproblem, which is also an LP problem, is solved. In our problem, for any given master problem solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$, if the dual of the subproblem is feasible, the corresponding subproblem also yields a feasible solution. Moreover, the current master problem solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$ is also feasible with respect to the original MILP formulation and it can be identified as a candidate solution. On the other hand, if the dual formulation of the subproblem is unbounded, it means that the corresponding subproblem is infeasible. In that case, we need to ensure that $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$ is eliminated from the feasible region of the master problem. This can be achieved by generating a Benders feasibility cut using an extreme ray of the dual of the subproblem and adding it to the master problem to be resolved in the next iteration [96].

Proposition 6.2. *For any given master problem solution* $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$*, the subproblem is feasible if and only if the following condition is satisfied:*

$$\sum_{q} \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \leq \sum_{l} n_l \tilde{X}_{sl} \qquad s \in S. \tag{6.33}$$

Proof. First assume that the condition (6.33) is satisfied. Let $\theta_{uqs} = 0$ for all $u \in U, q \in Q, s \in S$ and $C_{qs} = \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\}$ for all $q \in Q, s \in S$. Then, all constraints (6.27)–(6.32) are satisfied, hence the subproblem is feasible. Now, assume that the subproblem is feasible. Then, constraints (6.30) imply that

$$C_{qs} \geq \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad q \in Q, s \in S. \tag{6.34}$$

By integrating constraints (6.27), we obtain

$$\sum_{q} \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \leq \sum_{q} C_{qs} \leq \sum_{l} n_l \tilde{X}_{sl} \qquad s \in S \tag{6.35}$$

which is equivalent to condition (6.33). Hence, condition (6.33) is satisfied if the subproblem is feasible. □

Using Proposition 6.2, it can be seen that we do not need to solve the dual of the subproblem to check the feasibility of the subproblem. Alternatively, the feasibility of the subproblem can be checked using condition (6.33) for any given master problem solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$. With this observation, instead of classical LP duality-based Benders feasibility cuts, it is possible to derive combinatorial feasibility cuts without solving the dual of the subproblem. Note that for some $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$, if the condition (6.33) is not satisfied for some $s \in S$, we need to break the infeasibility of the subproblem. It follows that the value of at least one binary variable has to be different in all feasible solutions of the original MILP model. Then, as suggested by Codato and Fischetti [97], the combinatorial Benders cut to ensure the feasibility of the subproblem can be written

as

$$\sum_{l:\tilde{X}_{sl}=0} X_{sl} + \sum_{l:\tilde{X}_{sl}=1} (1 - X_{sl}) + \sum_{(u,q):\tilde{Z}_{uqs}=0} Z_{uqs} + \sum_{(u,q):\tilde{Z}_{uqs}=1} (1 - Z_{uqs}) \geq 1. \tag{6.36}$$

These constraints can be generated in a cutting-plane fashion and added to the master problem when the feasibility condition is not met by the current master problem solution. In our computational tests, we observed that if condition (6.33) is not satisfied for some $s \in S'$ such that $S' \subset S$, adding all potential infeasibility cuts of type (6.36) for all $s \in S'$ at the same time helps to generate feasible solutions during the initial iterations of the algorithm and accelerates the algorithm compared to the case where a single feasibility cut of type (6.36) is added in each iteration.

In any iteration of the decomposition algorithm, if the condition (6.33) is satisfied for the current master problem solution, the subproblem as well as its dual problem are feasible. For a candidate master problem solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$ satisfying the feasibility condition, let $\tilde{t}$ denote the objective function value of the master problem, κ_s, μ_{uq}, and ν_{qs} be the optimal dual multipliers associated with constraints (6.27), (6.28), and (6.30), and t^* be the optimal objective function value of the subproblem (or its dual since their objective function values are the same at optimality). If $\tilde{t} > t^*$, we add the following cut to the master problem and resolve it in the next iteration to obtain a new candidate solution:

$$t \leq \sum_s \kappa_s \Big(\sum_l n_l X_{sl} \Big) + \sum_u \sum_q \mu_{uq} + \sum_q \sum_s \nu_{qs} \Big(\frac{m_q}{\alpha_q - \beta_{u'qs}} Z_{u'qs} \Big) \tag{6.37}$$

where $u' = \underset{u \in U}{\operatorname{argmax}} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \right\}$. These constraints are derived using the optimality condition of the subproblem and called as Benders optimality cuts.

The decomposition approach causes the master problem to lose all the information associated with the decision variables in the subproblem. Hence, the master problem can provide a weak approximation for the original feasible region. This can result in a

large number of iterations and excessive solution times [98]. One way to accelerate the decomposition procedure is to strengthen the formulation of the master problem by introducing some valid inequalities at the beginning of the algorithm. In our problem, a key observation is that if a service request generated at an end-user location of any type is assigned to a server location, then there must be a server of any level located on that potential server location. Therefore, a valid inequality for the master problem formulation can be written as

$$Z_{uqs} \leq \sum_{l} X_{sl} \qquad u \in U, q \in Q, s \in S. \tag{6.38}$$

Although introducing this valid inequality to the master problem at the beginning of the decomposition algorithm increases the number of constraints in the formulation, it can improve the solution procedure by strengthening the formulation and result in faster convergence. Note that this inequality is also valid for the original MILP formulation given in the previous section. However, in the original MILP formulation, constraints (6.4) and (6.12) already imply (6.38). Hence, this inequality is redundant for the MILP model.

The overall decomposition procedure can be summarized as follows: The first step is to solve the master problem augmented with valid inequality (6.38) to optimality and obtain a candidate solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$ with the objective function value $\tilde{t}$. Since the master problem is obtained by relaxing some of the constraints from the MILP formulation, $\tilde{t}$ provides a UB for the optimal objective value. Then, the feasibility of the current master problem solution is checked using condition (6.33). If the solution does not satisfy the feasibility condition for the subproblem, a feasibility cut (6.36) is generated to eliminate the current solution and added to the master problem, which is then resolved in the next iteration to obtain a new candidate solution. Otherwise, the dual of the subproblem is solved. Let t^* denote the optimal objective value for the dual of the subproblem for the current solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$. Then, the corresponding subproblem solution with the same objective value is also feasible with respect to the original MILP model. Thus, t^* gives an LB for the optimal objective value of the original problem. If

$\tilde{t} > t^*$, a traditional Benders optimality cut of type (6.37) is generated using the optimal dual multipliers of the subproblem and added to the master problem to be resolved in the next iteration. Note that although there is no link between the objective function and the decision variables $\mathbf{X}$ and $\mathbf{Z}$ in the master problem formulation at the beginning of the algorithm, adding optimality cuts (6.37) provides a relationship between them in the upcoming iterations. The solution procedure is repeated until we have $\tilde{t} = t^*$. In that case, we have $LB = UB$ and the current solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}})$ provides an optimal server placement and binary task assignment decisions for the deterministic network slicing problem.

6.3.2. Decomposition based on Server Placement, Capacity Allocation, and Binary Task Assignment Decisions

The formulation of the second decomposition approach is very similar to the previous one, but it yields a special structure that enables us to decompose the subproblem into smaller and efficiently solvable parts. In this approach, the capacity allocation decisions are also determined in the master problem along with the server placement and binary task assignment decisions. We also use a continuous decision variable t in the objective function representing the revenue of the operator obtained by successfully handled task assignments. Thus, the master problem can be reformulated as

Master Problem 2:

$$\max\ t \tag{6.39}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{6.40}$$

$$\sum_{s} \sum_{l} a_l X_{sl} \leq b \tag{6.41}$$

$$\sum_{q} C_{qs} \leq \sum_{l} n_l X_{sl} \qquad s \in S \tag{6.42}$$

$$0 \leq t \leq t_{UB} \tag{6.43}$$

$$C_{qs} \geq 0 \qquad q \in Q, s \in S \tag{6.44}$$

$$X_{sl}, Z_{uqs} \in \{0,1\} \qquad u \in U, q \in Q, s \in S, l \in L. \tag{6.45}$$

An initial upper bound for the objective function value can be imposed by setting $t_{UB} = \sum_u \sum_q r_q d_{uq}$. Note that this formulation still contains fewer decision variables and constraints than the original MILP model. In addition, valid inequality (6.38) defined for the previous master problem is also valid for this model.

This master problem can be used to obtain a candidate solution that provides server placement, capacity allocation, and binary task assignment decisions, denoted as $(\tilde{\mathbf{X}}, \tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$. Then, the optimal fractional task assignment decisions and the maximum revenue that the operator can achieve corresponding to the candidate solution can be found by solving the following subproblem:

Subproblem 2 $(\tilde{\mathbf{X}}, \tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$ **:**

$$\max \sum_u \sum_q \sum_s r_q d_{uq} \theta_{uqs} \tag{6.46}$$

$$\text{s.t.} \sum_s \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{6.47}$$

$$F_{qs} - \sum_u m_q d_{uq} \theta_{uqs} = 0 \qquad q \in Q, s \in S \tag{6.48}$$

$$F_{qs} \leq \tilde{C}_{qs} - \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \qquad u \in U, q \in Q, s \in S \tag{6.49}$$

$$\theta_{uqs} \leq \tilde{Z}_{uqs} \qquad u \in U, q \in Q, s \in S \tag{6.50}$$

$$\theta_{uqs}, F_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S. \tag{6.51}$$

This subproblem formulation is still an LP problem. In addition, it can be observed that the formulation of the subproblem does not depend on the server placement decisions, namely $\tilde{\mathbf{X}}$, hence it can be defined for any given $(\tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$. Moreover, similar to our analysis in the first decomposition approach, by removing redundant constraints in (6.49) and (6.50) and fixing some decision variables, a simpler formulation for this subproblem can be expressed as

Subproblem 2 $(\tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$ **:**

$$\max \sum_u \sum_q \sum_s r_q d_{uq} \theta_{uqs} \tag{6.52}$$

$$\text{s.t.} \sum_s \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{6.53}$$

$$F_{qs} - \sum_u m_q d_{uq} \theta_{uqs} = 0 \qquad q \in Q, s \in S \tag{6.54}$$

$$F_{qs} \leq \tilde{C}_{qs} - \max_u \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad q \in Q, s \in S \tag{6.55}$$

$$\theta_{uqs} = 0 \qquad u \in U, q \in Q, s \in S : \tilde{Z}_{uqs} = 0 \tag{6.56}$$

$$\theta_{uqs}, F_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S. \tag{6.57}$$

Using the standard Benders decomposition procedure, the master problem is solved to optimality and a candidate solution $(\tilde{\mathbf{X}}, \tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$ is obtained. Then, the dual of the subproblem is solved for the given master problem solution. If it is unbounded, which means that the corresponding subproblem is infeasible, a Benders feasibility cut is generated by using an extreme ray of the dual problem to eliminate the current solution in the upcoming iterations. For our decomposition procedure, instead of deriving feasibility cuts when needed, we will present an alternative approach to ensure the feasibility of the subproblem in each iteration of the algorithm using the following observation:

Proposition 6.3. *For any given master problem solution* $(\tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$*, the subproblem is feasible if and only if the following condition is satisfied:*

$$\tilde{C}_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \qquad u \in U, q \in Q, s \in S. \tag{6.58}$$

Proof. First assume that the condition (6.58) is satisfied. Let $\theta_{uqs} = 0$ for all $u \in U, q \in Q, s \in S$. Then, all constraints (6.53)–(6.57) are satisfied, hence the subproblem is feasible. Now, assume that the subproblem is feasible. Then, constraints (6.55)

imply that

$$\tilde{C}_{qs} \geq \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad q \in Q, s \in S \tag{6.59}$$

which can be equivalently expressed as (6.58). Hence, the condition (6.58) is satisfied if the subproblem is feasible. □

Similar to the first decomposition approach, we can derive combinatorial feasibility cuts whenever condition (6.58) is not satisfied for some $u \in U, q \in Q, s \in S$ by the candidate master problem solution without solving the dual of the subproblem. However, it is known that feasibility cuts do not improve the upper bound for a maximization problem and we can have many iterations without any feasible solution at the initial iterations of the algorithm. However, it can be possible to eliminate all infeasible solutions and ensure the boundedness of the dual of the subproblem by introducing a set of valid inequalities to the master problem. So, the undesired feasibility cut generation process can be avoided [98]. Then, the algorithm needs to generate only Benders optimality cuts.

In our second decomposition approach, instead of generating a feasibility cut whenever the feasibility condition (6.58) is not satisfied by the current master problem solution, we add a set of valid inequalities to the master problem at the beginning of the decomposition algorithm having the following form

$$C_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S. \tag{6.60}$$

With this approach, it is ensured that for any given master problem solution $(\tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$, the feasibility condition (6.58) is met. Then, the dual subproblem is bounded and the corresponding subproblem is feasible. Thus, we can avoid deriving feasibility cuts. Although this approach may increase the complexity of the master problem, the feasibility of the subproblem and the boundedness of its dual are always guaranteed for

any given master problem solution.

In the classical Benders decomposition method, a single cut, either feasibility or optimality cut, is inserted to the master problem at each iteration of the algorithm using the dual information. However, once some variables are fixed in the master problem, it may be possible to determine the remaining variables by solving multiple smaller subproblems separately. Then, it is possible to add multiple cuts using the dual formulations of each smaller subproblems. In our formulations, it can be observed that the simplified version of the subproblem can be further decomposed into smaller problems for each service $q \in Q$ as follows:

Subproblem 2 for $\mathbf{q \in Q}$ $(\tilde{\mathbf{C}}, \tilde{\mathbf{Z}})$:

$$\max \sum_{u} \sum_{s} r_q d_{uq} \theta_{uqs} \tag{6.61}$$

$$\text{s.t.} \sum_{s} \theta_{uqs} \leq 1 \qquad u \in U \tag{6.62}$$

$$F_{qs} - \sum_{u} m_q d_{uq} \theta_{uqs} = 0 \qquad s \in S \tag{6.63}$$

$$F_{qs} \leq \tilde{C}_{qs} - \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}_{uqs} = 1 \right\} \qquad s \in S \tag{6.64}$$

$$\theta_{uqs} = 0 \qquad u \in U, s \in S : \tilde{Z}_{uqs} = 0 \tag{6.65}$$

$$\theta_{uqs}, F_{qs} \geq 0 \qquad u \in U, s \in S. \tag{6.66}$$

To generate multiple optimality cuts for each subproblem, instead of a single decision variable t, we need to define a set of decision variables t_q for each service type $q \in Q$ in the objective function of the master problem formulation. These variables are used to predict the maximum revenue of the operator that can be obtained by satisfied service requests of type q. Then, we replace the objective function (6.39) with $\sum_{q} t_q$ and the constraints (6.43) with $0 \leq t_q \leq t_{q,UB}$ where an inital upper bound can be set as $t_{q,UB} = \sum_u r_q d_{uq}$. Let τ_u and υ_s denote the optimal values of the dual variables associated with constraints (6.62) and (6.64), respectively. Then, using the

optimality conditions for each subproblem, the standard Benders optimality cut can also be decomposed for every service type $q \in Q$ as follows:

$$t_q \leq \sum_u \tau_u + \sum_s \upsilon_s \Big(C_{qs} - \frac{m_q}{\alpha_q - \beta_{u'qs}} Z_{u'qs} \Big) \tag{6.67}$$

where $u' = \underset{u \in U}{\operatorname{argmax}} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}_{uqs} \right\}$. Although this approach, referred to as multi-cut reformulation, causes the size of the master problem to grow rapidly, it generally outperforms the single-cut approach by more quickly strengthening the master problem [98]. Therefore, we prefer to use the multi-cut reformulation and add multiple Benders optimality cuts at each iteration.

As in the previous decomposition approach, we can introduce valid inequalities to the master problem in order to have better upper bounds during the initial iterations of the algorithm and accelerate the convergence.

Proposition 6.4. *The following inequality is valid for the multi-cut reformulation of the master problem:*

$$t_q \leq \frac{r_q}{m_q} \sum_s C_{qs} \qquad\qquad q \in Q. \tag{6.68}$$

Proof. In the objective function of the master problem formulation, t_q variables represent the revenue obtained by successfully handled task assignment operations for type q service requests. Therefore, we can write

$$t_q \leq r_q \sum_u \sum_s d_{uq} \theta_{uqs} \qquad\qquad q \in Q. \tag{6.69}$$

In addition, the total load on each server at potential location $s \in S$ cannot exceed the capacity allocated to any service instance $q \in Q$. By integrating constraints (6.63), we

have

$$F_{qs} = \sum_{u} m_q d_{uq} \theta_{uqs} \leq C_{qs} \qquad q \in Q, s \in S. \tag{6.70}$$

By summing both sides of the inequality (6.70) over all potential server locations $s \in S$, we obtain

$$m_q \sum_{u} \sum_{s} d_{uq} \theta_{uqs} \leq \sum_{s} C_{qs} \qquad q \in Q. \tag{6.71}$$

By combining (6.69) and (6.71), we can write

$$t_q \leq r_q \sum_{u} \sum_{s} d_{uq} \theta_{uqs} \leq \frac{r_q}{m_q} \sum_{s} C_{qs} \qquad q \in Q. \tag{6.72}$$

Thus, inequality (6.68) is valid for the multi-cut reformulation of the master problem. □

In Chapter 8, we apply multi-cut reformulation approach and add valid inequality (6.68) to the master problem in order to investigate the impact on the decomposition procedure through computational experiments.

7. STOCHASTIC NETWORK SLICING PROBLEM

7.1. Problem Definition

In this chapter, we study the stochastic network slicing problem to propose a comprehensive solution method in an environment having time-varying service demand. In fact, the total number of service requests at an end-user location may vary in time and the distribution of the service requests on the network may change due to human activity such as cultural and sports events. For example, the number of requests from a residential area may increase in the evenings and decrease during working hours. Hence, we let the number of service requests be stochastic, but assume that it can be described by a finite random vector whose probability distribution is known in advance based on some historical data. We aim to provide a network slicing scheme to optimize the server placement and capacity allocation decisions for an operator that will enter the market. We assume that the services are delay-sensitive and discrete capacity levels for servers are defined.

To formulate the stochastic network slicing problem, a two-stage stochastic integer programming model is constructed. The main objective of the problem is to minimize that the total cost which consists of the capital cost for server placement decisions and the expected cost of unsatisfied task offloading operations. In the first-stage problem, the model tries to optimize strategic level decisions made once at the beginning of the planning horizon including the server placement and the capacity allocation decisions before uncertainty in the service demand is revealed. Subsequently, given the first-stage decisions, the operational level decisions on task offload operations are optimized in the second stage while satisfying the stringent delay requirements of services with diversified characteristics. In this way, strategic and operational level decisions are integrated into a single model.

7.2. Problem Formulation

To formulate the problem, let F_{qs} denote the total load per second on server location s generated by type q service requests in terms of MIPS. Also let $\boldsymbol{\xi} = \boldsymbol{d}$ represent the random data vector corresponding to the total number of service requests in a second with known distribution and the parameters $\xi = d$ be actual realizations of the random data. Note that although the service requests arrive in a stochastic manner, it is assumed that the joint probability distribution can be expressed using historical data. By using the notation given in Chapter 4, a two-stage stochastic integer programming formulation of the problem can be written as

$$\min \sum_{s}\sum_{l} a_l X_{sl} + \gamma \mathbb{E}[Q(\boldsymbol{C}, \boldsymbol{\xi})] \tag{7.1}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{7.2}$$

$$\sum_{q} C_{qs} \leq \sum_{l} n_l X_{sl} \qquad s \in S \tag{7.3}$$

$$C_{qs} \geq 0 \qquad q \in Q, s \in S \tag{7.4}$$

$$X_{sl} \in \{0, 1\} \qquad s \in S, l \in L \tag{7.5}$$

where $\mathbb{E}[Q(\boldsymbol{C}, \boldsymbol{\xi})]$ denotes the recourse function. In this formulation, $Q(\boldsymbol{C}, \xi)$ is the optimal value of the second-stage problem, which can be expressed as

$$Q(\boldsymbol{C}, \xi) : \min \sum_{u}\sum_{q} o_q d_{uq}\big(1 - \sum_{s} \theta_{uqs}\big) \tag{7.6}$$

$$\text{s.t.} \sum_{s} \theta_{uqs} \leq 1 \qquad u \in U, q \in Q \tag{7.7}$$

$$F_{qs} = \sum_{u} m_q d_{uq} \theta_{uqs} \qquad q \in Q, s \in S \tag{7.8}$$

$$F_{qs} \leq C_{qs} - \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{7.9}$$

$$\theta_{uqs} \leq Z_{uqs} \qquad u \in U, q \in Q, s \in S \tag{7.10}$$

$$\theta_{uqs}, F_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S \tag{7.11}$$

$$Z_{uqs} \in \{0,1\} \qquad u \in U, q \in Q, s \in S. \tag{7.12}$$

Note that $Q(\boldsymbol{C}, \xi)$ is a function of the first-stage decision variable vector $\boldsymbol{C}$, and a realization ξ of the random parameters. $\mathbb{E}[Q(\boldsymbol{C}, \boldsymbol{\xi})]$ denotes the expected cost of unsatisfied service requests. In this formulation, we determine the server placement and capacity allocation decisions, denoted as $(\boldsymbol{X}, \boldsymbol{C})$ before the realization of the uncertain data in the first-stage problem. After a realization of ξ becomes available, we optimize the task assignments for the given server placement and capacity allocation decisions in the second stage problem.

In the first-stage problem, the objective function (7.1) minimizes the sum of server placement cost and the expected penalty cost of unsatisfied service demand. Constraints (7.2) state that for any potential server location, at most one server having any capacity level can be placed. The sum of allocated capacities for all service types is restricted by the capacity level of the deployed server at every potential server location, which is ensured by constraints (7.3).

After solving the first-stage problem, each realization of service demand ξ yields a second-stage model (7.6)–(7.12). The objective function of the second-stage problem (7.6) minimizes the penalty cost of unsatisfied task offload operations. Constraints (7.7) state that the total task assignment fractions generated at an end-user location can be at most one for each service type. The total load per second at potential server location s required by type q service requests can be written as equality (7.8) in terms of MIPS.

Since the novel services considered in this thesis are delay-sensitive, the overall latency should not exceed a pre-specified threshold. As in the deterministic network slicing problem, the potential transmission delay, denoted as β_{uqs}, is calculated using the equality (4.1) for each end-user and server location pair in the preprocessing step. The remaining execution delay on the server is formulated by using the expected time spent in an $M/M/1$ queuing system. Then, the maximum end-to-end delay requirement

of service type q requests from end-user location u and assigned to a server at location s can be written as inequality (6.11).

An indicator binary variable Z_{uqs} is introduced to represent the maximum delay requirement (6.2) as a set of linear constraints. This variable takes value 1 if type q service requests from end-user location u are ever assigned to potential server location s (i.e. $\theta_{uqs} > 0$) and 0 otherwise, which is guaranteed by constraints (7.10). Then, the maximum end-to-end delay requirement of services can be equivalently written as in constraints (7.9). Note that for $Z_{uqs} = 1$, both inequalities (6.11) and (7.9) are the same. On the other hand, for $Z_{uqs} = 0$, constraints (7.9) become $F_{qs} \leq C_{qs}$. This must also hold as the total load on a server is restricted by the allocated capacity for the corresponding service type in a unit time interval.

In the stochastic network slicing problem, the random vector $\boldsymbol{\xi}$ that shows the number of service requests in a second is assumed to have a finite probability distribution. Let K denote the set of possible realizations of $\boldsymbol{\xi}$, also called scenarios. In addition, let p_k represent the corresponding probabilities of each scenario $k \in K$ such that $\sum_{k \in K} p_k = 1$. Then, the stochastic parameter d_{uq} can be transformed into d^k_{uq} to represent the total number of service requests of type q generated at end-user location u in scenario k. Similarly, the decision variables in the second-stage problem can be replaced by θ^k_{uqs}, F^k_{qs}, and Z^k_{uqs} to represent the decisions under different scenarios. Hence, we can rewrite the two-stage stochastic integer programming problem (7.1)–(7.12) and obtain a deterministic equivalent MILP model as follows:

$$\min \sum_s \sum_l a_l X_{sl} + \gamma \sum_k p_k \left[\sum_u \sum_q o_q d^k_{uq} \big(1 - \sum_s \theta^k_{uqs}\big) \right] \tag{7.13}$$

$$\text{s.t.} \sum_l X_{sl} \leq 1 \qquad s \in S \tag{7.14}$$

$$\sum_q C_{qs} \leq \sum_l n_l X_{sl} \qquad s \in S \tag{7.15}$$

$$\sum_s \theta^k_{uqs} \leq 1 \qquad u \in U, q \in Q, k \in K \tag{7.16}$$

$$F_{qs}^{k} = \sum_{u} m_q d_{uq}^{k} \theta_{uqs}^{k} \qquad q \in Q, s \in S, k \in K \quad (7.17)$$

$$C_{qs} - F_{qs}^{k} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs}^{k} \qquad u \in U, q \in Q, s \in S, k \in K \quad (7.18)$$

$$\theta_{uqs}^{k} \leq Z_{uqs}^{k} \qquad u \in U, q \in Q, s \in S, k \in K \quad (7.19)$$

$$C_{qs}, \theta_{uqs}^{k}, F_{qs}^{k} \geq 0 \qquad u \in U, q \in Q, s \in S, k \in K \quad (7.20)$$

$$X_{sl}, Z_{uqs}^{k} \in \{0, 1\} \qquad u \in U, q \in Q, s \in S, k \in K, l \in L. \quad (7.21)$$

Thus, by defining a set of potential scenarios, the two-stage stochastic integer programming formulation of the problem (7.1)–(7.12) can be cast as a single large-scale MILP model.

7.3. Benders Decomposition

Note that the number of decision variables and constraints in the deterministic equivalent model (7.13)–(7.21) depend on the cardinality of the sets U, Q, S, and L. Thus, it can be computationally ineffective to solve this model as a single monolithic MILP problem for real-sized networks. In addition, an increase in the number of scenarios directly increases the complexity of the problem, which can deteriorate the stochastic programming solution quality. To deal with the complexity issue, we focus on deriving an exact solution algorithm based on Benders decomposition as an alternative solution approach.

7.3.1. Decomposition Structure

Our decomposition algorithm first solves the master problem containing binary variables $\boldsymbol{X}$ and $\boldsymbol{Z}$, which denote the server placement and binary task offload decisions, respectively. Although in the standard Benders decomposition algorithm, only integer variables are contained in the master problem and continuous variables are determined in the subproblem, we also include continuous capacity allocation decisions, namely variables $\boldsymbol{C}$, in the master problem because by fixing these decision variables,

the subproblem can be decomposed into smaller problems. These multiple smaller subproblems can be solved independently, which can be advantageous from a computational point of view. In addition, this structure enables adding multiple cuts in each iteration of the algorithm instead of inserting a single cut into the master problem. By this way, the master problem can be tightened in early iterations of the algorithm. We discuss the details and advantages of this approach in terms of computational effort in Section 7.3.2.

First, the deterministic equivalent model can be reformulated in terms of only variables $\boldsymbol{X}$, $\boldsymbol{C}$, and $\boldsymbol{Z}$. An additional continuous decision variable t is introduced to represent the expected cost of unsatisfied demand penalty corresponding to the current solution. Then, our master problem can be expressed as follows:

Master Problem:

$$\min \sum_{s} \sum_{l} a_l X_{sl} + \gamma t \tag{7.22}$$

$$\text{s.t.} \sum_{l} X_{sl} \leq 1 \qquad s \in S \tag{7.23}$$

$$\sum_{q} C_{qs} \leq \sum_{l} n_l X_{sl} \qquad s \in S \tag{7.24}$$

$$t, C_{qs} \geq 0 \qquad q \in Q, s \in S \tag{7.25}$$

$$X_{sl}, Z^k_{uqs} \in \{0,1\} \qquad u \in U, q \in Q, s \in S, k \in K, l \in L. \tag{7.26}$$

With this transformation, the number of decision variables and constraints are reduced compared to the deterministic equivalent model.

Any feasible solution to this master problem provides a candidate solution that contains server placement, capacity allocation, and binary task assignment decisions, denoted as $\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{C}}$, and $\tilde{\boldsymbol{Z}}$, respectively. Given a candidate master problem solution, the optimal fractional task assignment decisions are obtained by solving the following subproblem:

Subproblem $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$:

$$\min \sum_k p_k \big[\sum_u \sum_q o_q d^k_{uq} \big(1 - \sum_s \theta^k_{uqs}\big)\big] \tag{7.27}$$

$$\text{s.t.} \sum_s \theta^k_{uqs} \leq 1 \qquad u \in U, q \in Q, k \in K \tag{7.28}$$

$$F^k_{qs} - \sum_u m_q d^k_{uq} \theta^k_{uqs} = 0 \qquad q \in Q, s \in S, k \in K \tag{7.29}$$

$$F^k_{qs} \leq \tilde{C}_{qs} - \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}^k_{uqs} \qquad u \in U, q \in Q, s \in S, k \in K \tag{7.30}$$

$$\theta^k_{uqs} \leq \tilde{Z}^k_{uqs} \qquad u \in U, q \in Q, s \in S, k \in K \tag{7.31}$$

$$\theta^k_{uqs}, F^k_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S, k \in K. \tag{7.32}$$

The objective function of this subproblem (7.27) yields the expected penalty cost of unsatisfied service demand for the candidate master problem solution. Note that the subproblem (7.27)–(7.32) consists of only continuous decision variables, which makes it an LP problem. Then, its dual formulation can be utilized to generate the Benders feasibility and optimality cuts. The details of the cut generation techniques are presented in Section 7.3.2.

7.3.2. Algorithmic Improvements

7.3.2.1. Feasibility Cuts. In the standard Benders decomposition procedure, by solving the master problem, a candidate solution is obtained. If the dual of the subproblem corresponding to this candidate master problem solution is unbounded, it means that the subproblem is infeasible. In this case, this candidate solution should be removed from the search space in the next iteration of the algorithm since it cannot yield a feasible solution for the original MILP model. This is achieved by introducing a Benders feasibility cut generated by an extreme ray of the dual formulation of the subproblem. This feasibility cut ensures that the current solution will not be encountered in the next iterations. However, it is known that deriving feasibility cuts do not improve the lower bound for minimization problems, hence it is undesirable. An alternative approach can be to introduce a set of constraints to the master problem that can exclude infeasible

solutions at the beginning of the algorithm so that the boundedness of the dual of the subproblem is ensured [98].

Proposition 7.1. *For a given candidate master problem solution* $(\tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$*, the subproblem is feasible if and only if*

$$\tilde{C}_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}^k_{uqs} \qquad u \in U, q \in Q, s \in S, k \in K. \tag{7.33}$$

Proof. If condition (7.33) is satisfied, by setting $\theta^k_{uqs} = 0$ for $u \in U, q \in Q, s \in S, k \in K$, we can find a feasible solution for the subproblem, hence the subproblem is feasible. On the other hand, if the subproblem is feasible, constraints (7.30) and nonnegativity of the decision variables imply (7.33). Thus, the condition (7.33) must be satisfied when the subproblem is feasible. □

Using Proposition 7.1, we can avoid the burden of deriving feasibility cuts whenever the dual formulation of the subproblem is unbounded. Instead of generating a feasibility cut whenever the dual formulation of the subproblem is unbounded, or equivalently inequality (7.33) is not satisfied by the current master problem solution, we introduce the following set of constraints to the master problem formulation at the beginning of the decomposition process:

$$C_{qs} \geq \frac{m_q}{\alpha_q - \beta_{uqs}} Z^k_{uqs} \qquad u \in U, q \in Q, s \in S, k \in K. \tag{7.34}$$

By augmenting the master problem with (7.34), we can eliminate all potential infeasible solutions. Any candidate master problem solution $(\tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$ satisfies the feasibility condition given in inequality (7.33) and the boundedness of the dual formulation of the subproblem is guaranteed.

7.3.2.2. Simplified Subproblem and Multiple Benders Optimality Cuts. The formulation of the subproblem given in (7.27)–(7.32) can be further simplified by removing

redundant constraints and fixing some decision variables. First, it can be seen that the server placement decisions, namely $\tilde{\boldsymbol{X}}$ vector, does not affect the formulation of the subproblem, it only depends on the capacity allocation and binary task assignment decisions, $\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{Z}}$, respectively.

By inspecting the subproblem formulation, it can be noted that the constraint (7.30) corresponding to $u = \underset{u \in U}{\operatorname{argmax}} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} Z^k_{uqs} \right\}$ is tighter than the others as it has the smallest right hand side for each service type $q \in Q$ and potential server location $s \in S$ in scenario $k \in K$. Then, for a candidate master problem solution $(\tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$, we can identify the tightest constraint of type (7.30) and eliminate the remaining redundant constraints before solving the subproblem.

Moreover, it can be noted that constraints (7.31) are redundant if $\tilde{Z}^k_{uqs} = 1$ because constraints (7.28) are tighter than (7.31). Thus, these constraints can be eliminated from the subproblem formulaton if $\tilde{Z}^k_{uqs} = 1$. On the other hand, for $\tilde{Z}^k_{uqs} = 0$, constraints (7.31) imply $\theta^k_{uqs} = 0$. Then, we can set $\theta^k_{uqs} = 0$ and remove these decision variables from the subproblem formulation. Hence, the constraints (7.30) and (7.31) can be replaced with

$$F^k_{qs} \leq \tilde{C}_{qs} - \max_u \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}^k_{uqs} = 1 \right\} \qquad q \in Q, s \in S, k \in K \qquad (7.35)$$

$$\theta^k_{uqs} = 0 \qquad u \in U, q \in Q, s \in S, k \in K : \tilde{Z}^k_{uqs} = 0. \qquad (7.36)$$

Hence, we can decrease the number of decision variables and constraints in the subproblem and obtain a simpler formulation.

Another key observation is that the simplified version of the subproblem formulation can be divided into smaller problems for each service type $q \in Q$ and scenario $k \in K$ as follows:

Subproblem for $q \in Q, k \in K$ $(\tilde{C}, \tilde{Z})$:

$$\min \sum_u p_k o_q d^k_{uq}\big(1 - \sum_s \theta^k_{uqs}\big) \tag{7.37}$$

$$\text{s.t.} \sum_s \theta^k_{uqs} \leq 1 \qquad u \in U \tag{7.38}$$

$$F^k_{qs} - \sum_u m_q d^k_{uq} \theta^k_{uqs} = 0 \qquad s \in S \tag{7.39}$$

$$F^k_{qs} \leq \tilde{C}_{qs} - \max_u \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}^k_{uqs} = 1 \right\} \qquad s \in S \tag{7.40}$$

$$\theta^k_{uqs} = 0 \qquad u \in U, s \in S : \tilde{Z}^k_{uqs} = 0 \tag{7.41}$$

$$\theta^k_{uqs}, F_{qs} \geq 0 \qquad u \in U, s \in S. \tag{7.42}$$

By decomposing the subproblem into smaller problems, it becomes easier to solve each part separately. In addition, it enables to generate multiple cuts in every iteration of the algorithm using the dual information of each smaller subproblem. To apply this idea, nonnegative t^k_q variables are defined representing the unsatisfied demand penalty cost for each service type $q \in Q$ and scenario $k \in K$. Then, the t-variable in the objective function (7.22) is replaced with $\sum_q \sum_k t^k_q$. Let η_u and ϕ_s denote the optimal dual multipliers corresponding to constraints (7.38) and (7.40), respectively. Then, for each subproblem $q \in Q, k \in K$, Benders optimality cut can be written by using duality theory as follows:

$$t^k_q \geq \sum_u \eta_u + \sum_s \phi_s \Big(C_{qs} - \frac{m_q}{\alpha_q - \beta_{u'qs}} Z_{u'qsk} \Big) \tag{7.43}$$

where $u' = \underset{u \in U}{\operatorname{argmax}} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} \tilde{Z}^k_{uqs} \right\}$. Even though the the multi-cut reformulation causes a rapid increase in the number of constraints of the master problem, it can strengthen the formulation in the early iterations of the algorithm. The performance of this multi-cut reformulation is tested by computational experiments as presented in 8.2.3.

7.3.2.3. Valid Inequalities. As in the previous decomposition approach suggested for the deterministic network slicing problem, we can introduce valid inequalities to the master problem. Although valid inequalities increase the number of constraints in the master problem formulation, they may be beneficial to have better lower bounds during the initial iterations of the algorithm and accelerate the convergence by strengthening the formulation.

It can be trivially noted that for each scenario, to assign a service request from an end-user location to a potential server location, we must ensure that there is a server at any capacity level on that location. Thus, the following equality is valid for the master problem formulation:

$$Z^k_{uqs} \leq \sum_l X_{sl} \qquad u \in U, q \in Q, s \in S, k \in K. \tag{7.44}$$

Note that this equality is also valid for the deterministic equivalent MILP model.

By following a similar approach as in Proposition 6.4, a valid inequality for the multi-cut reformulation of the master problem can be written as

$$t^k_q \geq p^k o_q \sum_u d^k_{uq} - p^k \frac{o_q}{m_q} \sum_s C_{qs} \qquad q \in Q, k \in K. \tag{7.45}$$

7.3.2.4. Combinatorial Cuts. In any iteration of the decomposition algorithm, the master problem is solved to optimality and a candidate solution is obtained. For a candidate solution $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$, if the server placement and binary task assignment decisions, namely variables $\boldsymbol{X}$ and $\boldsymbol{Z}$ are fixed, and the resource allocation decisions are moved to the subproblem, we obtain the following model:

min (7.27)

s.t. (7.28), (7.29), (7.31)

$$\sum_{q} C_{qs} \leq \sum_{l} n_l \tilde{X}_{sl} \qquad s \in S \quad (7.46)$$

$$C_{qs} - F^k_{qs} \geq \max_{u} \left\{ \frac{m_q}{\alpha_q - \beta_{uqs}} : \tilde{Z}^k_{uqs} = 1 \right\} \qquad q \in Q, s \in S, k \in K \quad (7.47)$$

$$\theta^k_{uqs}, C_{qs}, F^k_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S, k \in K. \quad (7.48)$$

Note that this formulation is still an LP problem. It finds the optimal expected penalty cost of unsatisfied service demand that can be obtained by fixing $\boldsymbol{X} = \tilde{\boldsymbol{X}}$ and $\boldsymbol{Z} = \tilde{\boldsymbol{Z}}$. Let Δ be the optimal objective value of this LP problem. Based on this observation, a combinatorial cut for the multi-cut reformulation of the master problem can be written as

$$\sum_{q}\sum_{k} t^k_q \geq \Delta \Bigg[1 - \sum_{(s,l):\tilde{X}_{sl}=0} X_{sl} + \sum_{(s,l):\tilde{X}_{sl}=1} \big(1 - X_{sl}\big) + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=0} Z^k_{uqs} + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=1} \big(1 - Z^k_{uqs}\big) \Bigg]. \quad (7.49)$$

Note that the inequality (7.49) reduces to $\sum_{q}\sum_{k} t^k_q \geq \Delta$ for the master problem solutions with $\boldsymbol{X} = \tilde{\boldsymbol{X}}$ and $\boldsymbol{Z} = \tilde{\boldsymbol{Z}}$, and is redundant for the remaining feasible solutions. Thus, inequality (7.49) is valid and can be added to the master problem in each iteration of the algorithm.

For a candidate master problem solution, assume that $\tilde{Z}^k_{uqs} = 1$ for some $u \in U, q \in Q, s \in S, k \in K$. In addition, assume that we have $\theta^{k*}_{uqs} = 0$ in the optimal solution of the LP formulation given above. Then, if we set $\tilde{Z}^k_{uqs} = 0$, the current optimal solution of LP model still remains feasible. Thus, the optimal objective value can stay the same or improve, but cannot get worse. Using this observation, a stronger

version of the combinatorial cut (7.49) can be written as

$$\sum_q \sum_k t_q^k \geq \Delta \Bigg[1 - \sum_{(s,l):\tilde{X}_{sl}=0} X_{sl} + \sum_{(s,l):\tilde{X}_{sl}=1} \big(1 - X_{sl}\big) + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=0} Z^k_{uqs} + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=1,\theta^{k*}_{uqs}>0} \big(1 - Z^k_{uqs}\big) \Bigg]. \tag{7.50}$$

To apply this idea in the decomposition approach, the master problem is solved and a candidate solution $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{C}}, \tilde{\boldsymbol{Z}})$ is obtained at each iteration of the algorithm. Then, the above LP model is solved. In addition to Benders cuts, inequality (7.50) is generated and added to the master problem to be resolved in the next iteration. Although this strategy requires to solve an additional LP model in each iteration of the algorithm, it may be helpful to strengthen the master problem formulation and obtain better bounds.

7.4. A Variant of the Problem with Integer Capacity Levels

In this section, we examine a more realistic counterpart of the stochastic network slicing problem. In this variant, we assume that the total number of cores in a computational resource is partitioned among different service types. Hence, the capacity allocation of a server to different service instances can only be in discrete amounts. To formulate this problem, let W_{iqs} be a binary variable which takes value 1 if i cores are allocated to service type q at potential server location s, and 0 otherwise. Also, let the parameter B denote the processing capacity of a single core and χ_l denote the number of cores on a server at capacity level l. Using this notation, the first-stage problem can be written as

$$\min \sum_s \sum_l a_l X_{sl} + \gamma \mathbb{E}[Q(\boldsymbol{W}, \boldsymbol{\xi})] \tag{7.51}$$

s.t. (7.2)

$$\sum_q \sum_i i W_{iqs} \leq \sum_l \chi_l X_{sl} \qquad s \in S \tag{7.52}$$

$$X_{sl}, W_{iqs} \in \{0,1\} \qquad s \in S, l \in L, i \in I. \tag{7.53}$$

Then, the second-stage problem (7.6)–(7.12) can also be transformed by replacing constraints (7.9) with

$$F_{qs} \leq B \sum_{i} iW_{iqs} - \frac{m_q}{\alpha_q - \beta_{uqs}} Z_{uqs} \qquad u \in U, q \in Q, s \in S. \tag{7.54}$$

Hence, we can formulate a two-stage stochastic programming model for the case where the capacity allocation of a server to different service instances can only occur at discrete levels. By defining a set of scenarios for the random data vector $\boldsymbol{\xi}$, a deterministic equivalent MILP model of the two-stage stochastic programming model is obtained as follows:

$$\min \ (7.13) \tag{7.55}$$

$$\text{s.t. } (7.14), (7.16), (7.17), (7.19), (7.52), (7.54)$$

$$\theta^k_{uqs}, F^k_{qs} \geq 0 \qquad u \in U, q \in Q, s \in S, k \in K \tag{7.56}$$

$$X_{sl}, Z^k_{uqs}, W_{iqs} \in \{0,1\} \qquad u \in U, q \in Q, s \in S, k \in K, l \in L, i \in I. \tag{7.57}$$

Similar to its continuous capacity counterpart, a Benders decomposition approach can be applied to this MILP formulation as described in Section 7.3. Note that the valid inequality (7.44) is still valid for this formulation. However, the valid inequality (7.45) should be replaced with

$$t^k_q \geq p^k o_q \sum_{u} d^k_{uq} - p^k \frac{o_q}{m_q} B \sum_{s} \sum_{i} iW_{iqs} \qquad q \in Q, k \in K. \tag{7.58}$$

In addition to feasibility and optimality cuts, and valid inequalities introduced in Section 7.3.2, we can also derive a combinatorial cut for a given master problem solution $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{W}}, \tilde{\boldsymbol{Z}})$. By following a similar approach as in inequality (7.50), in any iteration of the algorithm, let Δ be the optimal objective value of the subproblem

corresponding to the current master problem solution. Also, let $\tilde{Z}^k_{uqs} = 1$ and $\theta^{k*}_{uqs} = 0$ in the optimal solution of the subproblem for some $u \in U, q \in Q, s \in S, k \in K$. Then, a strengthened combinatorial cut for the multi-cut reformulation of the master problem can be generated in the following form:

$$\begin{aligned}\sum_q \sum_k t^k_q \geq \Delta \Bigg[1 - \sum_{(s,l):\tilde{X}_{sl}=0} X_{sl} + \sum_{(s,l):\tilde{X}_{sl}=1} \big(1 - X_{sl}\big) \\ + \sum_{(i,q,s):\tilde{W}_{iqs}=0} W_{iqs} + \sum_{(i,q,s,):\tilde{W}_{i,q,s}=1} \big(1 - W_{iqs}\big) \\ + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=0} Z^k_{uqs} + \sum_{(u,q,s,k):\tilde{Z}^k_{uqs}=1, \theta^{k*}_{uqs}>0} \big(1 - Z^k_{uqs}\big) \Bigg].\end{aligned} \tag{7.59}$$

It can be observed that the inequality (7.59) reduces to $\sum_q \sum_k t^k_q \geq \Delta$ for the current master problem solution $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{W}}, \tilde{\boldsymbol{Z}})$. We can generate inequality (7.59) along with Benders cuts and add to the master problem in each iteration to strengthen the formulation.

8. COMPUTATIONAL EXPERIMENTS

We carry out a series of computational experiments on randomly generated test instances to assess the performance of the proposed solution methods and observe their robustness and scalability issues. In this chapter, we first outline the generation of test instances used in the experimental design and the test environment. Then, we present the details of the numerical results and the discussion on the performance of alternative solution approaches for each problem.

8.1. Generation of Test Instances

To simulate realistic next-generation network environments, the use-cases envisioned by the European Telecommunications Standards Institute (ETSI) and 5G-PPP [99, 100], which define the specification set for the next-generation cellular networks, are utilized. The performances of the proposed solution approaches are assessed on network topologies having number of vertices ranging from 100 to 1000. These topologies are randomly generated as connected networks where the degree of each vertex is set between 2 and 4 to mimic the characteristics of the real-life network architectures [92]. We use a Python package called NetworkX, which is used for creation of large complex networks and provides standard graph algorithms [101]. For each problem instance, the locations of end-users and potential servers are selected randomly among vertices of the network. For computational resources, three different capacity levels are defined. For each problem size, five random instances are generated and some statistics are reported.

The details of the parameters used to generate test instances are summarized in Table 8.1. The capacities of vertices (ψ_v) and links (ω_e) are set to 10k in Mbps. For computational resources, the capital cost (a_l), the processing capacity (n_l), the number of cores (χ_l), and the maximum number of service instances to be hosted (g_l) of a server at different capacity levels are presented in the table. The processing ca-

pacity of a single core is set to 5k in MIPS. Services with different characteristics and requirements are generated to reflect the next-generation use-cases. The corresponding computational and networking resource requirements (m_q, h_q^{req} and h_q^{res}), unit revenue (r_q), unit penalty cost (o_q), and the maximum acceptable delay limit (α_q) of these services are generated using a uniform distribution. Similarly, the total number of service requests at each end-user location for each service type (d_{uq}) is assumed to follow a uniform distribution. The maximum allowed utilization for computational and networking resources (ρ and σ) are set as 100% and 95%, respectively. For stochastic problems, a set of scenarios are specified with equal probability.

Table 8.1. The parameters used to generate test instances.

Parameter	Value
ψ_v	$10k$ (Mbps)
ω_e	$10k$ (Mbps)
a_l	$\{3k, 5k, 12k\}$
n_l	$\{10k, 20k, 50k\}$ (MIPS)
χ_l	$\{2, 4, 10\}$
g_l	$\{2, 4, 6\}$
B	$5k$ (MIPS)
m_q	$U(100, 200)$ (MI)
h_q^{req}	$U(1, 10)$ (Mbits)
h_q^{res}	$U(1, 10)$ (Mbits)
r_q	$U(1, 5)$
o_q	$U(1, 5)$
α_q	$U(0.5, 1.5)$ (s)
d_{uq}	$U(1, 10)$
ρ	100%
σ	95%

All mathematical programming formulations are implemented in C++ programming language, and solved using Gurobi 9.0.1, IBM CPLEX 12.9 and 20.1. The standard branch-and-bound schemes of optimization solvers are utilized to solve the MILP formulations. We set a time limit of one hour and collect some statistics over five instances for each problem size.

In the deterministic and stochastic network slicing problems, while implementing Benders decomposition, instead of repeatedly generating a branch-and-bound tree in each iteration, we use a single branch-and-bound tree that is tightened as necessary. In the traditional Benders decomposition algorithm, the master problem is solved to optimality by building a new branch-and-bound tree at every iteration. However, this causes a significant computational effort to revisit a candidate solution that has already been discarded in the early iterations. In our approach, instead of repeatedly solving the master problem and generating a branch-and-bound tree at each iteration, we build a single branch-and-bound tree and generate cuts violated by the current integer solution encountered inside the tree while solving the master problem. In the computational tests, this approach, which is often referred to as branch-and-Benders-cut [98], consistently outperformed the approach of reoptimizing the master problem at each iteration. The branch-and-Benders-cut method is implemented by interrupting the standard branch-and-bound solution process and adding cuts to the master problem using the generic callback procedures available in the Concert technology library of CPLEX.

Furthermore, CPLEX provides an automated Benders decomposition feature which decomposes the model into a master problem containing only integer variables and (possibly multiple) subproblems with continuous linear variables. We use the automated Benders decomposition feature of CPLEX by setting "Benders strategy" parameter as three. All the remaining parameters of optimization solvers are kept at their default values.

8.2. Numerical Results

The performances of the proposed solution methods are compared and the results are summarized in the following subsections. Each row of the tables shows the average values of five randomly generated instances. The columns in the tables are explained as follows:

- n: the number of instances for which the corresponding method can find a feasible solution within the allowed time limit
- Time: the average amount of time in seconds spent by each method
- Gap: the average percent optimality gap over five instances computed as $100 \times \frac{Z_{UB}-Z_{LB}}{Z_{LB}}$ for the problems with maximization objective and $100 \times \frac{Z_{UB}-Z_{LB}}{Z_{UB}}$ for the problems with minimization objective, where Z_{UB} and Z_{LB} denote, respectively, the best upper bound and the best lower bound for the optimal objective function value of the corresponding method
- s: the standard deviation of the percent optimality gap for the instances for which the corresponding method is able to find a feasible solution
- LP Dev: the average percent deviation between the best objective function value of the MILP model (Z_{MILP}) and the objective function value of the LP relaxation solution (Z_{LP}), measured as $100 \times \frac{Z_{LP}-Z_{MILP}}{Z_{MILP}}$
- LR Dev: the average percent deviation between Z_{MILP} and the objective function value of the Lagrangian relaxation solution (Z_{LR}), measured as $100 \times \frac{Z_{LR}-Z_{MILP}}{Z_{MILP}}$
- LB Dev: the average percent deviation between Z_{MILP} and Z_{LB}, measured as $100 \times \frac{Z_{LB}-Z_{MILP}}{Z_{MILP}}$
- UB Dev: the average percent deviation between Z_{MILP} and Z_{UB}, measured as $100 \times \frac{Z_{UB}-Z_{MILP}}{Z_{MILP}}$.

The last row of each table provides the total number of instances for which the corresponding method can find a feasible solution within the given time limit and the average values for the remaining quantities.

8.2.1. Computation Architecture Design Problem

In the computation architecture design problem, the size of the problem is determined by the number of vertices $|V|$, the number of end-user locations $|U|$, the number of potential server locations $|S|$, and budget level b. Five different problem instances are generated for each problem size. In order to evaluate the behavior of differentiated service types and observe their performance, ten different service types are generated where the characteristics and the requirements are adapted to represent the real-world settings. In addition, four different budget levels are generated depending on the number of potential server locations as shown in Table 8.2.

Table 8.2. The budget levels used in the test instances.

Budget level	Value
High (H)	$\|S\| \times 8000$
Mid-High (MH)	$\|S\| \times 7000$
Mid-Low (ML)	$\|S\| \times 6000$
Low (L)	$\|S\| \times 5000$

As discussed in Chapter 3, there is no other study in the literature that combines the three phases of the computation architecture design problem and proposes an integrated solution approach. Besides, the approaches proposed by our study are capable of obtaining optimal or near-optimal solutions with small optimality gaps. Hence, in this section, only the performances of the MILP model and the Lagrangian heuristic approach are compared for various network sizes. All mathematical programming formulations are solved using Gurobi 9.0.1 running on a computer with Intel Xeon E5-2690 2.60 GHz CPU and 64 GB main memory. The results of the experiments are presented in Table 8.3.

For the small topology with 100 vertices, it can be observed that the MILP model can provide feasible solutions with small optimality gaps for all of the instances

within the given time limit. With the increase in the number of end-user locations and potential server locations, the quality of solutions deteriorates slightly, but the solver can still find high-quality solutions. For some instances, the solver can guarantee the solution's optimality before the allowed time limit; hence "Time" column can be less than one hour. Small LP relaxation gaps shown in the "LP Dev" column reveal that the MILP formulation is tight and provides good LP relaxation bounds. When the number of end-user locations and potential server locations is kept constant, the increase in the topology size does not significantly affect the optimality gap for the MILP model. For instance, the average optimality gap for the instances with 500 vertices does not differ much from the instances with 1,000 vertices in which case there are 100 end-user and potential server locations.

However, the number of end-user locations and potential server locations affects the complexity of the model since the number of decision variables and the number of constraints mainly depend on the cardinality of the end-user and potential server locations. Therefore, for the same topology, as the number of end-user locations and potential server locations increases, the best feasible solutions are obtained with a higher optimality gap at the end of the time limit. Moreover, the MILP model fails to find a feasible solution within the given time limit for the instances with a large number of end-user locations and potential server locations. For example, the MILP model cannot find a feasible solution for most instances in which there are 400 end-user and potential server locations. For those cases, the optimality gaps and LP relaxation deviations are calculated using only the instances where the MILP model can find a feasible solution. In addition, the LP relaxation gap increases with the increase in the number of end-user and potential server locations. It shows that either the best feasible solution obtained at the end of the time limit is inferior or the upper bound obtained by the LP relaxation is not tight enough.

Table 8.3. Comparison of solution methods for computation architecture design problem.

		MILP model				Lagrangian relaxation			
$\|V\|, \|U\|, \|S\|$	b	n	Gap	Time	LP Dev	n	Gap	LR Dev	Time
(100, 20, 20)	H	5	0.02	2304.4	0.45	5	1.87	-1.81	3600.2
	MH	5	0.23	3237.2	1.17	5	2.25	-1.98	3600.2
	ML	5	0.25	2903.6	0.74	5	3.63	-3.25	3600.1
	L	5	0.23	1591.4	1.38	5	4.19	-3.78	3600.1
(100, 40, 40)	H	5	0.17	2169.6	0.34	5	1.36	-1.16	3601.3
	MH	5	0.46	3600.0	0.71	5	2.73	-2.20	3609.4
	ML	5	0.51	3087.6	0.81	5	3.15	-2.55	3601.4
	L	5	0.58	3011.7	1.44	5	3.50	-2.81	3602.4
(100, 80, 80)	H	5	0.56	2961.1	0.59	5	1.15	-0.58	3608.5
	MH	5	1.02	3600.1	1.19	5	1.80	-0.76	3605.2
	ML	5	2.87	3600.3	3.09	5	2.97	-0.10	3612.5
	L	5	1.32	3600.1	1.71	5	3.33	-1.94	3609.9
(500, 100, 100)	H	5	0.19	3024.3	0.43	5	3.10	-2.77	3603.2
	MH	5	1.12	3600.1	1.42	5	3.52	-2.31	3604.1
	ML	5	0.99	3191.3	1.36	5	5.84	-4.56	3612.6
	L	5	0.97	3600.1	1.60	5	4.50	-3.37	3603.1
(500, 200, 200)	H	5	11.49	3600.6	11.51	5	3.02	8.21	3624.9
	MH	5	11.15	3600.6	11.21	5	4.30	6.57	3639.1
	ML	4	2.50	3600.5	2.60	5	5.16	-3.10	3643.3
	L	4	7.79	3600.6	8.03	5	6.74	0.34	3637.3
(500, 400, 400)	H	1	9.43	3602.6	9.43	5	3.44	5.79	3772.8
	MH	1	9.70	3602.1	9.71	5	4.88	4.60	3829.9
	ML	1	11.88	3602.4	11.95	5	5.40	6.15	3800.5
	L	1	11.29	3602.2	11.53	5	6.46	4.54	3814.6
(1000, 100, 100)	H	5	0.30	2186.7	1.17	5	3.80	-3.34	3604.1
	MH	5	0.92	3600.1	1.92	5	5.33	-4.17	3602.2
	ML	5	0.82	3600.1	1.72	5	5.96	-4.80	3602.5
	L	5	0.87	3600.2	2.05	5	6.00	-4.77	3602.4
(1000, 200, 200)	H	5	2.67	3355.0	2.82	5	5.16	-2.30	3631.0
	MH	5	10.06	3600.5	10.27	5	6.90	2.98	3641.3
	ML	5	5.90	3600.5	6.09	5	6.22	-0.22	3610.3
	L	5	5.83	3600.5	6.15	5	8.19	-2.09	3613.1
(1000, 400, 400)	H	1	8.13	3602.5	8.13	5	6.65	1.39	3714.1
	MH	2	5.87	3602.2	5.89	5	6.21	0.40	3730.4
	ML	2	1.87	3602.0	1.94	5	2.67	-0.78	3768.4
	L	2	17.55	3601.9	17.74	5	8.22	8.46	3740.3

The performance of the Lagrangian relaxation-based heuristic algorithm is tested through the same instances. Although a time limit of one hour is set for the algorithm, the total time spent can be slightly larger than one hour since an optimization problem is solved at each iteration of the algorithm. During run time, the algorithm continues to iterate until the allowed time expires, and tries to improve the objective value in each iteration. In other words, numerous iterations, where each represents a separate model execution, are completed by the Lagrangian relaxation-based heuristic, and the time-complexity issue is successfully addressed. It is observed that the Lagrangian relaxation-based algorithm can find a feasible solution for all instances within one hour of a time limit. The average optimality gap becomes slightly higher as the topology size increases. However, the number of end-user and potential server locations, which is the main factor of the complexity of the MILP model, does not have a significant effect on the optimality gap.

For analyzing the performance of both methods, the best feasible solution of the Lagrangian heuristic obtained at the end of the time limit is compared against the best feasible solution found for the MILP model. The average deviation between their objective function values is reported as "LR Dev" column in Table 8.3. Even though the average optimality gap obtained by the Lagrangian relaxation-based heuristic algorithm is slightly higher than the MILP model for smaller cases, the difference is not significant. The performance of the MILP model deteriorates with the increase in the number of end-user and potential server locations. On the other hand, the Lagrangian relaxation-based heuristic algorithm performs better than the MILP model and finds better feasible solutions for large instances. For example, for the (500, 400, 400) case, the MILP model can find a feasible solution for only one out of five instances at every budget level. In addition, the optimality gap for those instances is high. On the other hand, when the Lagrangian heuristic algorithm is applied to those instances, it can find feasible solutions for all of the instances, and the average optimality gap is lower than that of the MILP model. It can be concluded that the Lagrangian relaxation approach is more robust for the cases studied in terms of the solution quality for larger instances, whereas the MILP model has a lower average optimality gap for small instances.

The budget level has some impact on the performance of the MILP model. The average optimality gap is usually lower for the cases with a high budget level where the feasible region is broader. However, there is no significant difference between mid-high, mid-low, and low budget levels. Some cases do not match precisely with this inference. For instance, the higher budget cases result in a higher average optimality gap for the topology with 500 vertices where 200 end-user and potential server locations are available. The main reason is that in lower-budget cases, the MILP model remains incapable of finding a feasible solution for one of the randomly generated instances. Since the optimality gap cannot be calculated for such instances, the statistics are reported only for instances where a feasible solution is found.

In addition to analyzing the proposed solutions and their comparisons, the service type behaviors are also inspected. As an exemplary case, the satisfaction ratio of each service type for the (1000, 200, 200) instance is depicted in Figure 8.1, concerning different budget levels. The satisfaction ratio is calculated as the ratio of successfully handled requests to the total number of requests for a particular service type. In general, it can be stated that the increase in the budget results in a higher service satisfaction ratio for each service type. Besides, in lower budget cases, it is observed that some services have a lower satisfaction ratio than the others, so fairness among service types cannot be achieved. The diversity in the satisfaction ratio for different service types decreases as the budget increases. It can be concluded that the model gives higher priority to the services with higher revenue and lower resource requirement due to the limited processing power in lower budget scenarios. As the total capacity increases in the higher budget scenarios, more service requests can be handled; thereby, prioritization between services is eliminated.

For example, let us consider service type 7, one of the services with the lowest satisfaction ratio in the lowest budget case. However, we see that this particular service type has the highest satisfaction ratio in the highest budget case. This situation can be explained as follows: Although this type of service yields high revenue, it is not prioritized by the optimization model in low-budget cases because of its high resource

capacity demand and stringent latency requirement. Therefore, more resource capacity is allocated to the service types demanding less resources and having relatively lower unit revenue. As a result, the optimization model can find a better solution by obtaining higher revenue through handling more task offload operations of these service types. However, as more resource capacity becomes available with the increasing budget, this service type is prioritized due to its high revenue despite its requirements.

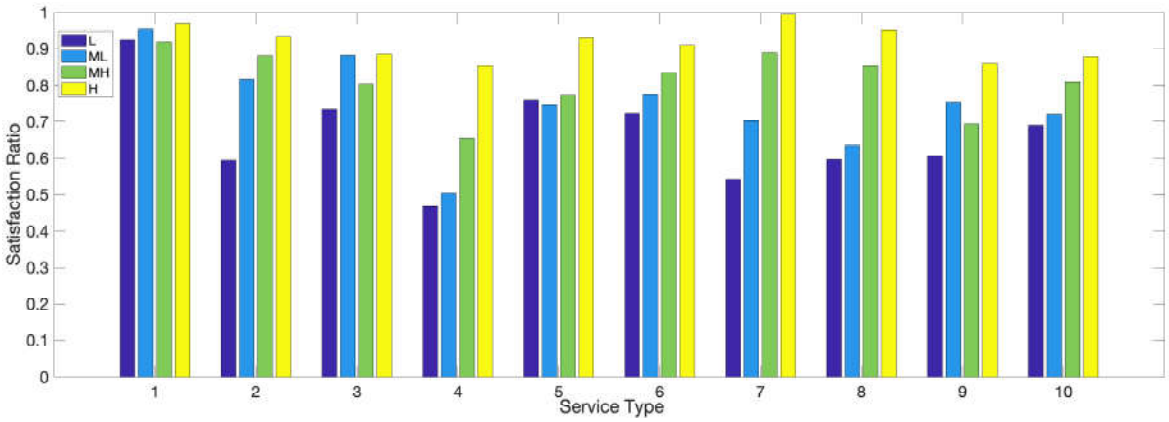

Figure 8.1. Satisfaction ratio of different service types with respect to various budget limitations for (1000, 200, 200) instance.

Finally, the impact of optimizing the server placement and service deployment decisions is investigated on instances with 100 vertices and the results of the experiments are presented in Table 8.4. In this case, a set of servers with arbitrary capacity levels is placed such that the total capital cost for server placements does not exceed the given budget of the operator. Then, based on the maximum number of service instance restriction on each server capacity level, service deployments are arbitrarily determined. Let $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}})$ denote the fixed server placement and service deployment decisions. Based on these decisions, the optimal task assignment is obtained by solving the optimization problem (5.15)–(5.24). The "Impact" column shows the average percent deviation of five instances between Z_{LB} and $Z_{FixedSS}$, where $Z_{FixedSS}$ denotes the best objective value obtained for the task assignments corresponding to the fixed server placement and service deployment decisions, measured as $\frac{Z_{FixedSS}-Z_{LB}}{Z_{LB}}$.

It can be observed that optimizing the server placement and service deployment decisions has a significant impact on the revenue gain for all instances. It is even more critical at low budget levels since the average performance of arbitrary server placement and service deployment strategy is very poor when resources are scarce. The impact becomes smaller with the increasing number of end-user and potential server locations for the same topology. In this case, end-users and servers reside at physically closer locations, and non-optimal server placement decisions can be partially compensated by assigning the service requests to another nearby server.

Table 8.4. Effect of server placement and server deployment decisions for computation architecture design problem.

$\|V\|, \|U\|, \|S\|$	b	Impact (%)
(100, 20, 20)	H	-11.94
	MH	-13.96
	ML	-17.89
	L	-18.95
(100, 40, 40)	H	-6.37
	MH	-7.68
	ML	-8.67
	L	-12.00
(100, 80, 80)	H	-3.82
	MH	-5.29
	ML	-4.68
	L	-7.62
Average		-9.90

For the stochastic variant of the problem, we implement the SAA algorithm described in Section 5.4.2. The SAA method is based on generating random samples and approximating the expected value function by the corresponding sample average function. To implement the SAA algorithm, we set $M = 20$, $N = 20$, $N' = 50$, and

$N'' = 1000$. In our preliminary analysis, we observe that these parameters are sufficient to obtain good results for the test instances. The proposed method is implemented in C++ with IBM CPLEX Optimization solver 12.9 running on a computer with Intel Xeon E5-2690 2.6 GHz CPU and 64GB main memory.

To evaluate the performance of the proposed method, three different topology instances from Topology Zoo [102] are utilized with varying number of vertices, number of end-user and potential server locations. For each problem, three service types are generated along with their diversified characteristics and requirements. Similarly, we allow three capacity levels for computational resources. For each topology, instances with low and high number of service requests are generated. The number of service requests from each end-user location for any service type follows a uniform distribution $U(10, 50)$ and $U(10, 100)$ for low and high service requests cases, respectively. Finally, we use three different budget levels to see their effect on the optimal objective value, optimality gap, and solution time. Note that a single problem instance is generated for each problem size.

The results of the computational study are provided in Table 8.5. First of all, the overall performance of the method seems to be satisfactory since small optimality gaps are obtained. It can be observed that the SAA method requires more time as the number of vertices on the network increases. For each problem size, as the expected number of service requests increases, the LB and UB for the optimal objective value increase. However, for the instances with high number of the service requests, the standard deviation of the optimality gap is also large since the uncertainty in the input parameters gets higher. In addition, the increase in the budget that can be spent on server placement decisions also increases the LB and UB for the optimal objective value. Since the feasible region of the problem becomes tighter at low budget values, the SAA method takes longer as the budget decreases.

Table 8.5. The results of the SAA method for the stochastic variant of the computation architecture design problem.

$(\|V\|, \|U\|, \|S\|)$	Service Requests	b	Z_{LB}	Z_{UB}	Gap	s	Time
(11, 10, 10)	Low	70k	2615.7	2628.6	0.49	13.3	202.8
		80k	2647.1	2658.3	0.42	13.8	157.9
		90k	2652.1	2664.8	0.48	14.1	133.6
	High	70k	3526.4	3586.8	1.71	44.5	156.4
		80k	3880.3	3939.1	1.51	43.5	155.5
		90k	4181.2	4237.7	1.35	39.4	142.1
(18, 15, 15)	Low	80k	3623.0	3651.5	0.79	24.7	1542.2
		100k	3907.6	3924.9	0.44	17.5	530.3
		120k	3973.8	3989.3	0.39	18.0	354.9
	High	80k	4357.1	4361.6	0.11	45.4	1357.8
		100k	5203.5	5214.1	0.20	50.9	403.7
		120k	5926.5	5962.3	0.60	50.4	409.1
(25, 20, 20)	Low	100k	4705.0	4710.8	0.12	34.0	4711.9
		120k	5085.8	5108.6	0.45	22.0	1804.5
		150k	5282.6	5318.3	0.68	15.7	727.4
	High	100k	5408.0	5491.2	1.54	60.9	6662.2
		120k	6321.1	6376.4	0.87	69.6	1337.5
		150k	7504.3	7561.3	0.76	72.5	848.0

8.2.2. Deterministic Network Slicing Problem

In the deterministic network slicing problem, the size of the problem is determined by the number of services $|Q|$, the number of end-user locations $|U|$, the number of potential server locations $|S|$, and budget level b. All mathematical programming formulations are solved using IBM CPLEX Optimization solver 12.9 running on a computer with Intel Xeon E5-2690 3.10 GHz CPU and 64 GB main memory using a single thread. The results of the computational study are presented in the following

tables. We set a time limit of one hour for each method. The problem instances are large enough and thus none of the methods can prove the optimality of the best solution within this time limit. Instead, each method utilizes the entire 3600 seconds and we report the optimality gap reached at the end of one hour. Thus, we omit the "Time" column in the following tables.

In Table 8.6, the performances of the MILP formulation and that of the first decomposition approach, referred to as BD1, augmented with valid inequality (6.38) are investigated on a rather small set of instances with 500 vertices. In addition, we investigate the solution quality obtained by the automated Benders decomposition feature of CPLEX optimization solver. In the table, we indicate by "–" that the corresponding method is unable to find a feasible solution for any instance within the allowed time limit. As we discuss in Chapter 3, the existing studies do not combine the server placement, resource allocation, and task assignment decisions into a single integrated model. Hence, only the performances of the MILP model and the decomposition approaches are compared for different problem sizes.

It can be observed that the MILP model can generally provide feasible solutions with small optimality gaps. The standard deviation values for the optimality gap are also low, which indicates the robustness of the solutions obtained by this method. Since the number of decision variables and constraints is $O(|U||Q||S||L|)$, the solution quality is affected by the number of end-user locations, services, and potential server locations. Moreover, the budget level determines the size of the feasible region, hence it also affects the solution quality. Out of 160 instances, the MILP formulation is unable to find a feasible solution for 17 instances within the given time limit.

The columns entitled "BD1 + (6.38)" show the results for the first decomposition method where the master problem is augmented with valid inequality (6.38). In our preliminary analysis, we observed that adding valid inequality (6.38) into the master problem at the beginning of the algorithm yields a significant improvement in the decomposition procedure. Thus, we only present the results for the augmented formu-

lation. In addition, instead of LP duality-based feasibility cuts, we use combinatorial Benders cuts given in (6.36).

First, it can be seen that our decomposition procedure can find a feasible solution for all of the instances. However, the average optimality gap is larger compared to the MILP formulation. In addition, the standard deviation for the optimality gap is higher. It means that either the best feasible solution obtained at the end of the time limit, i.e. the best lower bound, is inferior, or the upper bound obtained is not tight enough. To demonstrate the quality of our best feasible solutions compared to the MILP formulation, we also report "LB Dev" column. It shows that the decomposition method is generally able to find good-quality feasible solutions with lower bounds close to that of the MILP model. Thus, large optimality gaps can be explained by the information loss associated with the decision variables in the subproblem, which results in poor upper bound values. Hence, the overall performance of the first decomposition method in terms of solution quality is worse than solving the MILP formulation by CPLEX, but it is able to find feasible solutions for all of the instances.

Similarly, we compare the performance of the automated Benders decomposition feature of CPLEX. The results of this experiment are presented in the columns "MILP + (6.38) Automated Benders" in Table 8.6. As in the first decomposition approach, we observe that the usage of valid inequality (6.38) has a positive effect on the solution quality. Therefore, we only present the results of the improved method. It can be observed that the number of instances for which the automated Benders decomposition method is able to find a feasible solution is quite low. In addition, there is no clear improvement in terms of the solution quality between BD1 and automated Benders decomposition. Thus, the usage of automated Benders decomposition of CPLEX is not justified for the problem under consideration.

Table 8.6. Performance comparison of MILP formulation and BD1 for deterministic network slicing problem.

				MILP			BD1 + (6.38)				MILP + (6.38) Automated Benders			
$\|Q\|$	$\|U\|$	$\|S\|$	b	n	Gap	s	n	Gap	s	LB Dev	n	Gap	s	LB Dev
10	50	25	25k	5	0.27	0.32	5	1.53	0.90	-0.37	5	2.10	0.78	-0.40
10	50	25	50k	5	2.37	0.37	5	7.49	5.35	-1.11	4	4.27	0.55	-0.84
10	50	25	75k	5	0.21	0.10	5	1.40	0.42	-0.99	5	2.97	0.68	-0.74
10	50	25	100k	5	0.14	0.20	5	0.82	0.53	-0.38	5	2.93	2.46	-0.27
10	50	50	25k	5	0.78	1.15	5	73.78	69.34	-3.05	5	320.77	191.21	-8.84
10	50	50	50k	5	2.69	0.29	5	7.39	9.49	-1.03	5	2.21	1.09	-0.71
10	50	50	75k	5	0.42	0.10	5	1.29	0.34	-0.91	4	1.04	0.24	-0.65
10	50	50	100k	4	0.32	0.63	5	6.40	5.03	-0.16	1	1.40	-	-0.11
10	100	25	25k	5	0.24	0.20	5	1.54	1.68	-0.26	4	2.57	0.49	-0.09
10	100	25	50k	5	2.80	0.25	5	5.16	4.13	-0.72	4	3.86	0.18	-0.58
10	100	25	75k	5	0.48	0.24	5	1.27	0.30	-0.69	0	–	–	–
10	100	25	100k	5	1.22	0.21	5	2.73	2.26	-0.85	1	2.95	–	-0.87
10	100	50	25k	5	0.16	0.06	5	49.12	50.48	-2.60	5	899.24	149.75	-24.54
10	100	50	50k	5	2.84	0.64	5	3.30	0.95	-0.75	5	4.42	1.17	-0.82
10	100	50	75k	4	6.65	6.73	5	1.39	0.32	5.47	3	1.52	0.20	7.13
10	100	50	100k	5	19.44	7.99	5	7.15	9.14	18.96	0	–	–	–
20	50	25	25k	5	0.14	0.10	5	1.68	1.17	-0.45	0	–	–	–
20	50	25	50k	5	2.68	0.23	5	3.03	0.85	-1.08	2	4.61	0.18	-1.29
20	50	25	75k	5	0.28	0.17	5	2.21	0.30	-1.59	0	–	–	–
20	50	25	100k	5	1.23	0.10	5	3.04	0.98	-1.57	0	–	–	–
20	50	50	25k	5	2.50	0.33	5	174.77	154.85	-11.83	4	1090.51	148.72	-38.21
20	50	50	50k	5	2.81	0.47	5	23.08	33.20	-2.42	4	900.26	273.22	-42.03
20	50	50	75k	5	4.28	5.55	5	2.33	0.34	0.24	2	117.54	157.91	-11.33
20	50	50	100k	4	3.27	1.34	5	5.94	3.91	-0.37	0	–	–	–
20	100	25	25k	5	0.10	0.05	5	2.99	1.04	-0.35	0	–	–	–
20	100	25	50k	5	3.21	0.23	5	3.65	0.92	-0.68	0	–	–	–
20	100	25	75k	5	0.79	0.37	5	1.44	0.20	-0.94	0	–	–	-
20	100	25	100k	5	1.40	0.07	5	2.22	0.32	-1.22	0	–	–	–
20	100	50	25k	5	2.58	0.20	5	524.99	281.85	-13.41	0	–	–	–
20	100	50	50k	1	2.41	–	5	71.76	150.05	-0.38	0	–	–	–
20	100	50	75k	0	–	–	5	1.51	0.21	–	0	–	–	–
20	100	50	100k	0	–	–	5	14.70	13.06	–	0	–	–	–
				143	2.26	0.94	160	31.60	25.12	-0.91	68	212.30	55.09	-7.81

In Table 8.7, we measure the performance of the second decomposition approach, referred to as BD2, on the same test instances. According to our computational tests, the decomposition approach with multi-cut reformulation of the master problem always provides superior results compared to its single-cut version by quickly tightening the master problem. Thus, the results of the single-cut approach is omitted.

It can be noted that, similar to BD1, we can find a feasible solution for all instances within the allowed time limit using the second decomposition. However, the method itself without the addition of any valid inequalities results in very large optimality gaps. On the other hand, when the best objective value of the MILP model

and that of the decomposition approach are compared on the basis of "LB Dev" values, it can be seen that there is no significant difference in terms of the best feasible solution quality between these two methods. The reason for the large average optimality gap is that in the Benders decomposition method, the master problem loses all the information associated with the decision variables in the subproblem, hence results in poor upper bound for the optimal objective value, which can be improved by integrating valid inequalities and strengthening the master problem formulation.

The effect of valid inequalities given in Section 6.3 on the second decomposition approach is also investigated and the results of this experiment are summarized in Table 8.7. It can be noted that augmenting the master problem with valid inequalities (6.38) or (6.68) helps in improving the solution quality by reducing the average and the standard deviation of the optimality gap. It also allows us to obtain slightly better feasible solutions. In general, valid inequality (6.68) provides better performance than (6.38) since it has lower average optimality gap with smaller standard deviation. In fact, it slightly outperforms the MILP model as it can solve all of the instances and provide robust solutions with smaller average optimality gaps. For some instances, it can also provide better feasible solutions in terms of the objective function value than the MILP model.

When both valid inequalities (6.38) and (6.68) are integrated into the master problem at the same time, the number of constraints increases significantly. However, there is no clear impact on the performance in terms of the average optimality gap. In fact, there is a trade-off between strengthening the master problem formulation and increasing the number of constraints in the master problem. The performance of the decomposition method with both valid inequalities may get worse when the size of the problem increases.

Table 8.7. The performance of BD2 and the effect of valid inequalities for deterministic network slicing problem.

				BD2				BD2 + (6.38)				BD2 + (6.68)				BD2 + (6.38) + (6.68)			
$\|Q\|$	$\|U\|$	$\|S\|$	b	n	Gap	s	LB Dev	n	Gap	s	LB Dev	n	Gap	s	LB Dev	n	Gap	s	LB Dev
10	50	25	25k	5	116.48	60.19	-0.51	5	13.30	17.70	-0.29	5	2.76	0.45	-0.09	5	2.48	0.23	-0.02
10	50	25	50k	5	62.37	16.30	-1.00	5	32.13	14.05	-0.78	5	3.24	0.35	-0.41	5	2.87	0.60	-0.28
10	50	25	75k	5	3.91	1.88	-0.86	5	2.07	1.98	-0.85	5	0.93	0.23	-0.53	5	1.08	0.49	-0.68
10	50	25	100k	5	1.80	0.43	-0.37	5	0.89	0.15	-0.50	5	0.50	0.45	-0.37	5	0.68	0.41	-0.34
10	50	50	25k	5	350.01	304.02	-2.49	5	55.46	23.05	-0.28	5	2.60	0.45	-0.16	5	1.92	0.79	-0.06
10	50	50	50k	5	93.79	25.94	-1.95	5	44.72	15.37	-0.81	5	3.35	0.40	-0.44	5	3.49	1.19	-0.33
10	50	50	75k	5	37.92	28.24	-1.66	5	1.31	0.33	-0.85	5	1.15	0.38	-0.69	5	1.16	0.43	-0.71
10	50	50	100k	5	2.07	0.73	-0.21	5	1.41	1.39	-0.28	5	0.56	0.50	-0.16	5	0.59	0.53	-0.17
10	100	25	25k	5	157.03	42.60	-0.26	5	1.88	1.28	-0.03	5	2.55	0.48	0.00	5	2.01	0.34	-0.02
10	100	25	50k	5	127.21	59.69	-0.42	5	32.18	6.55	-0.36	5	3.36	0.20	-0.16	5	3.28	0.18	-0.20
10	100	25	75k	5	2.11	0.34	-0.49	5	1.00	0.30	-0.53	5	0.97	0.05	-0.50	5	1.00	0.27	-0.58
10	100	25	100k	5	2.36	0.47	-0.40	5	1.66	0.32	-0.61	5	1.59	0.30	-0.53	5	1.91	0.47	-0.74
10	100	50	25k	5	2760.84	3000.33	-0.32	5	6.99	10.12	-0.02	5	2.34	0.45	0.00	5	1.76	0.99	0.00
10	100	50	50k	5	1392.34	748.11	-0.89	5	71.09	56.12	-0.66	5	3.44	0.33	-0.28	5	3.39	0.21	-0.34
10	100	50	75k	5	252.20	122.28	4.38	5	0.93	0.22	5.83	5	1.00	0.19	5.71	5	0.91	0.13	5.76
10	100	50	100k	5	139.22	58.02	17.69	5	39.99	25.41	18.41	5	1.70	0.33	19.46	5	4.78	6.55	18.88
20	50	25	25k	5	128.97	28.76	-0.78	5	32.34	16.15	-0.16	5	2.49	0.34	-0.03	5	2.47	0.16	-0.09
20	50	25	50k	5	73.49	69.86	-1.16	5	32.61	18.43	-0.63	5	3.47	0.31	-0.34	5	3.32	0.36	-0.34
20	50	25	75k	5	2.67	0.27	-1.01	5	1.62	0.43	-1.05	5	1.50	0.19	-0.93	5	1.39	0.26	-0.84
20	50	25	100k	5	2.82	0.42	-0.78	5	2.90	0.27	-1.64	5	1.69	0.08	-0.44	5	2.36	0.12	-1.13
20	50	50	25k	5	5545.80	1637.69	-0.50	5	249.89	383.44	-0.41	5	2.65	0.50	-0.10	5	2.19	0.18	-0.02
20	50	50	50k	5	514.15	555.09	-5.45	5	20.36	4.66	-0.96	5	3.62	0.18	-0.36	5	5.05	1.77	-0.51
20	50	50	75k	5	219.05	119.69	-2.56	5	8.03	6.03	0.51	5	1.65	0.12	0.78	5	1.58	0.23	0.86
20	50	50	100k	5	89.49	19.32	-2.98	5	8.72	3.44	-0.20	5	2.11	0.66	0.56	5	2.37	0.30	0.31
20	100	25	25k	5	703.06	1270.46	-0.19	5	38.14	22.68	-0.06	5	2.26	0.44	0.00	5	2.04	0.47	-0.02
20	100	25	50k	5	29.03	17.42	-0.22	5	48.22	37.95	-0.40	5	3.40	0.26	-0.19	5	3.00	0.58	-0.18
20	100	25	75k	5	1.79	0.15	-0.46	5	0.90	0.21	-0.49	5	0.96	0.13	-0.55	5	0.90	0.16	-0.48
20	100	25	100k	5	2.72	0.10	-0.75	5	1.81	0.10	-0.70	5	1.85	0.14	-0.64	5	1.97	0.28	-0.84
20	100	50	25k	5	1360.68	1091.14	-0.25	5	39.05	21.48	-0.15	5	2.73	0.42	-0.02	5	2.01	1.09	-0.02
20	100	50	50k	5	702.47	898.00	-0.44	5	217.49	203.98	-0.26	5	3.48	0.31	-0.23	5	3.13	0.53	-0.18
20	100	50	75k	5	167.78	48.08	–	5	1.58	1.24	–	5	1.05	0.21	–	5	1.11	0.36	–
20	100	50	100k	5	129.01	163.57	–	5	14.14	1.44	–	5	1.86	0.25	–	5	2.04	0.12	–
				160	474.21	324.68	-0.25	160	32.03	28.01	0.35	160	2.15	0.32	0.61	160	2.20	0.65	0.55

To further evaluate the performance of the MILP model and the best implementations of the second decomposition approach, we generated new instances with 1000 vertices, which are larger than those considered before, and the results are presented in Table 8.8. For most of these larger instances, the MILP model cannot even find a feasible solution within the given time limit because of the increased complexity. On the other hand, the improvement of the second decomposition approach integrated with valid inequalities becomes striking. It can be noted that even though the problem sizes get larger, we can still find feasible solutions for all of the instances with the second decomposition approach. BD2 with only valid inequality (6.68) provides better feasible solutions for 121 out of 180 instances compared to BD2 with both valid inequalities (6.38) and (6.68). The increase in the problem size affects the performance of both methods. As a result, the best decomposition method turns out to be the one with the valid inequality (6.68).

8.2.3. Stochastic Network Slicing Problem

In the stochastic network slicing problem, the size of the problem is determined by the number of scenarios $|K|$, the number of end-user locations $|U|$, the number of services $|Q|$, and the number of potential server locations $|S|$. Two different values for the scaling factor (γ) are used as it affects the balance between different cost components in the objective function. We use the network topologies with 500 vertices. All mathematical programming formulations are solved using IBM CPLEX Optimization solver 20.1 running on a computer with Intel Xeon E5-2690 3.10 GHz CPU and 64 GB main memory using a single thread. The following tables show the outcomes of the computational study. For each problem instance, we set a time limit of one hour. The "Time" column is omitted in the tables since each solution method uses the entire 3600 seconds and is unable to prove the optimality of the best solution within this time limit. Instead, we compare the performances of different solution methods based on the average optimality gap and the quality of the best feasible solution obtained at the end of the time limit.

Table 8.8. Performance comparison of MILP formulation and BD2 on larger instances for deterministic network slicing problem.

				MILP			BD2 + (6.68)			BD2 + (6.38) + (6.68)		
$\|Q\|$	$\|U\|$	$\|S\|$	b	n	Gap	s	n	Gap	s	n	Gap	s
10	100	50	50k	5	3.03	0.63	5	3.34	0.22	5	3.43	0.82
10	100	50	100k	3	6.21	1.80	5	1.78	0.31	5	1.84	0.30
10	100	100	50k	1	2.59	–	5	3.47	0.28	5	4.70	1.57
10	100	100	100k	0	–	–	5	1.77	0.31	5	1.88	0.20
10	200	50	50k	2	19.19	23.93	5	3.17	0.22	5	3.45	0.76
10	200	50	100k	0	–	–	5	1.72	0.30	5	1.55	0.25
10	200	100	50k	0	–	–	5	3.31	0.53	5	5.86	2.15
10	200	100	100k	0	–	–	5	1.52	0.18	5	3.76	3.31
10	300	50	50k	2	3.81	0.05	5	3.25	0.72	5	3.32	0.33
10	300	50	100k	0	–	–	5	1.38	0.15	5	1.39	0.21
10	300	100	50k	0	–	–	5	3.48	0.71	5	4.05	0.96
10	300	100	100k	0	–	–	5	1.23	0.14	5	5.32	4.61
20	100	50	50k	1	3.16	–	5	3.51	0.26	5	4.52	1.31
20	100	50	100k	0	–	–	5	1.93	0.08	5	2.00	0.22
20	100	100	50k	0	–	–	5	3.40	0.35	5	5.07	1.16
20	100	100	100k	0	–	–	5	2.04	0.11	5	3.95	4.05
20	200	50	50k	0	–	–	5	3.80	0.93	5	4.97	2.02
20	200	50	100k	0	–	–	5	4.31	2.81	5	5.31	2.67
20	200	100	50k	0	–	–	5	5.61	1.10	5	5.16	3.27
20	200	100	100k	0	–	–	5	2.50	1.48	5	5.78	4.34
20	300	50	50k	0	–	–	5	4.50	0.21	5	3.74	0.99
20	300	50	100k	0	–	–	5	2.73	2.46	5	2.90	1.45
20	300	100	50k	0	–	–	5	4.70	1.16	5	4.74	0.73
20	300	100	100k	0	–	–	5	7.97	1.87	5	8.53	4.01
30	100	50	50k	2	15.07	17.71	5	3.59	0.16	5	5.37	1.32
30	100	50	100k	0	–	–	5	1.91	0.09	5	2.05	0.20
30	100	100	50k	0	–	–	5	3.49	0.38	5	5.12	1.93
30	100	100	100k	0	–	–	5	2.11	0.27	5	4.01	2.91
30	200	50	50k	0	–	–	5	4.51	1.01	5	4.52	0.78
30	200	50	100k	0	–	–	5	1.50	0.18	5	1.81	0.51
30	200	100	50k	0	–	–	5	4.43	1.43	5	4.25	1.23
30	200	100	100k	0	–	–	5	8.38	2.23	5	10.51	9.55
30	300	50	50k	0	–	–	5	2.91	0.56	5	3.22	0.27
30	300	50	100k	0	–	–	5	3.98	1.75	5	8.02	7.29
30	300	100	50k	0	–	–	5	5.29	0.84	5	7.00	2.83
30	300	100	100k	0	–	–	5	10.16	1.55	5	20.43	2.66
				16	7.23	5.75	180	3.57	0.76	180	4.82	2.03

In Table 8.9, the performances of the deterministic equivalent MILP model (shown by the columns titled "SP_MILP"), the performance of the decomposition approach (shown by the columns entitled "SP_BD"), the effect of valid inequalities (7.44) and (7.45), the effect of combinatorial cut (7.50), and the performance of the automated Benders decomposition feature of CPLEX are compared on the instances having 10 different scenarios. The dashes show that either the corresponding method cannot find a feasible solution for any instance within the given time limit or the statistics cannot be computed. As discussed in Chapter 3, the existing studies do not integrate the server placement, resource allocation, and task assignment decisions, and most of them suggest heuristic algorithms for the subproblems arising in network slicing design problem. Thus, we do not compare the results of our solution approaches against the proposed methods in the existing studies.

It can be observed that by solving the deterministic equivalent model, we can obtain feasible solutions with small optimality gaps. However, the increase in the problem size heavily affects its performance and deteriorates the solution quality obtained at the end of one hour time limit. In addition, the scaling parameter γ has some impact on the complexity of the problem as it defines the balance between strategic and operational level cost components. Out of 180 instances, the deterministic equivalent MILP model can find a feasible solution for 139 instances within the given time limit. The average optimality gap for the instances where the model identifies a feasible solution turns out to be %18.9.

In our computational experiments, we observed that eliminating all potential infeasible solutions by augmenting the master problem with inequalities (7.34) provides better results compared to the case where a feasibility cut is generated when the dual formulation of the subproblem turns out to be unbounded. In addition, partitioning the subproblem into smaller problems and adding multiple Benders optimality cuts of type (7.43) improves the solution process compared to the single-cut approach. In this section, we only report the results of multi-cut approach where master problem is augmented with inequalities (7.34) at the beginning of the algorithm.

The columns titled "SP_BD" show the results for the decomposition approach where the master problem has no valid inequality at the beginning. Since the feasibility of the subproblem is ensured by inequalities (7.34), any candidate master problem solution provides a feasible solution for the original problem. Hence, the method can find a feasible solution for all of the instances within the given time limit. However, the average optimality gap is always worse compared to the deterministic equivalent model for the instances for which a feasible solution is obtained. The comparison of the best objective values of both methods obtained at the end of the time limit is demonstrated on "UB Dev" column. It can be observed that the decomposition approach without the addition of any valid inequalities can find better feasible solutions for some of the large instances, but its overall performance is not justified.

The performance of the decomposition approach can be improved by integrating valid inequalities into the master problem formulation. It can be observed that by augmenting the master problem with valid inequalities (7.44) or (7.45), we can still find a feasible solution for all of the instances. We can also improve the solution quality in terms of the average optimality gap obtained at the end of one hour time limit. Moreover, both methods can find better feasible solutions compared to the deterministic equivalent model for larger instances. In general, valid inequality (7.45) has better performance than (7.44) as it has a smaller average optimality gap. Although its performance is slightly worse than the deterministic equivalent MILP model for small instances, it outperforms for large instances since it can find a feasible solution for all of the instances with smaller average optimality gap. For those instances, it can usually find better feasible solutions in terms of the objective value. If both valid inequalities (7.44) and (7.45) are integrated at the same time, the number of constraints in the master problem increases significantly. However, the solution quality is not improved. Although we can find feasible solutions for all of the instances, the average optimality gap is higher.

We also investigate the effect of combinatorial cuts of type (7.50) on the decomposition approach. We first augment the master problem with valid inequality (7.45)

as it provides the best solutions so far. Then, in each iteration of the decomposition algorithm, we generate a combinatorial cut of type (7.50) in addition to regular Benders optimality cuts. With this approach, we are still able to find a feasible solution for all of the instances. The average optimality gap is slightly lower than the case where the master problem is only augmented with valid inequality (7.45).

Finally, the performance of the automated Benders decomposition feature of CPLEX is tested on the same test instances and the results of this experiment are presented in the columns "SP_Auto Benders + (7.44)". In the preliminary analysis, we observed that introduction of valid inequality (7.44) performs better in terms of the solution quality. Thus, we omit the results with no valid inequality introduced. The number of instances for which the automated Benders feature is able to find a feasible solution is quite low and the average optimality gap for those instances are higher compared to other solution approaches. Hence, the usage of automated Benders decomposition feature of CPLEX is not justified.

The performances of all solution methods are also investigated on the test instances having 20 scenarios and the results are reported in Table 8.10. It can be noted that solution quality of all methods gets worse when the number of scenarios increases since either the number of instances for which the corresponding method is able to find a feasible solution is smaller or the average optimality gap is higher for all methods. One thing to note is that the decomposition approach where the master problem is augmented with valid inequality (7.45) yields smaller average optimality gap if combinatorial cut of type (7.50) is also added in each iteration of the algorithm. Its effect becomes apparent especially when the number of end-user and potential server locations is increased.

Table 8.9. Performance comparison of deterministic equivalent model and decomposition approach for stochastic network slicing problem when $|K| = 10$.

					SP_MILP		SP_BD			SP_BD + (7.44)			SP_BD + (7.45)			SP_BD + (7.44) + (7.45)			SP_BD + (7.45) + (7.50)			SP_Auto Benders + (7.44)		
$\|K\|$	$\|U\|$	$\|Q\|$	$\|S\|$	γ	n	Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap
10	25	5	5	50	5	0.7	5	2.6	1.1	5	2.3	0.7	5	1.4	0.7	5	2.2	0.8	5	1.9	0.7	5	3.2	1.0
10	25	5	5	100	5	0.6	5	3.2	1.3	5	2.3	0.7	5	1.3	0.7	5	2.4	0.8	5	1.5	0.7	5	19.3	114.9
10	50	5	5	50	5	1.0	5	3.1	1.6	5	2.4	1.2	5	2.3	1.1	5	2.2	1.1	5	2.3	1.1	5	21.7	50.5
10	50	5	5	100	5	1.0	5	2.4	1.2	5	2.1	1.1	5	2.0	0.9	5	1.9	1.1	5	2.0	0.9	4	2.5	126.8
10	100	5	5	50	5	0.6	5	1.3	0.6	5	1.3	0.5	5	1.3	0.5	5	1.3	0.5	5	1.3	0.5	5	45.9	76.2
10	100	5	5	100	5	1.3	5	1.7	0.2	5	1.6	0.2	5	1.6	0.2	5	1.6	0.2	5	1.6	0.2	5	50.9	107.2
10	25	10	5	50	5	0.7	5	5.2	4.3	5	3.6	2.7	5	3.7	2.7	5	3.5	2.6	5	3.7	2.7	1	95.9	269.3
10	25	10	5	100	5	1.4	5	4.8	3.7	5	4.4	3.6	5	4.5	3.4	5	4.3	3.5	5	4.6	3.5	0	-	614.0
10	50	10	5	50	5	0.6	5	3.1	2.2	5	2.8	1.9	5	2.8	1.9	5	2.8	1.9	5	2.8	1.9	5	20.9	48.1
10	50	10	5	100	5	0.6	5	3.8	2.8	5	3.5	2.6	5	3.5	2.6	5	3.5	2.6	5	3.5	2.6	5	22.7	69.6
10	100	10	5	50	5	4.9	5	99.9	54.2	5	0.7	-4.2	5	0.7	-4.2	5	0.7	-4.2	5	0.7	-4.2	5	31.2	40.3
10	100	10	5	100	5	19.9	5	100.0	36.3	5	0.8	-19.4	5	0.8	-27.3	5	0.8	-19.4	5	0.8	-19.4	5	2.4	-18.6
10	25	5	10	50	5	1.4	5	4.1	1.2	5	2.8	0.7	5	2.2	0.6	5	2.8	0.7	5	2.4	0.7	5	18.1	49.3
10	25	5	10	100	5	1.9	5	9.3	2.9	5	3.3	0.8	5	3.1	0.8	5	3.3	0.7	5	3.3	0.8	5	41.7	226.1
10	50	5	10	50	5	3.6	5	15.8	0.8	5	2.6	-1.4	5	2.9	-1.1	5	2.6	-1.2	5	2.6	-1.4	5	50.3	141.5
10	50	5	10	100	5	3.2	5	4.8	1.5	5	3.7	0.2	5	2.7	-0.9	5	2.6	-0.8	5	2.7	-0.9	5	20.8	83.3
10	100	5	10	50	3	70.7	5	93.8	-51.8	5	2.3	-70.0	5	2.2	-70.0	5	2.2	-70.0	5	2.0	-70.0	1	86.8	0.0
10	100	5	10	100	5	82.7	5	97.4	-27.0	5	3.1	-82.2	5	2.2	-82.3	5	3.2	-82.1	5	2.6	-82.2	0	-	15.6
10	25	10	10	50	5	2.9	5	62.5	44.6	5	4.4	0.9	5	4.4	1.0	5	4.2	0.7	5	4.5	1.0	5	85.2	261.4
10	25	10	10	100	5	2.2	5	62.3	132.8	5	5.6	2.9	5	4.8	2.1	5	4.7	2.0	5	5.1	2.4	0	-	608.8
10	50	10	10	50	5	63.1	5	99.8	37.5	5	4.0	-61.7	5	4.5	-61.6	5	4.7	-61.4	5	4.4	-61.6	1	96.6	39.6
10	50	10	10	100	5	77.8	5	99.9	66.2	5	4.6	-76.8	5	3.7	-77.0	5	5.6	-76.5	5	3.8	-77.0	0	-	66.3
10	100	10	10	50	0	-	5	100.0	-	5	3.1	-	5	3.1	-	5	3.1	-	5	3.0	-	5	73.7	-
10	100	10	10	100	0	-	5	100.0	-	5	3.8	-	5	3.9	-	5	3.9	-	5	4.0	-	5	81.1	-
10	25	5	20	50	5	1.6	5	30.3	5.9	5	4.6	2.5	5	2.8	0.6	5	2.7	0.5	5	2.8	0.6	5	81.7	221.7
10	25	5	20	100	5	2.0	5	43.4	10.0	5	5.2	2.8	5	3.0	0.3	5	4.0	1.4	5	3.2	0.6	5	90.9	522.2
10	50	5	20	50	5	46.3	5	83.6	44.0	5	5.6	-43.3	5	2.8	-44.8	5	5.0	-43.7	5	2.8	-44.9	3	63.1	46.8
10	50	5	20	100	5	49.1	5	72.9	-26.3	5	5.3	-46.8	5	2.8	-47.9	5	3.0	-47.9	5	3.0	-47.8	1	91.7	237.8
10	100	5	20	50	0	-	5	99.9	-	5	29.9	-	5	5.2	-	5	23.2	-	5	6.1	-	0	-	-
10	100	5	20	100	0	-	5	99.9	-	5	45.9	-	5	5.3	-	5	45.0	-	5	11.4	-	0	-	-
10	25	10	20	50	5	43.3	5	99.6	47.7	5	10.0	-37.6	5	4.7	-40.9	5	5.1	-40.7	5	4.7	-40.9	5	86.1	113.9
10	25	10	20	100	5	51.7	5	99.7	82.0	5	7.1	-48.0	5	5.5	-49.2	5	6.5	-48.2	5	4.8	-49.4	5	93.0	240.8
10	50	10	20	50	0	-	5	99.9	-	5	20.6	-	5	8.4	-	5	34.4	-	5	6.6	-	5	83.4	-
10	50	10	20	100	1	87.4	5	99.9	-100.0	5	30.5	-100.0	5	4.8	-86.8	5	49.9	-69.1	5	9.2	-85.8	0	-	0.0
10	100	10	20	50	0	-	5	99.9	-	5	35.4	-	5	11.9	-	5	46.6	-	5	9.7	-	1	96.3	-
10	100	10	20	100	0	-	5	99.9	-	5	42.7	-	5	26.0	-	5	59.3	-	5	13.1	-	0	-	-
					139	18.9	180	55.8	10.6	180	8.7	-15.7	180	4.1	-15.9	180	9.9	-15.1	180	4.0	-15.7	117	49.5	110.6

Table 8.10. Performance comparison of deterministic equivalent model and decomposition approach for stochastic network slicing problem when $|K| = 20$.

					SP_MILP		SP_BD			SP_BD + (7.44)			SP_BD + (7.45)			SP_BD + (7.44) + (7.45)			SP_BD + (7.45) + (7.50)			SP_Auto Benders + (7.44)		
20	25	5	5	50	5	1.0	5	3.4	2.0	5	2.5	0.7	5	1.6	0.7	5	2.1	0.7	5	2.0	0.9	5	29.3	81.5
20	25	5	5	100	5	1.7	5	4.5	3.1	5	3.2	1.0	5	1.9	0.9	5	2.9	1.0	5	2.6	0.8	5	3.9	1.0
20	50	5	5	50	5	6.1	5	3.2	-3.2	5	2.1	-4.1	5	2.0	-4.2	5	2.0	-4.2	5	2.0	-4.1	2	40.7	176.8
20	50	5	5	100	5	1.2	5	2.5	1.2	5	2.0	1.0	5	1.9	0.9	5	1.9	0.9	5	2.0	0.9	3	62.2	442.8
20	100	5	5	50	5	41.7	5	95.1	-5.3	5	1.3	-41.1	5	1.3	-41.1	5	1.3	-41.1	5	1.3	-41.1	5	74.7	50.5
20	100	5	5	100	5	50.6	5	97.3	-1.9	5	1.7	-49.9	5	1.6	-49.9	5	1.7	-49.9	5	1.6	-49.9	5	83.1	68.4
20	25	10	5	50	5	1.9	5	4.1	2.0	5	3.6	1.3	5	3.8	1.5	5	3.5	1.4	5	3.7	1.4	1	95.4	261.7
20	25	10	5	100	5	2.8	5	8.9	2.7	5	4.7	2.1	5	4.8	1.9	5	4.8	2.2	5	4.3	1.5	0	-	592.9
20	50	10	5	50	5	14.0	5	99.0	104.5	5	2.9	-11.7	5	2.9	-11.7	5	2.9	-11.7	5	2.9	-11.7	5	73.4	134.2
20	50	10	5	100	5	7.9	5	98.9	142.1	5	3.6	-4.9	5	3.6	-4.9	5	3.6	-4.9	5	3.5	-4.9	5	80.8	215.4
20	100	10	5	50	5	35.3	5	99.9	-80.2	5	0.8	-34.9	5	0.7	-34.9	5	0.8	-34.9	5	0.7	-34.9	5	40.1	4.3
20	100	10	5	100	5	41.9	5	100.0	-80.2	5	0.8	-41.5	5	0.8	-45.7	5	0.8	-41.5	5	0.8	-41.5	5	43.8	0.0
20	25	5	10	50	5	1.3	5	42.1	16.6	5	3.5	1.6	5	2.6	0.7	5	5.3	3.7	5	2.6	0.6	5	43.8	118.2
20	25	5	10	100	5	2.5	5	21.5	0.9	5	3.3	0.2	5	3.3	0.2	5	3.3	0.1	5	3.2	0.1	2	50.6	424.9
20	50	5	10	50	5	23.9	5	95.7	121.8	5	3.3	-21.8	5	2.5	-22.3	5	3.0	-21.9	5	2.5	-22.3	5	82.9	147.8
20	50	5	10	100	5	20.0	5	86.8	122.9	5	3.5	-17.4	5	2.8	-18.0	5	9.9	-6.8	5	3.5	-17.8	4	94.5	415.9
20	100	5	10	50	0	-	5	99.9	-	5	6.5	-	5	2.0	-	5	3.0	-	5	2.6	-	0	-	-
20	100	5	10	100	0	-	5	99.9	-	5	8.5	-	5	2.6	-	5	12.0	-	5	2.2	-	0	-	-
20	25	10	10	50	5	29.9	5	99.6	114.0	5	5.0	-26.8	5	4.9	-26.9	5	5.1	-26.8	5	4.8	-27.0	5	83.3	157.5
20	25	10	10	100	5	19.5	5	99.9	379.3	5	6.6	-14.1	5	5.3	-15.5	5	6.3	-14.7	5	4.9	-15.8	0	-	470.6
20	50	10	10	50	0	-	5	99.9	-	5	6.9	-	5	4.2	-	5	14.9	-	5	4.8	-	1	95.5	-
20	50	10	10	100	0	-	5	99.9	-	5	19.0	-	5	4.3	-	5	17.5	-	5	3.7	-	0	-	-
20	100	10	10	50	0	-	5	99.9	-	5	23.7	-	5	3.4	-	5	3.2	-	5	3.2	-	5	73.6	-
20	100	10	10	100	0	-	5	99.9	-	5	26.8	-	5	4.3	-	5	4.1	-	5	4.0	-	5	81.1	-
20	25	5	20	50	5	10.3	5	78.9	8.9	5	18.2	8.8	5	3.4	-7.7	5	7.1	-3.6	5	2.8	-8.2	5	83.2	192.4
20	25	5	20	100	5	13.2	5	80.2	58.3	5	12.4	1.3	5	3.3	-10.8	5	7.4	-6.0	5	3.3	-10.9	5	91.6	444.7
20	50	5	20	50	0	-	5	99.9	-	5	39.0	-	5	6.8	-	5	11.4	-	5	6.1	-	2	77.5	-
20	50	5	20	100	0	-	5	99.9	-	5	35.0	-	5	4.4	-	5	24.1	-	5	3.9	-	2	91.0	-
20	100	5	20	50	0	-	5	99.9	-	5	48.6	-	5	23.5	-	5	51.5	-	5	7.9	-	0	-	-
20	100	5	20	100	0	-	5	99.9	-	5	47.8	-	5	12.3	-	5	69.1	-	5	7.4	-	0	-	-
20	25	10	20	50	0	-	5	99.9	-	5	14.7	-	5	5.1	-	5	19.3	-	5	6.1	-	5	98.5	-
20	25	10	20	100	1	86.1	5	99.9	-100.0	5	22.5	-100.0	5	5.0	-85.5	5	30.6	-83.8	5	5.6	-85.6	5	99.2	0.0
20	50	10	20	50	0	-	5	99.9	-	5	33.3	-	5	19.2	-	5	44.8	-	5	9.8	-	4	83.9	-
20	50	10	20	100	0	-	5	99.9	-	5	43.2	-	5	19.6	-	5	74.4	-	5	18.5	-	0	-	-
20	100	10	20	50	0	-	5	99.9	-	5	72.5	-	5	53.5	-	5	67.6	-	5	15.7	-	0	-	-
20	100	10	20	100	0	-	5	99.9	-	5	79.4	-	5	39.5	-	5	83.5	-	5	24.2	-	0	-	-
					101	17.0	180	78.3	22.5	180	17.0	-9.7	180	7.4	-10.3	180	16.8	-9.5	180	5.0	-10.3	106	70.2	118.3

The performances of the deterministic equivalent MILP model and the best implementations of the decomposition approach are further evaluated on the instances with 50 scenarios. In Table 8.11, it can be observed that the deterministic equivalent model is unable to find a feasible solution within the given time limit for most of these larger instances due to the high complexity of the problem with large number of variables and constraints. On the other hand, the improvement of the decomposition approaches integrated with valid inequality (7.45) becomes striking as they can still find feasible solutions for all of the instances. Although introduction of combinatorial cut (7.50) requires solving an additional LP model in each iteration of the algorithm, it can strengthen the master problem formulation and improve the lower bound obtained at the end of time limit. Thus, it provides smaller average optimality gap for large instances.

For the stochastic network slicing problem with integer capacity levels, the performances of the solution methods are investigated on the same instances having 10 and 20 scenarios. The results of this experiment are given in Tables 8.12 and 8.13. We omit the results for the automated Benders decomposition feature of CPLEX as it is ineffective to find good-quality results within the given time limit. The performance of the deterministic equivalent MILP model is shown by the columns entitled "SP_MILP_int". It can be noted that although it can find feasible solutions with small optimality gaps for relatively smaller instances, it is unable to find a feasible solution for large instances with high number of decision variables and constraints. The average optimality gap is also high especially when the numbers of end-user and potential server locations are high.

Table 8.11. Performance comparison of deterministic equivalent model and decomposition approach for stochastic network slicing problem when $|K| = 50$.

					SP_MILP		SP_BD + (7.45)			SP_BD + (7.45) + (7.50)		
$\|K\|$	$\|U\|$	$\|Q\|$	$\|S\|$	γ	n	Gap	n	Gap	UB Gap	n	Gap	UB Gap
50	25	5	5	50	5	2.4	5	2.5	0.0	5	2.8	0.0
50	25	5	5	100	5	1.9	5	2.5	0.5	5	2.9	0.7
50	50	5	5	50	3	45.4	5	2.2	-44.4	5	2.3	-44.5
50	50	5	5	100	5	26.9	5	2.8	-24.5	5	2.7	-24.5
50	100	5	5	50	0	-	5	1.4	-	5	1.3	-
50	100	5	5	100	0	-	5	1.7	-	5	1.7	-
50	25	10	5	50	5	32.9	5	3.9	-30.3	5	4.0	-30.3
50	25	10	5	100	5	30.1	5	5.0	-25.9	5	4.6	-26.2
50	50	10	5	50	0	-	5	3.0	-	5	3.1	-
50	50	10	5	100	0	-	5	3.9	-	5	3.9	-
50	100	10	5	50	0	-	5	0.8	-	5	0.7	-
50	100	10	5	100	0	-	5	0.8	-	5	0.8	-
50	25	5	10	50	5	16.1	5	3.5	-13.5	5	2.8	-14.2
50	25	5	10	100	5	25.0	5	3.1	-23.1	5	3.5	-22.7
50	50	5	10	50	0	-	5	3.5	-	5	3.3	-
50	50	5	10	100	0	-	5	4.4	-	5	4.5	-
50	100	5	10	50	0	-	5	13.2	-	5	22.3	-
50	100	5	10	100	0	-	5	29.8	-	5	14.8	-
50	25	10	10	50	0	-	5	5.3	-	5	5.2	-
50	25	10	10	100	1	86.6	5	5.4	-86.0	5	5.5	-85.8
50	50	10	10	50	0	-	5	11.4	-	5	11.6	-
50	50	10	10	100	0	-	5	44.2	-	5	17.1	-
50	100	10	10	50	0	-	5	3.9	-	5	3.5	-
50	100	10	10	100	0	-	5	4.6	-	5	4.1	-
50	25	5	20	50	0	-	5	3.8	-	5	4.3	-
50	25	5	20	100	0	-	5	6.3	-	5	3.6	-
50	50	5	20	50	0	-	5	22.7	-	5	15.2	-
50	50	5	20	100	0	-	5	35.2	-	5	12.0	-
50	100	5	20	50	0	-	5	34.4	-	5	31.2	-
50	100	5	20	100	0	-	5	39.0	-	5	33.1	-
50	25	10	20	50	0	-	5	22.8	-	5	11.6	-
50	25	10	20	100	0	-	5	14.9	-	5	20.7	-
50	50	10	20	50	0	-	5	48.9	-	5	36.3	-
50	50	10	20	100	0	-	5	60.4	-	5	33.2	-
50	100	10	20	50	0	-	5	68.6	-	5	53.3	-
50	100	10	20	100	0	-	5	80.2	-	5	58.5	-
					39	23.1	180	16.7	-6.9	180	12.3	-6.9

The performance of the Benders decomposition approach is demonstrated by the columns entitled "SP_BD_int". It can be observed that the decomposition approach can always find a feasible solution for all of the instances within the given time limit. However, if no valid inequality is introduced into the master problem, the average optimality gap is worse compared to the deterministic equivalent model for some of the instances. When the best objective values of both methods obtained at the end of the time limit are compared using "UB Dev" column, we can notice that the decomposition approach is able to find better feasible solutions for most of the instances. Thus, its poor performance on the average optimality gap can be explained by the poor lower bounds, which can be avoided by introducing valid inequalities into the master problem.

In fact, the solution quality obtained by the decomposition approach can be improved by augmenting the master problem with valid inequalities (7.44) or (7.58). By this way, the average optimality gap can be reduced and we are able to find better feasible solutions compared to the deterministic equivalent model for most of the instances. It can be noticed that valid inequality (7.58) has better performance with smaller average optimality gap. When both valid inequalities are integrated into the master problem, the performance gets worse in terms of the average optimality gap and the quality of the best solution because of large number of constraints in the master problem. If combinatorial cut (7.59) is also integrated into the decomposition process where the master problem is augmented with valid inequality (7.58), there is no significant improvement in terms of the solution quality.

When the number of scenarios is increased to 20, the performances of all solution methods deteriorate. However, the improvement of the decomposition approach integrated with valid inequality (7.58) becomes striking. Although the problem sizes are increased, it can still find a feasible solution for all of the instances and the average optimality gap is smaller compared to the deterministic equivalent model. As a result, the best solution approach for the stochastic network slicing problem with integer capacity levels turns out to be the decomposition approach with the valid inequality (7.58).

Table 8.12. Performance comparison of deterministic equivalent model and decomposition approach for stochastic network slicing problem with integer capacity levels when $|K| = 10$.

					SP_MILP_int		SP_BD_int			SP_BD_int + (7.44)			SP_BD_int + (7.58)			SP_BD_int + (7.44) + (7.58)			SP_BD_int + (7.58) + (7.59)		
$\|K\|$	$\|U\|$	$\|Q\|$	$\|S\|$	γ	n	Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap
10	25	5	5	50	5	3.9	5	2.0	0.2	5	1.5	0.1	5	1.1	-0.1	5	1.8	0.1	5	1.5	0.4
10	25	5	5	100	5	5.2	5	1.8	0.0	5	1.3	0.0	5	0.9	-0.1	5	1.9	-0.1	5	2.3	0.0
10	25	5	10	50	5	5.4	5	4.0	-0.7	5	3.0	-1.0	5	2.2	-1.2	5	2.2	-1.1	5	2.2	-0.3
10	25	5	10	100	5	7.0	5	4.2	-1.1	5	3.5	-1.2	5	2.6	-1.2	5	2.8	-1.1	5	4.0	-0.2
10	25	5	20	50	5	6.7	5	36.1	8.6	5	10.2	3.6	5	2.3	-2.5	5	4.3	-1.5	5	5.2	0.7
10	25	5	20	100	5	8.1	5	34.1	6.8	5	12.6	2.1	5	2.8	-2.3	5	9.9	4.0	5	9.0	2.7
10	25	10	5	50	5	5.0	5	3.9	1.2	5	3.1	0.0	5	2.5	0.1	5	2.6	0.1	5	3.2	1.1
10	25	10	5	100	5	7.1	5	4.0	0.6	5	3.3	0.8	5	2.6	0.0	5	2.7	0.1	5	3.9	0.3
10	25	10	10	50	5	8.5	5	14.7	6.6	5	8.3	2.6	5	4.6	-1.8	5	5.7	-0.7	5	6.9	1.0
10	25	10	10	100	5	11.2	5	10.0	0.5	5	5.8	-1.0	5	4.3	-3.4	5	7.3	-0.4	5	6.8	-1.2
10	25	10	20	50	5	24.4	5	33.5	-9.9	5	7.2	-14.4	5	6.0	-17.4	5	12.5	-11.5	5	8.6	-12.8
10	25	10	20	100	5	43.3	5	37.3	-29.6	5	12.7	-33.1	5	8.5	-36.4	5	13.6	-33.9	5	11.9	-33.3
10	50	5	5	50	5	3.8	5	2.6	-0.8	5	2.5	-1.0	5	2.4	-0.8	5	1.6	-1.6	5	2.5	-1.0
10	50	5	5	100	5	4.7	5	2.5	-0.9	5	2.0	-1.3	5	1.0	-1.9	5	1.0	-1.8	5	0.9	-0.8
10	50	5	10	50	5	9.3	5	5.1	-4.9	5	4.3	-5.7	5	3.2	-6.0	5	3.1	-6.2	5	4.3	-4.5
10	50	5	10	100	5	5.9	5	3.2	-2.6	5	3.1	-3.2	5	2.9	-2.8	5	4.2	-1.4	5	3.4	-3.0
10	50	5	20	50	5	50.5	5	39.5	-43.2	5	14.7	-45.4	5	2.4	-49.1	5	5.0	-47.7	5	8.4	-46.3
10	50	5	20	100	5	29.8	5	37.2	-19.3	5	15.9	-23.4	5	4.6	-26.6	5	7.3	-24.4	5	10.3	-22.7
10	50	10	5	50	5	2.6	5	2.8	0.6	5	2.4	-0.2	5	1.8	-0.1	5	2.0	0.1	5	2.9	1.0
10	50	10	5	100	5	4.7	5	2.9	-1.5	5	2.1	-2.2	5	2.1	-1.9	5	2.9	-1.3	5	2.8	-2.3
10	50	10	10	50	5	61.6	5	7.6	-58.7	5	6.0	-59.3	5	5.5	-59.0	5	2.8	-60.1	5	5.7	-59.0
10	50	10	10	100	5	76.4	5	14.8	-74.2	5	8.6	-74.0	5	6.2	-74.6	5	5.2	-74.7	5	6.5	-73.9
10	50	10	20	50	0	-	5	61.3	-	5	24.6	-	5	8.6	-	5	30.7	-	5	12.2	-
10	50	10	20	100	1	87.4	5	57.1	-82.3	5	28.4	-82.5	5	9.7	-85.1	5	35.7	-78.5	5	24.6	-82.2
10	100	5	5	50	5	3.4	5	1.6	-1.9	5	1.2	-1.3	5	0.8	-2.4	5	0.9	-2.3	5	1.0	-1.7
10	100	5	5	100	5	3.8	5	3.5	-0.5	5	2.7	-1.9	5	1.5	-2.3	5	1.0	-2.4	5	1.5	-1.8
10	100	5	10	50	2	68.7	5	21.4	-67.7	5	7.3	-68.3	5	2.3	-68.6	5	3.6	-68.2	5	4.1	-68.1
10	100	5	10	100	5	78.8	5	4.4	-77.9	5	3.2	-78.1	5	1.9	-78.3	5	2.9	-78.0	5	3.2	-77.7
10	100	5	20	50	0	-	5	63.2	-	5	22.7	-	5	6.4	-	5	29.1	-	5	20.2	-
10	100	5	20	100	0	-	5	86.4	-	5	29.2	-	5	5.7	-	5	30.0	-	5	20.8	-
10	100	10	5	50	5	9.9	5	0.7	-9.3	5	0.6	-8.9	5	0.5	-9.5	5	0.5	-9.5	5	0.6	-9.9
10	100	10	5	100	5	10.6	5	1.0	-9.8	5	0.9	-9.8	5	0.5	-10.2	5	0.5	-10.2	5	0.5	-8.9
10	100	10	10	50	0	-	5	30.7	-	5	12.7	-	5	2.3	-	5	2.1	-	5	5.2	-
10	100	10	10	100	0	-	5	23.9	-	5	13.8	-	5	2.7	-	5	2.2	-	5	6.4	-
10	100	10	20	50	0	-	5	97.7	-	5	33.8	-	5	23.0	-	5	31.0	-	5	29.7	-
10	100	10	20	100	0	-	5	98.7	-	5	27.8	-	5	16.6	-	5	33.3	-	5	26.4	-
					138	19.4	180	23.8	-13.1	180	9.5	-14.1	180	4.3	-15.2	180	8.5	-14.3	180	7.5	-14.0

Table 8.13. Performance comparison of deterministic equivalent model and decomposition approach for stochastic network slicing problem with integer capacity levels when $|K| = 20$.

					SP_MILP_int		SP_BD_int			SP_BD_int + (7.44)			SP_BD_int + (7.58)			SP_BD_int + (7.44) + (7.58)			SP_BD_int + (7.58) + (7.59)		
$\|K\|$	$\|U\|$	$\|Q\|$	$\|S\|$	γ	n	Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap	n	Gap	UB Gap
20	25	5	5	50	5	3.9	5	2.3	-0.1	5	1.8	-0.1	5	1.7	0.0	5	2.2	0.1	5	2.2	0.2
20	25	5	5	100	5	5.3	5	4.5	0.0	5	4.2	0.1	5	2.3	0.2	5	1.9	-0.4	5	3.1	0.5
20	25	5	10	50	5	5.1	5	9.4	1.8	5	8.8	0.2	5	3.0	-0.4	5	4.3	0.8	5	3.8	0.1
20	25	5	10	100	5	8.2	5	7.7	-1.3	5	7.5	-2.4	5	2.7	-2.9	5	3.4	-2.2	5	3.1	-1.9
20	25	5	20	50	5	25.9	5	43.4	-10.9	5	16.6	-16.8	5	3.4	-21.9	5	9.8	-17.3	5	4.6	-21.2
20	25	5	20	100	5	41.1	5	52.2	-24.6	5	17.5	-27.2	5	3.4	-37.2	5	12.5	-32.5	5	4.1	-36.0
20	25	10	5	50	5	15.1	5	7.2	-6.9	5	4.3	-9.3	5	3.1	-10.6	5	3.2	-10.3	5	4.4	-10.3
20	25	10	5	100	5	36.0	5	6.0	-30.5	5	4.2	-30.3	5	3.0	-31.6	5	4.7	-30.3	5	3.8	-30.4
20	25	10	10	50	5	56.5	5	18.9	-48.2	5	10.0	-50.2	5	7.9	-51.9	5	8.5	-52.3	5	8.9	-51.6
20	25	10	10	100	5	62.4	5	19.3	-54.4	5	8.9	-56.5	5	6.0	-59.0	5	8.4	-58.3	5	8.7	-58.6
20	25	10	20	50	0	-	5	57.4	-	5	15.3	-	5	9.8	-	5	33.5	-	5	11.3	-
20	25	10	20	100	0	-	5	63.5	-	5	26.2	-	5	13.6	-	5	45.1	-	5	14.5	-
20	50	5	5	50	5	15.9	5	4.3	-12.0	5	3.0	-12.6	5	2.2	-13.4	5	2.2	-13.6	5	3.6	-13.1
20	50	5	5	100	5	21.4	5	4.1	-17.6	5	3.8	-18.1	5	2.2	-18.5	5	1.6	-19.0	5	3.6	-18.3
20	50	5	10	50	5	51.4	5	12.2	-47.4	5	8.2	-48.5	5	4.1	-49.2	5	4.1	-49.5	5	6.4	-48.7
20	50	5	10	100	5	49.8	5	13.3	-46.0	5	9.5	-46.9	5	5.1	-46.8	5	5.3	-47.0	5	7.4	-46.2
20	50	5	20	50	0	-	5	60.9	-	5	28.2	-	5	4.5	-	5	32.3	-	5	6.5	-
20	50	5	20	100	0	-	5	48.2	-	5	36.1	-	5	6.2	-	5	47.3	-	5	8.9	-
20	50	10	5	50	5	5.7	5	4.5	-1.4	5	3.3	-2.7	5	2.4	-3.2	5	3.0	-2.7	5	4.2	-2.0
20	50	10	5	100	5	15.5	5	4.7	-11.5	5	4.2	-12.2	5	3.0	-12.5	5	3.3	-12.3	5	4.8	-12.2
20	50	10	10	50	0	-	5	29.3	-	5	11.6	-	5	4.1	-	5	25.8	-	5	6.5	-
20	50	10	10	100	0	-	5	22.9	-	5	12.4	-	5	7.0	-	5	24.6	-	5	8.9	-
20	50	10	20	50	0	-	5	91.6	-	5	35.3	-	5	16.3	-	5	39.2	-	5	26.9	-
20	50	10	20	100	0	-	5	90.5	-	5	39.4	-	5	21.1	-	5	54.7	-	5	32.6	-
20	100	5	5	50	5	24.6	5	3.7	-22.6	5	2.7	-23.3	5	1.3	-23.6	5	1.1	-23.8	5	4.5	-22.5
20	100	5	5	100	5	47.4	5	5.3	-44.4	5	3.9	-45.6	5	1.4	-46.7	5	1.3	-46.7	5	2.7	-45.8
20	100	5	10	50	0	-	5	31.3	-	5	18.6	-	5	5.3	-	5	9.3	-	5	9.7	-
20	100	5	10	100	0	-	5	45.4	-	5	15.8	-	5	2.7	-	5	6.1	-	5	9.8	-
20	100	5	20	50	0	-	5	96.2	-	5	47.2	-	5	16.7	-	5	50.5	-	5	28.9	-
20	100	5	20	100	0	-	5	95.7	-	5	44.6	-	5	17.5	-	5	67.7	-	5	32.4	-
20	100	10	5	50	5	37.5	5	0.7	-37.2	5	0.2	-37.4	5	0.5	-37.3	5	0.5	-37.3	5	1.9	-36.3
20	100	10	5	100	5	40.7	5	3.6	-40.1	5	3.0	-39.9	5	0.6	-40.4	5	0.6	-40.4	5	2.0	-39.4
20	100	10	10	50	0	-	5	97.5	-	5	39.4	-	5	5.5	-	5	6.9	-	5	8.5	-
20	100	10	10	100	0	-	5	98.6	-	5	40.7	-	5	6.2	-	5	10.8	-	5	8.2	-
20	100	10	20	50	0	-	5	98.3	-	5	49.7	-	5	26.1	-	5	27.4	-	5	28.9	-
20	100	10	20	100	0	-	5	99.1	-	5	49.4	-	5	20.6	-	5	21.1	-	5	23.6	-
					100	28.5	180	37.6	-12.6	180	17.6	-13.3	180	6.7	-14.1	180	16.2	-13.7	180	9.8	-13.7

9. CONCLUSIONS

A wide variety of novel services have been envisioned lately due to recent innovations on mobile networking technologies and increased usage of mobile devices and wearable gadgets. This shift has significantly improved our daily lives and also transformed a wide range of verticals from manufacturing to entertainment. Meanwhile, 5G networks along with IoT devices are expected to provide a diverse set of services with stringent QoS requirements in terms of reliability, latency, and data traffic volume to satisfy user expectations. However, the traditional network architectures are no longer suitable to effectively accommodate these services with diversified characteristics and requirements. Due to the tremendous growth in mobile data traffic and SLA requirements of diversified services, it is critical for the operators to optimize their decisions in an economical way.

In this thesis, we focus on long-term investment planning decisions of a new entrant operator to design the computational architecture in an environment offering numerous service types with diversified characteristics. We introduce three problem instances under different assumptions. In the computation architecture design problem, we assume that the operator has an initial investment budget and future service demand is unknown. The objective of the problem is to maximize the revenue of the operator by optimizing server placement, service deployment, and task assignment decisions using the expected service demand. We formulate an MILP model that takes the latency restrictions of services into account. We then propose a Lagrangian relaxation-based heuristic algorithm to deal with the complexity of the problem. We also consider the case where demand and resource requirements of services can be stochastic. For this problem, we introduce a two-stage stochastic integer programming model and suggest SAA method as a solution approach.

Then, we study the network slicing concept that has emerged as an enabling technology in 5G networks to address conflicting requirements of services on a single

physical network infrastructure by transforming it into a set of logical networks. It helps the operators to provide customized solutions to various verticals and manage the operations in a more efficient and sustainable way. In the deterministic network slicing problem, we consider the optimal network slicing scheme that arises from the design of next-generation mobile networks to satisfy the demand for diversified services. We introduce an MILP formulation that integrates server placement, capacity allocation, and task assignment decisions in a comprehensive model. We then develop two exact solution methods based on branch-and-Benders-cut framework and suggest some valid inequalities and cut generation techniques to increase the efficiency of the decomposition strategy.

Then, we define the stochastic network slicing problem to deal with the dynamic changes in the service request demand over time. In this problem, we assume that the probability distribution of the number of service requests can be predicted based on some historical data. We construct a two-stage stochastic integer programming model where the objective is to minimize the cost of server placement decisions and the expected cost of unsatisfied service requests within the allowed delay limit. We also formulate a deterministic equivalent MILP model of the corresponding stochastic programming model by generating a set of scenarios for random parameters. In addition, we study a variant of this problem in which the capacity allocation of computational resources can only occur at discrete levels. For both variants, we develop a decomposition algorithm to obtain good-quality solutions in a reasonable amount of time for large instances.

The performances of the solution methods are tested on randomly generated instances. The computational results obtained on a large set of realistic test instances show that the proposed formulations address the key components of investment planning activities of an operator. We demonstrate that the suggested solution methods are highly effective and efficient, as they can deliver high-quality and robust solutions in a reasonable amount of computational effort.

The thesis opens up novel research perspectives. Since the increase in the problem size worsens the solution quality, one can focus on alternative operations research techniques on the suggested solution approaches, such as accelerated SAA method, Benders dual decomposition method suggested by Rahmaniani et al. [103], etc., to enhance the solution procedure. Alternatively, an online solution approach can be investigated for addressing the highly dynamic environment of next-generation mobile networks assuming that service request patterns and operations within the network may change over time.

REFERENCES

1. Cisco, *Cisco Annual Internet Report (2018–2023) White Paper*, `https://www.cisco.com/solutions/annual-internet-report.html`, accessed in December 2021.

2. Foukas, X., G. Patounas, A. Elmokashfi and M. K. Marina, "Network Slicing in 5G: Survey and Challenges", *IEEE Communications Magazine*, Vol. 55, No. 5, pp. 94–100, 2017.

3. Hassan, N., K.-L. A. Yau and C. Wu, "Edge Computing in 5G: A Review", *IEEE Access*, Vol. 7, pp. 127276–127289, 2019.

4. Qualcomm, *The 5G Economy*, `https://www.qualcomm.com/5g/the-5g-economy`, accessed in December 2021.

5. Xia, F., L. T. Yang, L. Wang and A. Vinel, "Internet of Things", *International Journal of Communication Systems*, Vol. 25, No. 9, p. 1101, 2012.

6. Li, S., L. Da Xu and S. Zhao, "5G Internet of Things: A Survey", *Journal of Industrial Information Integration*, Vol. 10, pp. 1–9, 2018.

7. Khan, L. U., I. Yaqoob, N. H. Tran, Z. Han and C. S. Hong, "Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges", *IEEE Access*, Vol. 8, pp. 36009–36028, 2020.

8. Hu, Y. C., M. Patel, D. Sabella, N. Sprecher and V. Young, "Mobile Edge Computing: A Key Technology Towards 5G", *ETSI White Paper*, Vol. 11, No. 11, pp. 1–16, 2015.

9. Afolabi, I., T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solu-

tions", *IEEE Communications Surveys & Tutorials*, Vol. 20, No. 3, pp. 2429–2453, 2018.

10. Su, R., D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang and Z. Zhu, "Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models", *IEEE Network*, Vol. 33, No. 6, pp. 172–179, 2019.

11. Shen, X., J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li and J. Rao, "AI-Assisted Network-Slicing based Next-Generation Wireless Networks", *IEEE Open Journal of Vehicular Technology*, Vol. 1, pp. 45–66, 2020.

12. Taleb, T., B. Mada, M.-I. Corici, A. Nakao and H. Flinck, "PERMIT: Network Slicing for Personalized 5G Mobile Telecommunications", *IEEE Communications Magazine*, Vol. 55, No. 5, pp. 88–93, 2017.

13. Baktır, A. C., B. Ahat, N. Aras, A. Özgövde and C. Ersoy, "SLA-Aware Optimal Resource Allocation for Service-Oriented Networks", *Future Generation Computer Systems*, Vol. 101, pp. 959–974, 2019.

14. Kleywegt, A. J., A. Shapiro and T. Homem-de Mello, "The Sample Average Approximation Method for Stochastic Discrete Optimization", *SIAM Journal on Optimization*, Vol. 12, No. 2, pp. 479–502, 2002.

15. Lin, L., X. Liao, H. Jin and P. Li, "Computation Offloading Toward Edge Computing", *Proceedings of the IEEE*, Vol. 107, No. 8, pp. 1584–1607, 2019.

16. Dinh, H. T., C. Lee, D. Niyato and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches", *Wireless Communications and Mobile Computing*, Vol. 13, No. 18, pp. 1587–1611, 2013.

17. Pan, J. and J. McElhannon, "Future Edge Cloud and Edge Computing for Internet of Things Applications", *IEEE Internet of Things Journal*, Vol. 5, No. 1, pp.

439–449, 2017.

18. Shi, W., J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges", *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp. 637–646, 2016.

19. Mach, P. and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading", *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 3, pp. 1628–1656, 2017.

20. Ordonez-Lucena, J., P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges", *IEEE Communications Magazine*, Vol. 55, No. 5, pp. 80–87, 2017.

21. Zhang, N., Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong and Z.-Q. Luo, "Network Slicing for Service-Oriented Networks under Resource Constraints", *IEEE Journal on Selected Areas in Communications*, Vol. 35, No. 11, pp. 2512–2521, 2017.

22. Vassilaras, S., L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah and G. S. Paschos, "The Algorithmic Aspects of Network Slicing", *IEEE Communications Magazine*, Vol. 55, No. 8, pp. 112–119, 2017.

23. Lin, H., S. Zeadally, Z. Chen, H. Labiod and L. Wang, "A Survey on Computation Offloading Modeling for Edge Computing", *Journal of Network and Computer Applications*, p. 102781, 2020.

24. Guo, Y., S. Wang, A. Zhou, J. Xu, J. Yuan and C.-H. Hsu, "User Allocation-Aware Edge Cloud Placement in Mobile Edge Computing", *Software: Practice and Experience*, Vol. 50, No. 5, pp. 489–502, 2020.

25. Cao, K., L. Li, Y. Cui, T. Wei and S. Hu, "Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing",

IEEE Transactions on Industrial Informatics, Vol. 17, No. 1, pp. 494–503, 2020.

26. Zhao, X., Y. Shi and S. Chen, "MAESP: Mobility Aware Edge Service Placement in Mobile Edge Networks", *Computer Networks*, p. 107435, 2020.

27. Garcia-Saavedra, A., G. Iosifidis, X. Costa-Perez and D. J. Leith, "Joint Optimization of Edge Computing Architectures and Radio Access Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 36, No. 11, pp. 2433–2443, 2018.

28. Ceselli, A., M. Premoli and S. Secci, "Mobile Edge Cloud Network Design Optimization", *IEEE/ACM Transactions on Networking*, Vol. 25, No. 3, pp. 1818–1831, 2017.

29. Mondal, S., G. Das and E. Wong, "Cost-Optimal Cloudlet Placement Frameworks over Fiber-Wireless Access Networks for Low-Latency Applications", *Journal of Network and Computer Applications*, Vol. 138, pp. 27–38, 2019.

30. Couto, R. S., S. Secci, M. E. M. Campista and L. H. M. Costa, "Server Placement with Shared Backups for Disaster-Resilient Clouds", *Computer Networks*, Vol. 93, pp. 423–434, 2015.

31. Wang, S., Y. Zhao, J. Xu, J. Yuan and C.-H. Hsu, "Edge Server Placement in Mobile Edge Computing", *Journal of Parallel and Distributed Computing*, Vol. 127, pp. 160–168, 2019.

32. IBM, *IBM CPLEX Optimization Studio*, `http://www.ibm.com/cplex-optimizer`, accessed in December 2021.

33. Li, D., P. Hong, K. Xue and J. Pei, "Virtual Network Function Placement and Resource Optimization in NFV and Edge Computing Enabled Networks", *Computer Networks*, Vol. 152, pp. 12–24, 2019.

34. Zeng, F., Y. Ren, X. Deng and W. Li, "Cost-Effective Edge Server Placement in Wireless Metropolitan Area Networks", *Sensors*, Vol. 19, No. 1, p. 32, 2019.

35. Li, C., J. Bai, Y. Chen and Y. Luo, "Resource and Replica Management Strategy for Optimizing Financial Cost and User Experience in Edge Cloud Computing System", *Information Sciences*, Vol. 516, pp. 33–55, 2020.

36. Farhadi, V., F. Mehmeti, T. He, T. La Porta, H. Khamfroush, S. Wang and K. S. Chan, "Service Placement and Request Scheduling for Data-Intensive Applications in Edge Clouds", *IEEE Conference on Computer Communications*, pp. 1279–1287, 2019.

37. Sun, G., V. Chang, G. Yang and D. Liao, "The Cost-Efficient Deployment of Replica Servers in Virtual Content Distribution Networks for Data Fusion", *Information Sciences*, Vol. 432, pp. 495–515, 2018.

38. Santoyo-González, A. and C. Cervelló-Pastor, "Latency-Aware Cost Optimization of the Service Infrastructure Placement in 5G Networks", *Journal of Network and Computer Applications*, Vol. 114, pp. 29–37, 2018.

39. Xu, K., X. Li, S. K. Bose and G. Shen, "Joint Replica Server Placement, Content Caching, and Request Load Assignment in Content Delivery Networks", *IEEE Access*, Vol. 6, pp. 17968–17981, 2018.

40. Ceselli, A., M. Premoli and S. Secci, "Cloudlet Network Design Optimization", *IFIP Networking Conference*, pp. 1–9, 2015.

41. Ren, Y., F. Zeng, W. Li and L. Meng, "A Low-Cost Edge Server Placement Strategy in Wireless Metropolitan Area Networks", *27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, 2018.

42. Ouyang, T., Z. Zhou and X. Chen, "Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing", *IEEE Journal on Selected*

Areas in Communications, Vol. 36, No. 10, pp. 2333–2345, 2018.

43. Yu, Y., J. Yang, C. Guo, H. Zheng and J. He, "Joint Optimization of Service Request Routing and Instance Placement in the Microservice System", *Journal of Network and Computer Applications*, Vol. 147, p. 102441, 2019.

44. Baccarelli, E., M. Scarpiniti and A. Momenzadeh, "EcoMobiFog–Design and Dynamic Optimization of a 5G Mobile-Fog-Cloud Multi-Tier Ecosystem for the Real-Time Distributed Execution of Stream Applications", *IEEE Access*, Vol. 7, pp. 55565–55608, 2019.

45. Li, Y. and S. Wang, "An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing", *IEEE International Conference on Edge Computing (EDGE)*, pp. 66–73, 2018.

46. Li, R., Q. Zheng, X. Li and Z. Yan, "Multi-Objective Optimization for Rebalancing Virtual Machine Placement", *Future Generation Computer Systems*, Vol. 105, pp. 824–842, 2020.

47. Wang, L., L. Jiao, T. He, J. Li and M. Mühlhäuser, "Service Entity Placement for Social Virtual Reality Applications in Edge Computing", *IEEE Conference on Computer Communications*, pp. 468–476, 2018.

48. Tsai, J.-S., I.-H. Chuang, J.-J. Liu, Y.-H. Kuo and W. Liao, "QoS-Aware Fog Service Orchestration for Industrial Internet of Things", *IEEE Transactions on Services Computing*, 2020.

49. Zhang, N., S. Guo, Y. Dong and D. Liu, "Joint Task Offloading and Data Caching in Mobile Edge Computing Networks", *Computer Networks*, Vol. 182, p. 107446, 2020.

50. Scarpiniti, M., E. Baccarelli and A. Momenzadeh, "VirtFogSim: A Parallel Toolbox for Dynamic Energy-Delay Performance Testing and Optimization of 5G

Mobile-Fog-Cloud Virtualized Platforms", *Applied Sciences*, Vol. 9, No. 6, p. 1160, 2019.

51. Yin, H., X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao and F. Li, "Edge Provisioning with Flexible Server Placement", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 4, pp. 1031–1045, 2016.

52. Shao, Y., C. Li, Z. Fu, L. Jia and Y. Luo, "Cost-Effective Replication Management and Scheduling in Edge Computing", *Journal of Network and Computer Applications*, Vol. 129, pp. 46–61, 2019.

53. Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. De Rose and R. Buyya, "CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software: Practice and Experience*, Vol. 41, No. 1, pp. 23–50, 2011.

54. Destounis, A., G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte and P. Medagliani, "Slice-based Column Generation for Network Slicing", *IEEE Conference on Computer Communications Workshops*, pp. 1–2, 2018.

55. Leconte, M., G. S. Paschos, P. Mertikopoulos and U. C. Kozat, "A Resource Allocation Framework for Network Slicing", *IEEE Conference on Computer Communications*, pp. 2177–2185, 2018.

56. De Domenico, A., Y.-F. Liu and W. Yu, "Optimal Virtual Network Function Deployment for 5G Network Slicing in a Hybrid Cloud Infrastructure", *IEEE Transactions on Wireless Communications*, Vol. 19, No. 12, pp. 7942–7956, 2020.

57. Xiang, B., J. Elias, F. Martignon and E. Di Nitto, "Joint Network Slicing and Mobile Edge Computing in 5G Networks", *IEEE International Conference on Communications*, pp. 1–7, 2019.

58. Fossati, F., S. Moretti, P. Perny and S. Secci, "Multi-Resource Allocation for

Network Slicing", *IEEE/ACM Transactions on Networking*, Vol. 28, No. 3, pp. 1311–1324, 2020.

59. Lee, Y. L., J. Loo, T. C. Chuah and L.-C. Wang, "Dynamic Network Slicing for Multitenant Heterogeneous Cloud Radio Access Networks", *IEEE Transactions on Wireless Communications*, Vol. 17, No. 4, pp. 2146–2161, 2018.

60. Jiang, M., M. Condoluci and T. Mahmoodi, "Network Slicing Management & Prioritization in 5G Mobile Systems", *22th European Wireless Conference*, pp. 1–6, 2016.

61. Bega, D., M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis and X. Costa-Perez, "Optimising 5G Infrastructure Markets: The Business of Network Slicing", *IEEE Conference on Computer Communications*, pp. 1–9, 2017.

62. Sattar, D. and A. Matrawy, "Optimal Slice Allocation in 5G Core Networks", *IEEE Networking Letters*, Vol. 1, No. 2, pp. 48–51, 2019.

63. Chen, W.-K., Y. F. Liu, A. De Domenico and Z. Q. Luo, "Network Slicing for Service-Oriented Networks with Flexible Routing and Guaranteed E2E Latency", *arXiv preprint arXiv:2002.07380*, 2020.

64. Vo, P. L., M. N. Nguyen, T. A. Le and N. H. Tran, "Slicing the Edge: Resource Allocation for RAN Network Slicing", *IEEE Wireless Communications Letters*, Vol. 7, No. 6, pp. 970–973, 2018.

65. Caballero, P., A. Banchs, G. De Veciana, X. Costa-Pérez and A. Azcorra, "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games", *IEEE Transactions on Wireless Communications*, Vol. 17, No. 10, pp. 6419–6432, 2018.

66. Feng, J., Q. Pei, F. R. Yu, X. Chu, J. Du and L. Zhu, "Dynamic Network Slicing and Resource Allocation in Mobile Edge Computing Systems", *IEEE Transac-*

tions on Vehicular Technology, Vol. 69, No. 7, pp. 7863–7878, 2020.

67. Richart, M., J. Baliosian, J. Serrati, J.-L. Gorricho, R. Agüero and N. Agoulmine, "Resource Allocation for Network Slicing in WiFi Access Points", *13th International Conference on Network and Service Management (CNSM)*, pp. 1–4, 2017.

68. Zhang, H., N. Liu, X. Chu, K. Long, A.-H. Aghvami and V. C. Leung, "Network Slicing based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges", *IEEE Communications Magazine*, Vol. 55, No. 8, pp. 138–145, 2017.

69. Ksentini, A. and N. Nikaein, "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction", *IEEE Communications Magazine*, Vol. 55, No. 6, pp. 102–108, 2017.

70. Wen, R., G. Feng, J. Tang, T. Q. Quek, G. Wang, W. Tan and S. Qin, "On Robustness of Network Slicing for Next-Generation Mobile Networks", *IEEE Transactions on Communications*, Vol. 67, No. 1, pp. 430–444, 2018.

71. Wang, G., G. Feng, T. Q. Quek, S. Qin, R. Wen and W. Tan, "Reconfiguration in Network Slicing: Optimizing the Profit and Performance", *IEEE Transactions on Network and Service Management*, Vol. 16, No. 2, pp. 591–605, 2019.

72. Cunha, V. A., E. da Silva, M. B. de Carvalho, D. Corujo, J. P. Barraca, D. Gomes, L. Z. Granville and R. L. Aguiar, "Network Slicing Security: Challenges and Directions", *Internet Technology Letters*, Vol. 2, No. 5, p. e125, 2019.

73. Yousaf, F. Z., M. Gramaglia, V. Friderikos, B. Gajic, D. Von Hugo, B. Sayadi, V. Sciancalepore and M. R. Crippa, "Network Slicing with Flexible Mobility and QoS/QoE Support for 5G Networks", *IEEE International Conference on Communications Workshops*, pp. 1195–1201, 2017.

74. Luu, Q.-T., S. Kerboeuf and M. Kieffer, "Uncertainty-Aware Resource Provision-

ing for Network Slicing", *IEEE Transactions on Network and Service Management*, Vol. 18, No. 1, pp. 79–93, 2021.

75. Rost, P., C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz and H. Bakker, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks", *IEEE Communications Magazine*, Vol. 55, No. 5, pp. 72–79, 2017.

76. Olimid, R. F. and G. Nencioni, "5G Network Slicing: A Security Overview", *IEEE Access*, Vol. 8, pp. 99999–100009, 2020.

77. Baumgartner, A., T. Bauschert, A. M. Koster and V. S. Reddy, "Optimisation Models for Robust and Survivable Network Slice Design: A Comparative Analysis", *IEEE Global Communications Conference*, pp. 1–7, 2017.

78. Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2020, `https://www.gurobi.com/documentation/9.0/refman.pdf`, accessed in December 2021.

79. Sharma, S., A. Gumaste and M. Tatipamula, "Dynamic Network Slicing Using Utility Algorithms and Stochastic Optimization", *IEEE 21st International Conference on High Performance Switching and Routing*, pp. 1–8, 2020.

80. Zhang, H. and V. W. Wong, "A Two-Timescale Approach for Network Slicing in C-RAN", *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 6, pp. 6656–6669, 2020.

81. Bektaş, T., J. F. Cordeau, E. Erkut and G. Laporte, "Exact Algorithms for the Joint Object Placement and Request Routing Problem in Content Distribution Networks", *Computers & Operations Research*, Vol. 35, No. 12, pp. 3860–3884, 2008.

82. Sen, G., M. Krishnamoorthy, N. Rangaraj and V. Narayanan, "Exact Approaches

for Static Data Segment Allocation Problem in an Information Network", *Computers & Operations Research*, Vol. 62, pp. 282–295, 2015.

83. Gendron, B., M. G. Scutellà, R. G. Garroppo, G. Nencioni and L. Tavanti, "A Branch-and-Benders-Cut Method for Nonlinear Power Design in Green Wireless Local Area Networks", *European Journal of Operational Research*, Vol. 255, No. 1, pp. 151–162, 2016.

84. Li, X. and Y. P. Aneja, "A Branch-and-Benders-Cut Approach for the Fault Tolerant Regenerator Location Problem", *Computers & Operations Research*, Vol. 115, p. 104847, 2020.

85. Boffey, B., R. Galvao and L. Espejo, "A Review of Congestion Models in the Location of Facilities with Immobile Servers", *European Journal of Operational Research*, Vol. 178, No. 3, pp. 643–662, 2007.

86. Berman, O. and R. R. Mandowsky, "Location-Allocation on Congested Networks", *European Journal of Operational Research*, Vol. 26, No. 2, pp. 238–250, 1986.

87. Desrochers, M., P. Marcotte and M. Stan, "The Congested Facility Location Problem", *Location Science*, Vol. 3, No. 1, pp. 9–23, 1995.

88. Berman, O. and D. Krass, "Stochastic Location Models with Congestion", *Location Science*, pp. 477–535, Springer, 2019.

89. Sheu, S.-T. and J. Chen, "A Novel Delay-Oriented Shortest Path Routing Protocol for Mobile Ad Hoc Networks", *IEEE International Conference on Communications*, Vol. 6, pp. 1930–1934, 2001.

90. Kurose, J. F. and K. W. Ross, *Computer Networking: A Top-Down Approach*, Addison Wesley, 2017.

91. Jia, M., J. Cao and W. Liang, "Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks", *IEEE Transactions on Cloud Computing*, Vol. 5, No. 4, pp. 725–737, 2017.

92. Ahat, B., A. C. Baktır, N. Aras, İ. K. Altınel, A. Özgövde and C. Ersoy, "Optimal Server and Service Deployment for Multi-Tier Edge Cloud Computing", *Computer Networks*, Vol. 199, p. 108393, 2021.

93. Kariv, O. and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems. II: The p-medians", *SIAM Journal on Applied Mathematics*, Vol. 37, No. 3, pp. 539–560, 1979.

94. Fisher, M. L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science*, Vol. 27, No. 1, pp. 1–18, 1981.

95. Geoffrion, A. M., "Lagrangean Relaxation for Integer Programming", *Mathematical Programming Study*, Vol. 2, pp. 82–114, 1974.

96. Taskın, Z. C., "Benders Decomposition", *Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Malden (MA)*, 2010.

97. Codato, G. and M. Fischetti, "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming", *Operations Research*, Vol. 54, No. 4, pp. 756–766, 2006.

98. Rahmaniani, R., T. G. Crainic, M. Gendreau and W. Rei, "The Benders Decomposition Algorithm: A Literature Review", *European Journal of Operational Research*, Vol. 259, No. 3, pp. 801–817, 2017.

99. European Telecommunications Standard Institute (ETSI), *ETSI GS MEC-IEG 004 v1.1.1 - Mobile-Edge Computing Service Scenarios*, 2015, `https://www.etsi.org/deliver/`, accessed in December 2021.

100. 5G PPP, *5G PPP Use Cases and Performance Evaluation Models*, 2016,

`https://5g-ppp.eu/white-papers`, accessed in December 2021.

101. Hagberg, A., P. Swart and D. S Chult, *Exploring Network Structure, Dynamics, and Function using NetworkX*, Tech. rep., Los Alamos National Lab (LANL), Los Alamos, NM (United States), 2008.

102. Knight, S., H. X. Nguyen, N. Falkner, R. Bowden and M. Roughan, "The Internet Topology Zoo", *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 9, pp. 1765–1775, 2011.

103. Rahmaniani, R., S. Ahmed, T. G. Crainic, M. Gendreau and W. Rei, "The Benders Dual Decomposition Method", *Operations Research*, Vol. 68, No. 3, pp. 878–895, 2020.

APPENDIX A: ABOUT THE FIGURES USED IN THE DOCUMENT

The figures that emerged within the scope of this document and whose copyrights were transferred to the publishing house are used in the thesis in accordance with the "publishing policy valid for the reuse of the text and graphics produced by the author" on the website of the publisher.