Routing optimization algorithms in integrated fronthaul/backhaul networks supporting multitenancy

by

Nuria Molner Siurana

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematic Engineering

Universidad Carlos III de Madrid

Advisor:
Antonio de la Oliva Delgado

September 2021

Routing optimization algorithms in integrated fronthaul/backhaul networks supporting multitenancy

Prepared by:

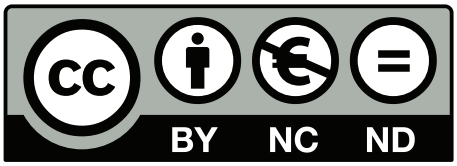
Nuria Molner Siurana
IMDEA Networks Institute, Universidad Carlos III de Madrid
contact: nuria.molner@imdea.org

Under the advice of:

Antonio de la Oliva Delgado
Universidad Carlos III de Madrid

Telematic Engineering Department, Universidad Carlos III de Madrid

This work has been supported by:



"The secret of getting ahead is getting started."
– Mark Twain

"It does not matter how slowly you go as long as you do not stop."
– Confucius

"It always seems impossible until it's done."
– Nelson Mandela

Acknowledgements

I would like to thank all the people who crossed my path during the PhD and helped me to improve and grow both as a researcher and as a person. All the experiences lived along those years made me the person I am today.

First of all, a big thank you to my lab colleagues, most of which became friends, for making my everyday easy at the office and gave me motivation, hope and a reason to continue even in the hardest times.

Also a big thank you to Arturo and Antonio, who offered me the opportunity to do this PhD and believed in me from the very beginning.

I cannot forget to thank Ioannis, for all the help offered in my research. This thesis would have not been the same without your contribution.

And another big thank you goes for the coauthors of the papers developed during the period of this PhD, who played a very important role on the development and on the value of this thesis.

Last, but not least, a huge thank you to my family and friends, without whose support this journey would have not been possible. I thank them for always being a source of inspiration, and believing in me and gave me the courage to start this adventure and continue day by day.

Thank you mom, dad, Sonia, for being my biggest fans and supporters no matter what I decide or do, this thesis is dedicated to you!

THANK YOU to all of you.

Published Content

The content of this thesis is based on the following published papers:

[1] **Nuria Molner**, Antonio de la Oliva, Ioannis Stavrakakis, Arturo Azcorra. Optimization of an integrated fronthaul/backhaul network under path and delay constraints. Published in *Ad Hoc Networks, vol. 83, pp. 41-54, ISSN 1570-8705*, 2019. https://doi.org/10.1016/j.adhoc.2018.08.025

- This work is fully included and its content is reported in Chapter 2.
- The author's role in this work is focused on the design, implementation and experimentation of the proposed methodology.

[2] Balázs Németh, **Nuria Molner**, Jorge Martín Pérez, Carlos Jesús Bernardos, Antonio de la Oliva, Balázs Sonkoly. Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure. Published in *IEEE Transactions on Mobile Computing*, 2021. https://doi.org/10.1109/TMC.2021.3055426

- This work is fully included and its content is reported in Chapter 3.
- The author's role in this work is focused on the design of the proposed optimization model and its implementation in AMPL.

[3] Francesco Malandrino, Carla Fabiana Chiasserini, **Nuria Molner**, Antonio de la Oliva. Network Support for High-performance Distributed Machine Learning. *[Submitted to IEEE Transactions on Networking, Under Revision].*

- This work is fully included and its content is reported in Chapter 4.
- The author's role in this work is focused on the design of the proposed optimization model and characterization of the theoretical constraints.

Other Publications and Submitted Content

[4] **Nuria Molner**, Sergio González, Thomas Deiß, Antonio de la Oliva. The 5G-Crosshaul Packet Forwarding Element pipeline: Measurements and analysis. Published in *Fifth International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN)*, 2017. https://doi.org/10.23919/CLEEN.2017.8045908

[5] Anna Tzanakaki, Markos Anastasopoulos, Nathan Gomes, Philippos Assimakopoulos, Josep M. Fàbrega, Michela Svaluto Moreolo, Laia Nadal, Jesús Gutiérrez, Vladica Sark, Eckhard Grass, Daniel Camps-Mur, Antonio de la Oliva, **Nuria Molner**, Xavier Costa Perez, Josep Mangues, Ali Yaver, Paris Flegkas, Nikos Makris, Thanasis Korakis, Dimitra Simeonidou. Transport network architecture. Published in *5G System Design: Architectural and Functional Considerations and Long Term Research*, pp. 151-180, John Wiley & Sons, 2018, ISBN: 978-1-119-42512-0.

[6] Claudio Casetti, Carla Fabiana Chiasserini, Thomas Deiß, Pantelis A. Frangoudis, Adlen Ksentini, Giada Landi, Xi Li, **Nuria Molner**, Josep Mangues. Network slices for vertical industries. Published in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 254-259*, 2018. https://doi.org/10.1109/WCNCW.2018.8368981

[7] Claudio Casetti, Carla Fabiana Chiasserini, Jorge Martín Pérez, **Nuria Molner**, Thomas Deiß, Jose Enrique Blázquez González, Cao-Thanh Phan, Farouk Messaoudi, Giada Landi, Nicolás Serrano, Josep Mangues, Charles Turyagyenda. The Vertical Slicer: Verticals' Entry Point to 5G Networks. Published in *The 27th European Conference on Networks and Communications (EuCNC 2018)*, 18-21 June 2018, Ljubljana, Slovenia.

[8] Claudio Casetti, Carla Fabiana Chiasserini, **Nuria Molner**, Jorge Martín Pérez, Thomas Deiß, Cao-Thanh Phan, Farouk Messaoudi, Giada Landi, Juan Brenes Baranzano. Arbitration Among Vertical Services. Published in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 153-157*, 2018. https://doi.org/10.1109/PIMRC.2018.8580852

[**9**] Kiril Antevski, Jorge Martín Pérez, **Nuria Molner**, Carla Fabiana Chiasserini, Francesco Malandrino, Pantelis A. Frangoudis, Adlen Ksentini, Xi Li, Josep Salvat Lozano, Ricardo Martínez, Iñaki Pascual, Josep Mangues Bafalluy, Jorge Baranda, Barbara Martini, Molka Gharbaoui. Resource Orchestration of 5G Transport Networks for Vertical Industries. Published in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 158-163*, 2018. https://doi.org/10.1109/PIMRC.2018.8581029

Resumen

Esta tesis pretende ayudar en la definición y el diseño de la quinta generación de redes de telecomunicaciones (5G) a través del modelado matemático de las diferentes cualidades que las caracterizan. En general, la ambición de estos modelos es realizar una optimización de las redes, ensalzando sus capacidades recientemente adquiridas para mejorar la eficiencia de los futuros despliegues tanto para los usuarios como para los operadores. El periodo de realización de esta tesis se corresponde con el periodo de investigación y definición de las redes 5G, y, por lo tanto, en paralelo y en el contexto de varios proyectos europeos del programa H2020. Por lo tanto, las diferentes partes del trabajo presentado en este documento cuadran y ofrecen una solución a diferentes retos que han ido apareciendo durante la definición del 5G y dentro del ámbito de estos proyectos, considerando los comentarios y problemas desde el punto de vista de todos los usuarios finales, operadores y proveedores.

Así, el primer reto a considerar se centra en el núcleo de la red, en particular en cómo integrar tráfico *fronthaul* y *backhaul* en el mismo estrato de transporte. La solución propuesta es un marco de optimización para el enrutado y la colocación de recursos que ha sido desarrollado teniendo en cuenta restricciones de retardo, capacidad y caminos, maximizando el grado de despliegue de Unidades Distribuidas (DU) mientras se minimizan los agregados de las Unidades Centrales (CU) que las soportan. El marco y los algoritmos heurísticos desarrollados (para reducir la complexidad computacional) son validados y aplicados a redes tanto a pequeña como a gran (nivel de producción) escala. Esto los hace útiles para los operadores de redes tanto para la planificación de la red como para el ajuste dinámico de las operaciones de red en su infraestructura (virtualizada).

Moviéndonos más cerca de los usuarios, el segundo reto considerado se centra en la colocación de servicios en entornos de nube y borde (*cloud/edge*). En particular, el problema considerado consiste en seleccionar la mejor localización para cada función de red virtual (VNF) que compone un servicio en entornos de robots en la nube, que implica restricciones estrictas en las cotas de retardo y fiabilidad. Los robots, vehículos y otros dispositivos finales proveen competencias significativas como impulsores, sensores y computación local que son esenciales para algunos servicios. Por contra, estos dispositivos están en continuo movimiento y pueden perder la conexión con la red o quedarse sin

batería, cosa que reta aún más la entrega de servicios en este entorno dinámico. Así, el análisis realizado y la solución propuesta abordan las restricciones de movilidad y batería. Además, también se necesita tener en cuenta los aspectos temporales y los objetivos conflictivos de fiabilidad y baja latencia en el despliegue de servicios en una red volátil, donde los nodos de cómputo móviles actúan como una extensión de la infraestructura de cómputo de la nube y el borde. El problema se formula como un problema de optimización para colocación de VNFs minimizando el coste y también se propone un heurístico eficiente. Los algoritmos son evaluados de forma extensiva desde varios aspectos por simulación en escenarios que reflejan la realidad de forma detallada.

Finalmente, el último reto analizado se centra en dar soporte a servicios basados en el borde, en particular, aprendizaje automático (ML) en escenarios del Internet de las Cosas (IoT) distribuidos. El enfoque tradicional al ML distribuido se centra en adaptar los algoritmos de aprendizaje a la red, por ejemplo, reduciendo las actualizaciones para frenar la sobrecarga. Las redes basadas en el borde inteligente, en cambio, hacen posible seguir un enfoque opuesto, es decir, definir la topología de red lógica alrededor de la tarea de aprendizaje a realizar, para así alcanzar el resultado de aprendizaje deseado. La solución propuesta incluye un modelo de sistema que captura dichos aspectos en el contexto de ML supervisado, teniendo en cuenta tanto nodos de aprendizaje (que realizan las computaciones) como nodos de información (que proveen datos). El problema se formula para seleccionar (i) qué nodos de aprendizaje e información deben cooperar para completar la tarea de aprendizaje, y (ii) el número de iteraciones a realizar, para minimizar el coste de aprendizaje mientras se garantizan los objetivos de error predictivo y tiempo de ejecución. La solución también incluye un algoritmo heurístico que es evaluado ensalzando una topología de red real y considerando tanto las tareas de clasificación como de regresión, y cuya solución se acerca mucho al óptimo, superando las soluciones alternativas encontradas en la literatura.

Abstract

This thesis aims to help in the definition and design of the 5th generation of telecommunications networks (5G) by modelling the different features that characterize them through several mathematical models. Overall, the aim of these models is to perform a wide optimization of the network elements, leveraging their newly-acquired capabilities in order to improve the efficiency of the future deployments both for the users and the operators. The timeline of this thesis corresponds to the timeline of the research and definition of 5G networks, and thus in parallel and in the context of several European H2020 programs. Hence, the different parts of the work presented in this document match and provide a solution to different challenges that have been appearing during the definition of 5G and within the scope of those projects, considering the feedback and problems from the point of view of all the end users, operators and providers.

Thus, the first challenge to be considered focuses on the core network, in particular on how to integrate fronthaul and backhaul traffic over the same transport stratum. The solution proposed is an optimization framework for routing and resource placement that has been developed taking into account delay, capacity and path constraints, maximizing the degree of Distributed Unit (DU) deployment while minimizing the supporting Central Unit (CU) pools. The framework and the developed heuristics (to reduce the computational complexity) are validated and applied to both small and large-scale (production-level) networks. They can be useful to network operators for both network planning as well as network operation adjusting their (virtualized) infrastructure dynamically.

Moving closer to the user side, the second challenge considered focuses on the allocation of services in cloud/edge environments. In particular, the problem tackled consists of selecting the best the location of each Virtual Network Function (VNF) that compose a service in cloud robotics environments, that imply strict delay bounds and reliability constraints. Robots, vehicles and other end-devices provide significant capabilities such as actuators, sensors and local computation which are essential for some services. On the negative side, these devices are continuously on the move and might lose network connection or run out of battery, which further challenge service delivery in this dynamic environment. Thus, the performed analysis and proposed solution tackle the

mobility and battery restrictions. We further need to account for the temporal aspects and conflicting goals of reliable, low latency service deployment over a volatile network, where mobile compute nodes act as an extension of the cloud and edge computing infrastructure. The problem is formulated as a cost-minimizing VNF placement optimization and an efficient heuristic is proposed. The algorithms are extensively evaluated from various aspects by simulation on detailed real-world scenarios.

Finally, the last challenge analyzed focuses on supporting edge-based services, in particular, Machine Learning (ML) in distributed Internet of Things (IoT) scenarios. The traditional approach to distributed ML is to adapt learning algorithms to the network, e.g., reducing updates to curb overhead. Networks based on intelligent edge, instead, make it possible to follow the opposite approach, i.e., to define the logical network topology around the learning task to perform, so as to meet the desired learning performance. The proposed solution includes a system model that captures such aspects in the context of supervised ML, accounting for both learning nodes (that perform computations) and information nodes (that provide data). The problem is formulated to select (i) which learning and information nodes should cooperate to complete the learning task, and (ii) the number of iterations to perform, in order to minimize the learning cost while meeting the target prediction error and execution time. The solution also includes an heuristic algorithm that is evaluated leveraging a real-world network topology and considering both classification and regression tasks, and closely matches the optimum, outperforming state-of-the-art alternatives.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3GPP 3rd Generation Partnership Project

4G 4th generation of telecommunications networks

5G 5th generation of telecommunications networks

AI Artificial Intelligence

AP Access Point

CAPEX CAPital EXpenditures

CDF Cumulative Distribution Function

CoMP Coordinated Multi-Point

CPRI Common Public Radio Interface

CPU Central Processing Unit

C-RAN Cloud or Centralized Radio Access Network

CU Central Unit

DU Distributed Unit

E2E End-to-End

eCPRI Enhanced Common Public Radio Interface

eNB evolved Node B

GA Genetic Algorithm

gNB next generation Node B

H2020 Horizon 2020 EU Research and Innovation programme

IEEE Institute of Electrical and Electronics Engineers

i.i.d. Independent and Identically Distributed

IoT Internet of Things

JCR Journal Citation Reports

LTE Long Term Evolution

MEC Multi-access Edge Computing

MIMO Multiple-Input Multiple-Output

ML Machine Learning

NF Network Function

NFV Network Function Virtualization

NR New Radio

NS Network Service

OPEX OPerational EXpenditures

OTT Over the Top

pdf Probabiliy Density Function

RAN Radio Access Network

RMSE Root Mean Square Error

RU Radio Unit

SDN Software Defined Network

SFC Service Function Chain

URLLC Ultra-Reliable Low-Latency Communication

vCPU virtual Central Processing Unit

VM Virtual Machine

VNF Virtual Network Function

XPU Crosshaul Processing Unit

1 Introduction

During the last five years, between 2016 and 2021, mobile data traffic has increased more than ten times, a trend that is expected to continue in the future [11, 12] with the increasing number of users and the growing field of Internet of Things (IoT). This unprecedented and exponential increment of traffic was predicted [13], as indeed occured, to affect the networks as they were defined and deployed, as well as the scope of the evolution between generations of telecommunications networks. Incremental evolution of previous-generation networks was not sufficient anymore to adapt to the new demands, and, thus, the necessity of a redefinition of the telecommunication networks appeared. The new 5th generation of telecommunications networks (5G), or 5G networks, as it is commonly known, has been designed from scratch during the last five years and it is currently starting to be commercially deployed.

Until the 4th generation of telecommunications networks (4G) a big part of the infrastructure was based on special-purpose, often proprietary, hardware, which means that in order to modify and adjust any component it was necessary to modify, reconfigure or change the hardware device itself. Furthermore, since the hardware was proprietary, every component had its own software developed by the manufacturer with its own rules, thereby severely limiting interoperability between hardware of different deployment moments and/or coming from different vendors. This issue made it hard for the networks to adapt to the changes at the required speed of traffic increment.

Around this point, several trends have been gaining importance in the field of telecommunications. One of them is the concept of *Software Defined Network (SDN)* [14, 15], which allows to control the vast majority of the network hardware, e.g., switches, with standardized protocols, thereby making them independent from the manufacturer-specific protocols. This new concept permits the hardware components of the networks to be interoperable independently of the manufacturer, and communicate through a common language. Additionally, having the components software-defined flexibilizes the networks, making them able to be modified remotely and without requiring to change the hardware, thereby attaining low costs and without deploying new infrastructure.

A key enabler for SDN is the softwarization and virtualization of the network functions, i.e., converting the traditional network function into a *Virtual Network Function (VNF)* through *Network Function Virtualization (NFV)*. The consequence of this virtualization is that new ecosystems appear, also thanks to the emerging trend of the service oriented market through *Over the Top (OTT)* companies, where traditional products have been mutating into services. An example of it is Netflix, coming from the traditional cinema market and moving into a service where a platform with such a big catalog of films that it eliminates the need for people to buy or rent them: films are always available at home on demand. This is a trend that is continuously growing, and it has been accelerated due to the Covid-19 pandemics, with customers getting used to have a wide number of services always at their disposal, to immediately satisfy their needs. This service oriented market contributes to the increasing traffic demand, as well as to incentivize users to have more devices connected to the networks and requesting content at the same time. Such a phenomenon further contributes to the overload of the network and makes it challenging to satisfy all the demands while guaranteeing the appropriate requirements for their delivery.

The paradigm of *cloud computing* [16, 17] has long been the most popular solution for companies to host and deliver their services. It is a big challenge for small companies to have their own servers to host the offered services, due to the costs of deployment, maintenance and security. In cloud computing, there are cloud providers offering the infrastructure, including maintenance and security, and the service providers rent part of it to host their services, which allows the end users to receive good services at reasonable prices.

However, cloud computing is centralized at few companies and locations around the world and all the traffic requests need to go through the same networks at some points, creating bottlenecks and overload at those segments of the network. In order to solve this problem, *edge computing* [18, 19] was proposed, envisioning to deploy part of those servers closer to the user, that is, at the edge of the network infrastructure. Thanks to edge computing, it is possible to deploy within the network infrastructure the most demanded services and offer better performance while eliminating a heavy part of the traffic in the overloaded segments in the center of the networks. Thus, edge computing offers a completely new perspective to the networks thanks to this reduction of load and the correspondant reduction of End-to-End (E2E) latency for services.

Another emerging paradigm complementing and reinforcing this cloud/edge tendency has been the *IoT* paradigm, whose main purpose is to connect all the devices people use in their every day life to the Internet [20]. This is another trend that contributes to the overload of the network and to increase the amount of continuous demands for services. To relieve such a load, it is envisioned to perform part of the processing of the demands at the devices themselves. Such *fog* devices have limited processing capacity, hence part

of the processing is performed in the cloud or edge servers, finding a balance between performance of the service and not overloading the networks. Such balance can be sought through *Artificial Intelligence (AI)*, in particular distributed *Machine Learning (ML)* algorithms, by defining the logical network topology of IoT devices around the learning task to perform, instead of adapting the learning algorithms to the network as in the traditional approach, in order to meet the desired learning performance. As a result, the network is optimized to guarantee a balance between performance and load.

Considering all the emerging features and trends, the 5th generation of networks has been designed thinking on the current and predicted future trends and embracing the emerging concepts making the new networks: (i) more flexible, to deal with the changes of demands within the already deployed components, thanks to SDN; (ii) more capable and resilient for the future, thanks to softwarization; and (iii) capable of processing data as well as transferring it, thanks to edge computing.

Moving the current networks into the new generation implies a process of evolution and incremental deployment from traditional networks composed of legacy equipment to 5G networks completely built with new generation equipment. An important technology in this transition is *Cloud or Centralized Radio Access Network (C-RAN)* [21], which enhances the traditional radio elements (e.g., evolved Node B (eNB) or next generation Node B (gNB) in 5G) to be split into a small footprint basic radio part (Distributed Unit (DU)), which may include lowest levels of the protocol stack, and a pool-able base band processing part (Central Unit (CU)). The deployment of this technology started in 4G and it will be massively adopted in 5G to reduce the costs of deployment and maintenance, CAPEX and OPEX, associated with the Radio Access Network (RAN), and to provide an additional performance gain due to the pooling of resources, thanks to edge computing, and the coordinated processing of signals from different cells by using Coordinated Multi-Point (CoMP) [22, 23].

This transition process will occur during the timelife of the current and already deployed equipment. During this period, when a legacy component will end its useful life, it will be replaced by an equivalent component of the new generation, ending the process with a network fully composed of new generation equipment.

Considering all the features required to develop and deploy the new generation of networks, new, complex decisions must be made and several challenges addressed in order to guarantee an optimal design and management. The biggest challenges to consider are: (i) to guarantee the coexistance of the new equipment and features deployed with the legacy one during the transition period; (ii) the resource allocation in the cloud/edge environment in order to guarantee the perfect balance between performance and service costs; and (iii) to support edge-based services, most importantly, AI and ML in distributed IoT networks.

In order to address these challenges, an optimization of the several parts of the network

must be performed. This is the point where this thesis contributes to the solution of the aforementioned challenges by:

- Optimizing the placements of core network resources in order to guarantee the coexistance of the new equipment and features deployed with the legacy one during the transition period and at the end of it. See Contribution 1 in Section 1.1.

- Optimizing the resource allocation in the cloud/edge environment in order to guarantee the perfect balance between performance and service costs. See Contribution 2 in Section 1.1.

- Optimizing the edge network in order to enhance and optimize the performance of the growing fields of IoT, ML and AI. See Contribution 3 in Section 1.1.

Finally, it is worth to notice that the contributions of this thesis have been designed in the context of several Horizon 2020 EU Research and Innovation programme (H2020) projects focused on 5G. In particular, considering some of the challenges of 5G-Crosshaul[1], 5G-TRANSFORMER[2], 5G-Coral[3] and 5GROWTH[4], as explained in more detail in Section 1.2. Thus, the contributions are a response to the needs that were appearing during the process of the design and developement of the 5G ecosystem. Hence, the solutions presented in this thesis are based on feedback from both academia and industry and they are developed satisfying the requirements of the future commercial real world deployments of those types of networks.

1.1. Main Contributions

The main contributions of this thesis have been published in 2 venues, of which 1 has been published in *Elsevier AdHoc Networks* and 1 published in *IEEE Transactions on Mobile Computing*, both indexed in Journal Citation Reports (JCR). Aldditionally, 1 contribution is submitted to IEEE Transactions on Networking and currently *under revision.* In details,

Contribution 1. *Core elements placement optimization.*
This contribution, that responds to the aforementioned challenge (i), is gathered in Chapter 2 and it can be divided into the development of: (i) an optimization framework for joint routing and resource placement, taking into account delay, capacity and path constraints, maximizing the degree of DU deployment while minimizing the supporting CUs; (ii) an efficient heuristic approach for solving the optimization problem in large

[1]H2020-ICT-2014-2 Grant Agreement no. 671598. URL: http://5g-crosshaul.eu
[2]H2020-ICT-2016-2 Grant Agreement no. 761536. URL: http://5g-transformer.eu
[3]H2020-ICT-2016-2 Grant Agreement no. 761585. URL: http://5g-coral.eu
[4]H2020-ICT-2018-3 Grant Agreement no. 856709. URL: https://5growth.eu

scale environments, allowing the operator to derive solutions aiming at maximizing the Air Bandwidth (that is boosted by properly splitting a RAN element) while minimizing the number of Crosshaul Processing Unit (XPU) (edge/cloud nodes hosting an array of CUs) by determining the placement of XPUs and the RAN elements that can be split into DUs and; (iii) a heuristic that allows the operator to compute the minimum number of XPUs and their placement for a given mixed RAN/C-RAN deployment.

- **Nuria Molner**, Antonio de la Oliva, Ioannis Stavrakakis, Arturo Azcorra. Optimization of an integrated fronthaul/backhaul network under path and delay constraints. Published in *Ad Hoc Networks, vol. 83, pp. 41-54, ISSN 1570-8705*, 2019. https://doi.org/10.1016/j.adhoc.2018.08.025

Contribution 2. *Resource allocation optimization on cloud/edge environments with mobile actors.*
This contribution, that responds to the aforementioned challenge (ii), is threefold and it is exposed in Chapter 3. First, the VNF placement problem is formulated as a cost-minimizing optimization problem for allocation of resources in an edge ecosystem, extending formulations in the state of the art imposing the radio coverage of mobile fog devices, and preventing that VNF deployments use fog devices that may run out of battery. Second, the optimization problem is solved by a novel heuristic algorithm that gets close to optimal results while tackling both radio coverage and battery restrictions of fog environments. Finally, the proposed algorithms are evaluated via extensive simulations on a real-world scenario, confirming the beneficial properties of the proposed solutions in terms of scalability, cost, and runtime.

- Balázs Németh, **Nuria Molner**, Jorge Martín Pérez, Carlos Jesús Bernardos, Antonio de la Oliva, Balázs Sonkoly. Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure. Published in *IEEE Transactions on Mobile Computing*, 2021. https://doi.org/10.1109/TMC.2021.3055426

Contribution 3. *Network optimization for distributed ML.*
This contribution, that responds to the aforementioned challenge (iii), is gathered in Chapter 4. It focuses on the edge and IoT ecosystems, developing the work around AI, in particular on distributed, supervised learning, and aims at filling the gap of defining the logical network topology of the nodes that cooperate towards the ML task around the ML task to perform and it can be divided into: (i) the development of a system model that can represent several relevant supervised ML tasks and account for the specific features of a 5G-and-beyond environment, most notably, the interaction between learning nodes

and information nodes; (ii) the formulatation of the problem of choosing the computing nodes and data sources, as well as the links connecting them, with the goal of minimizing the (monetary or energy) cost of the learning process, subject to prediction quality and learning time requirements; (iii) the development of an iterative algorithm that works over real-world topologies, closely matching optimal decisions and outperforming state-of-the-art alternatives.

- Francesco Malandrino, Carla Fabiana Chiasserini, **Nuria Molner**, Antonio de la Oliva. Network Support for High-performance Distributed Machine Learning. *[Submitted to IEEE Transactions on Networking, Under Revision].* https://arxiv.org/pdf/2102.03394.pdf

1.2. Thesis Outline

The rest of the thesis is organized in different chapters detailing the contributions mentioned in the previous subsection. It is structured from the core of the network to the users for an easier understanding, and it also corresponds to the timeline of improving each part of the network: from the need to improve the core of the network and offloading part of the load closer to the user, to decentralizing the networks and avoid bottlenecks in the center. Thus, this thesis as a whole presents an optimization of the network, supporting the transition from the previous generation of networks to the new one in an optimized manner.

Hence, Chapter 2 responds to some of the challenges emerged within the 5G-Crosshaul project, which aimed at developing a 5G integrated backhaul and fronthaul transport network enabling a flexible and software-defined reconfiguration of all networking elements in a multi-tenant and service-oriented unified management environment. This transport network targeted to flexibly interconnect distributed 5G radio access and core network functions, hosted on in-network cloud nodes, through the implementation of: (i) a control infrastructure using a unified, abstract network model for control plane integration (Crosshaul Control Infrastructure, XCI); (ii) a unified data plane encompassing innovative high-capacity transmission technologies and novel deterministic-latency switch architectures (Crosshaul Packet Forwarding Element, XFE). Thus, Chapter 2's main goal is to enhance the core part of the network by optimizing the RAN placements to move from legacy RAN equipments to flexible RAN ones. The work exposed in this chapter considers both the transition period from legacy equipment to new generation equipment, where both types of resources cohexist in the same transport stratum, as well as the optimization of the network RAN elements in a new generation network, where the optimal allocation of resources will help to reduce costs and increase the performance of the network both for end users and operators.

As a natural extension of this project, 5G-TRANSFORMER appeared aiming to transform the mobile transport network into an SDN/NFV-based Mobile Transport and Computing Platform (MTP), and bringing the "Network Slicing" paradigm into mobile transport networks by provisioning and managing MTP slices tailored to the specific needs of vertical industries. The technical approach was twofold: (i) enable vertical industries to meet their service requirements within customised MTP slices; and (ii) aggregate and federate transport networking and computing fabric, from the edge all the way to the core and cloud, to create and manage MTP slices throughout a federated virtualized infrastructure.

This project was followed by 5GROWTH, that leverages on the results of 5G-PPP Phase 2 projects where slicing, virtualization and multi-domain solutions for the creation and provisioning of vertical services were developed and validated. The vision is to empower verticals industries such as Industry 4.0, Transportation, and Energy with an AI-driven Automated and Sharable 5G End-to-End Solution that allows these industries to achieve simultaneously their respective key performance targets. Towards this vision, the project automates the process for supporting diverse industry verticals through (i) a vertical portal in charge of interfacing verticals with the 5G End-to-End platforms, receiving their service requests and building the respective network slices on top, (ii) closed-loop automation and SLA control for vertical services lifecycle management and (iii) AI-driven end-to-end network solutions to jointly optimize Access, Transport, Core and Cloud, Edge and Fog resources, across multiple technologies and domains.

In parallel to those projects, 5G-Coral was developed leveraging the pervasiveness of edge and fog computing in the RAN to create a unique opportunity for access convergence. This was envisioned by the means of an integrated and virtualised networking and computing solution where virtualised functions, context-aware services, and user and third-party applications were blended together to offer enhanced connectivity and better quality of experience. The proposed solution contemplated two major building blocks: (i) the Edge and Fog computing System (EFS) subsuming all the edge and fog computing substrate offered as a shared hosting environment for virtualised functions, services, and applications; and (ii) the Orchestration and Control System (OCS) responsible for managing and controlling the EFS, including its interworking with other (non-EFS) domains (e.g. transport and core networks, distant clouds, etc.).

Thus, Chapter 3, which is developed under the context of 5G-TRANSFORMER, 5G-Coral and 5GROWTH projects, gathers the challenges and architectures from those projects and aims to optimize the allocation of network functions in cloud/edge environments. This enables the possibility of offloading parts of the load from end-user terminals to edge servers, thus enabling new services that require low latencies and proximity to the originating devices. Adittionally, it considers networks managed by multiple tenants and its policy agreements. In particular, the application selected for the

experimentation and adjustments of the models is automated robots.

Finally, Chapter 4 has been developed within the context of 5GROWTH project, and aims at optimizing the part of the network that is closer to the users in order to enable emerging applications in the growing fields of IoT and ML. In this chapter, the behavior of the nodes composing the edge/IoT networks have been characterized in order to optimize their interactions to perform distributed ML tasks in an optimal manner under constraints of learning cost, learning time and learning accuracy.

2 Optimization of an Integrated Fronthaul/Backhaul Network

The 5th generation of telecommunications networks (5G) has been designed to accomodate the exponential increment of mobile data traffic in an environment of reduced revenues per user. This new generation of networks is characterized by an increment of available bandwidth to the users, providing the user with unprecedented speeds, fostering the evolution and deployment of new services which were not possible before. In addition, to increase the available resources per area unit, it is expected that 5G deployments will feature a higher capillarity, effectively increasing the density of the network. Through this densification, spectrum can be reused in a more effective way, paving the way towards higher bandwidths available to the end user as foreseen by the Cooper's law.

One key element to support the increased bandwidth to the user is the transport network that feeds the Radio Access Network (RAN). The future 5G RAN must support an unprecedented amount of traffic, with very stringent requirements in terms of latency and jitter. This will heavily impact on the design of the transport network feeding the RAN that must support more demanding transport requirements. In addition, RAN designers are looking for innovative ways of improving the performance achievable by the RAN. One of the mechanisms already identified in the literature is to split the radio elements (e.g., evolved Node B (eNB) or next generation Node B (gNB) in 5G) into a small footprint basic radio part (Distributed Unit (DU)), which may include lowest levels of the protocol stack, and a pool-able base band processing part (Central Unit (CU)). This technology, known as Cloud or Centralized Radio Access Network (C-RAN), will be massively used in 5G since it helps reduce the costs associated with the RAN and provide an additional performance gain due to the pooling of resources and the coordinated processing of signals from different cells. The disadvantage of the C-RAN technology is the need for a high bandwidth and low delay network connection between the radio and processing parts. This network segment has traditionally been known as fronthaul and has recently been the subject of a lot of research on protocols (Common Public Radio Interface (CPRI) [24], Enhanced Common Public Radio Interface (eCPRI) [25]) and analysis of the possible functional splits of the protocol stack [26, 27]. The evolution

of C-RAN technologies from a serial transmission (CPRI) to fully packetized protocols (eCPRI) allows for the integration of fronthaul and backhaul networks.

4th generation of telecommunications networks (4G) networks started deploying the C-RAN concept in the already deployed networks. The seminal paper [21] proposes the use of fronthaul as a mechanism to reduce the CAPital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX) due to reduced expenses on the site antenna. Later, fronthaul was also proven useful to improve the performance of the air interface due to the easiness of synchronization of the CUs, allowing the use of Coordinated Multi-Point (CoMP). The main problem with the C-RAN approaches of the moment was the use of CPRI (the predominant fronthaul technology) which was using a serial transmission, not encapsulated, requiring of point to point high bandwidth and dedicated fibers between the DUs and CUs. This increases the cost of management and operation of the network, since the operator has to face the operation of two different networks, one based on packets (the normal backhaul or transport network) and a second one using a completely different technology. This fact triggered a change in how standardization bodies were focusing on C-RAN for 4G and 5G, working on solutions based on packets that can use standard switching technologies.

In this context, the operator had to face a very complex and challenging network to manage, which was no longer divided into RAN, transport and core domains but places different RAN and core elements within data-centers distributed in the transport network. This new network, which is being referred to as Crosshaul [28], encompasses the front- and back-hauling network segments and requires new approaches for the planning and operation of the network. For this new network, operators need to decide not only on the placement of each radio node but also whether it should be split, where the higher layers of such a split should be placed and how the resulting traffic sources affect the rest of the links. This C-RAN approach was possible thanks to the deployment of intelligence at the edge of the network, in the form of micro data-centers that can host virtual network functions including C-RAN baseband processing, i.e. the higher layers of the functional split. Hence, the edge inteligence has been a key component to transform the network from a mere data pipe into a smart application hosting environment.

Some background information on each of these technologies, as well as related works on the optimization of the resulting converged network are introduced in the following paragraphs in order to understand better the challenges faced in order to optimize the Crosshaul network.

The deployment of the 5G RAN was expected to capitalize on the concept of C-RAN [29]. In C-RAN systems the base station functionality is split at a certain point of the protocol stack (such as, for example, the physical layer), and the upper part is moved to a central unit, typically within or co-located with an edge data center facility [22,30]. A C-RAN system consists at least of the following three main components: the distributed

units implementing the radio functions, the central processing units which are typically aggregated in pools, and the network interconnecting them, typically referred to as fronthaul [31]. The different points in the protocol stack that determine the separation of the functions processed in central or distributed units define what is referred to as *functional splits* [26, 32]. The implementation of a given functional split uniquely defines the properties of the system design [27]. The complexity, benefits and drawbacks of the distributed and centralized units depend on the functional split chosen.

By the moment, the most common functional split corresponded to the division at the physical layer (as implemented by the CPRI). CPRI is a non-packetized serial protocol which cannot be integrated with other packetized transmissions unless a circuit (e.g., a wavelength) is reserved for it. Hence in the work performed for this section the packetized evolution of CPRI (eCPRI) is considered, which packetizes the I/Q samples in an Ethernet compatible way.

The C-RAN approach benefits significantly from virtualization. By virtualizing and centralizing the baseband processing of multiple cells, an operator is able to better manage inter-cell interference and traffic load, as well as reduce overall costs. At the same time, by co-locating multiple centralized units pooling gains appear and scaling up the system when RAN demands increase is facilitated. The trend towards the deployment of Edge data centers, aiming at hosting delay constrained applications, such as augmented reality, opened the door for deploying C-RAN centralized units in virtualized infrastructure at the edge of the network.

The efficient design and operation of such an environment requires a joint consideration of routing, placement of Edge data center and C-RAN cell deployment in the presence of traffic with multiple priorities and strict deadlines and, thus, extending the state of the art. Work related to delay constrained routing may be found in [33], where a Djikstra shortest path algorithm that uses link delay as the weight of a link is proposed. In [34], the authors propose a heuristic algorithm based on the minimum delay path and shortest path for networks with time-dependent edge-lengths. Heuristic algorithms to derive the minimum cost (delay) tree between the source and the destination can also be found in [35]. Routing with delay constraints and analysis of delay variation has also been studied for multicast networks in [36]. The M/G/1 queuing model is one widely adopted for modeling the queuing delay in network nodes. For instance, in [37], the authors introduce fixed parameters for the arrival rates and exit rates (λ and μ) in order to make the problem tractable. In [38–41] the authors approximate the delay with non linear equations. In [42] authors deal with the M/G/1 queueing model with priorities for the problem of the mixed fronthaul and backhaul networks proving that it is a good approximation for this traffic. Authors in [43] used M/M/1 queueing model for Virtual Network Function (VNF) placement problems without several priorities and dealing with non-linear equations. Finally, works on un-splittable flow problems, such as [44], [45]

and [46], develop heuristic algorithms for NP-hard problems in the general case, but these papers do not consider networks integrating traffic with different priorities.

The unified problem considered in this work also addresses the optimal placement of the Edge data centers, further enhancing the applicability and complexity of the work. This problem is related to the problem of the Virtual Network Embedding (VNE) and the problem of placing chains of virtual functions [47]. In this work, we consider the problem of determining the optimal placement of the data centers subject to the transformation of fronthaul flows (with special characteristics) into backhaul flows. Finally, work on transporting fronthaul traffic over Institute of Electrical and Electronics Engineers (IEEE) 802.1Q switches has also been recently carried out. Works such as [48] conclude that such a transport is possible through the extension provided by 802.1Qbu, 802.1Qbv and by employing buffers at the receivers.

Although previous work, as the one exposed above shows that RAN centralization has many advantages and has been tackled through multiple perspectives, the work presented in this chapter, to the best of our knowledge, was the first to study the complete problem of the joint optimization of C-RAN deployment and Edge data center placement, taking into consideration the stringent fronthaul flow deadlines and the accumulated delay in the switching devices.

In summary, a new methodology was necessary for the planning and operational optimization of the network, focusing on the integrated transport of fronthaul and backhaul traffic, the placement of the pool-able resources containing the radio nodes' higher layers (CUs) and the overall delay achievable in the network. This new methodology is explained in the following sections. Section 2.1 introduces the problem in its context. Section 2.2 presents a mathematical formulation that maximizes the DU deployment and yields the optimal number of data-centers containing the pool-able CUs and their location, while taking into account the stringent delay requirements of the resulting fronthaul traffic by incorporating proper queuing models. The general formulation is non-convex and non-linear and since non-tractable (as shown in Appendix B.1), certain approximations are also introduced in Section 2.2 to yield a tractable formulation that can provide with reasonable computational complexity for the optimal results, at least for the case of small scale environments. For larger-scale environments, a computationally tractable heuristic is introduced in Section 2.3 that provides for an efficient (though not necessarily optimal) solution, achieved in reduced time. In Section 2.4 the developed approaches are validated and applied to both small- and large-scale (production) networks and some results are presented.

2.1. System Model

Figure 2.1 depicts the general environment considered in this chapter, where a number of sources are connected to the Internet (where the potential destination of a source flow

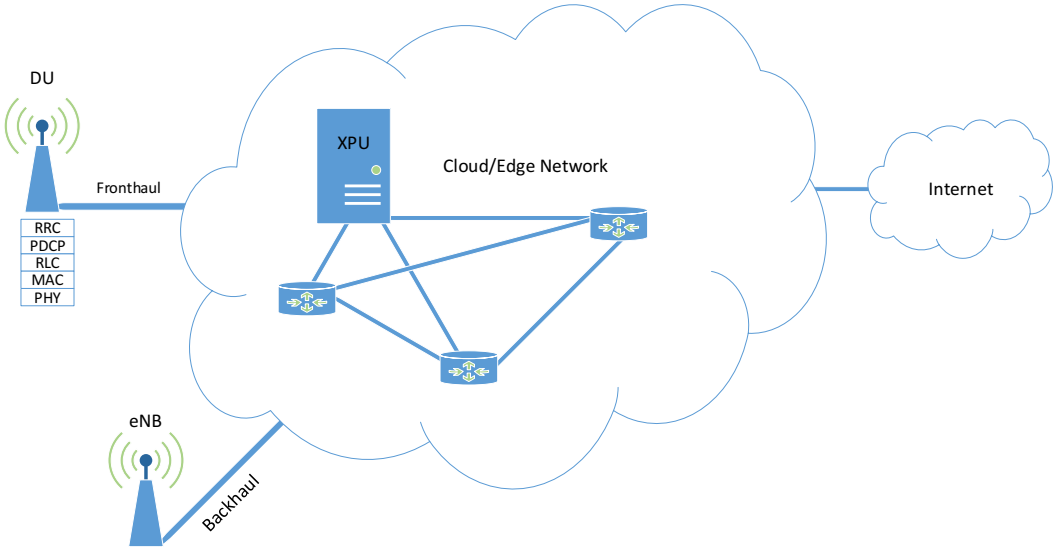


Figure 2.1: The general network environment

is assumed to reside) through an edge/cloud access network. The sources represent either the base station of a classical RAN (e.g. an eNB node) or just a DU of a C-RAN whose upper layer functions are executed somewhere in the Edge/Cloud network by the CU. The traffic flow departing a DU source (fronthaul) is typically of high rate (e.g., 0.9Gbps), although its exact bandwidth depends on the functional split used and the channel bandwidth used (among others, depending on the functional split). On the other hand, the traffic flows departing an eNB node (backhaul) are of much lower rate, can be several and up to a maximum total rate of typically 0.15Gbps under full utilization of the air medium of the eNB node (depending on Multiple-Input Multiple-Output (MIMO) and bandwidth configuration of the eNB). This topology of mixed RAN and C-RAN components is expected to dominate for the foreseen future, as a progressive migration from a RAN to a C-RAN dominated world takes place for the benefits discussed earlier.

The main objective in this work was to provide for efficient or optimal designs of such mixed RAN/C-RAN environments. These mixed environments emerge as operators attempt to maximize their adoption of the C-RAN technology in the most effective way, subject to the constraints imposed by the available supporting infrastructure (explained below). Or, such environments may emerge in a more dynamic (operation-level) case, where operators may switch on or off DUs or aggregate them in a lower number of CUs according to the demand to reduce OPEX; re-optimization of the resulting mixed RAN/C-RAN environment is then needed as well.

To facilitate the discussion on the formulation of the optimization problem in this section, an originally all RAN environment will be considered and seek to optimize

the degree of migration towards a mixed RAN/C-RAN environment by minimizing the number of (remaining) RAN components and optimizing the C-RAN deployment in the resulting environment. That is, maximize the number of RAN components that are replaced by a DU, while maximizing the degree of aggregation/pooling of the CU components by minimizing the number of locations hosting the CUs. The latter pooling provides for some wireless capacity enhancement through coordinated signal processing and reduces costs for the operator. All the co-located CUs will be considered as components of a single data-center, to be referred to as Crosshaul Processing Unit (XPU). The capacity of an XPU is considered to be equal to the number of CUs available/implemented in the specific location.

The main challenge in this migration is due to the fact that the (single) flow departing a DU (referred to as fronthaul): (a) has a (much) higher rate compared to that of the original eNB; (b) it must be routed towards a XPU (containing a CU) facility to be processed first, before it is transformed into backhaul flow(s) and be routed from there towards the destination; (c) it has stringent delay requirements for reaching the CU facility. In addition, the location of the CU facility - where a fronthaul flow is forwarded to - needs to be determined by minimizing the number of such locations (maximizing CU pooling), or minimizing the number of XPU facilities deployed. These challenges are incorporated and addressed through the optimization formulation developed and solved in this work.

As indicated earlier, this work considers a single type of packetized fronthaul flow (eCPRI), although the model presented could be used for any other kind of packetized fronthaul. This flow exists only over the path between the generating DU and the associated XPU and it becomes a standard backhaul flow over the path between the XPU and its destination. Such standard backhaul flows are also generated by the classical RAN nodes (e.g., eNBs) and will coexist with fronthaul flows. The key parameters of the two types of flows that are considered in this chapter are given in Table 2.1 from [24] and [49]. A quick cite of the two flows shows that the fronthaul flow is of much higher rate and has much less delay requirements. This large asymmetry in these parameters leads to a number of observations and design considerations.

As a fronthaul flow has very stringent delay requirements (compared to a backhaul flow) it will be treated as a class of traffic of (non-preemptive) priority 1. A severe consequence of changing an eNB node to a DU+CU is that a high rate increase will be observed in the links departing the eNB/DU node towards the location of the associated CU (XPU to be determined). This severe increase in the rates along with the stringent delay requirement over that part of the network constraint the migration from a RAN to a fully C-RAN environment. Finally, the high rate asymmetry makes the approximation of considering in our optimization formulation a single (as opposed to multiple) backhaul flow departing the associated CU a reasonable one, as it is expected to have only minor

impact on the solution of the optimization problem, which is primarily shaped by the pre-XPU part of the access network.

Flow	Value	Delay	Class
fronthaul (eCPRI)	900 Mbps	250 μs	1
backhaul (fronthaul after XPU usage)	150 Mbps	100 ms	2
backhaul	15 Mbps	100 ms	2

Table 2.1: Parameters of fronthaul and backhaul traffic considered.

The notation for the various parameters employed in the formulation of the optimization are shown in Table 2.2. Notice that a fronthaul source generates one flow, while a backhaul source generates up to K flows, denoted by k, $0 \leq k \leq K$. It is also worth to mention that the work performed in this chapter assumes values for the air interface in line with 4G deployments, since in the moment the work was developed there were no deployments of 5G C-RAN or even 5G air interfaces. However, as 5G is deployed, operators will need to enhance their transport networks, that now need to transport much more capacity to the RAN. In that moment, these values may differ from the ones we chose but this is just a parameter of the model, which can be easily changed and simulations re-run, yielding to different results in terms of total air capacity or number of XPUs. Therefore, the operators will only have to perform the simulations with the new values, but the main contribution, the mathematical model, heuristics and tendencies on the results will be the same.

Parameters	Definition
$\mathcal{F}$	set of sources
f^l	rate of fronthaul source/flow l, $l \in \mathcal{F}$ (Mbps)
f^{l+}	rate of fronthaul flow l, $l \in \mathcal{F}$, after using its CU (Mbps)
b_k^l	rate of flow k of backhaul source l, $l \in \mathcal{F}$ (Mbps)
p^l	packet size of fronthaul flow l (0.012 Mbits=1500bytes)
p_k^l	packet size of flow k of backhaul source l (0.012 Mbits)
D^{l-}	delay constraint of fronthaul flow l, to reach its CU
D^l	delay constraint of fronthaul flow l
D_k^l	delay constraint of flow k of backhaul source l
$\mathcal{E}$	set of links of access/edge network
c_{ij}	capacity of link (i, j) (Mbps)
μ_{ij}	capacity of link (i, j) (in packets/sec)
$L_{i,j}$	length of link (i, j)
$\mathcal{X}$	set of available XPU facilities
N_r	capacity of XPU r, $r \in \mathcal{X}$ (in CUs)

Table 2.2: Parameters employed in the formulation of the optimization.

Let $\mathbb{I}_{**}^{*}$ denote a binary variable (b.v.) assuming the value 1 if an event specified

through ** and * has occurred, and 0 otherwise. The main binary and other variables employed in the formulation of the optimization problem are defined in Table 2.3.

Variables	Definition
$\mathbb{I}^l$	b.v. indicating if source (flow) l is a DU (fronthaul), $l \in \mathcal{F}$ (if not, it is an eNB)
$\mathbb{I}^l_{ij}$	b.v. indicating if link (i,j) is used by fronthaul flow l, before reaching its CU, $l \in \mathcal{F}$
$\mathbb{I}^{l+}_{ij}$	b.v. indicating if link (i,j) is used by fronthaul flow l, after leaving its CU, $l \in \mathcal{F}$, (and has then been transformed into a backhaul flow)
$\mathbb{I}^l_{ijk}$	b.v. indicating if link (i,j) is used by flow k of backhaul source l, $l \in \mathcal{F}$
$\mathbb{I}^{XPU}_r$	b.v. indicating if XPU r is used
$\mathbb{I}^{XPU,l}_r$	b.v. indicating if XPU r is used by flow l, $l \in \mathcal{F}$
λ^n_{ij}	rate of priority class n entering link (i,j) (in packets/sec)
d^l	delay of fronthaul flow l, $l \in \mathcal{F}$
d^{l-}	delay of fronthaul flow l, $l \in \mathcal{F}$, until it reaches its CU
d^l_k	delay of flow k of backhaul source l, $l \in \mathcal{F}$

Table 2.3: Variables employed in the formulation of the optimization.

2.2. Problem Formulation

In this subsection we employ the notation presented above to formulate the optimization problem by presenting the objectives, the various traffic and resource constraints and the supporting equations. The treatment of the delay constraints is deferred to the next subsection.

A set of locations of the sources of traffic are given (whose type fronthaul/backhaul is to be determined), along with the Edge/Cloud network topology (link capacities and lengths), the set of network nodes which are capable of hosting an XPU facility and the maximum number of XPUs to be possibly deployed. The solution of the optimization problem will determine the type of each one of the sources, while maximizing the number of fronthaul traffic sources and minimizing the number of XPUs deployed whose location is also determined. As discussed earlier a source can be accepted as a fronthaul source only if it is supported by a non-dedicated XPU, to yield some pooling gain; that is, if its CU can be hosted in an XPU (location) that supports at least one more CU serving another fronthaul source. The available link capacity between the sources and the XPU location will be the constraining factor determining whether CU collocation is possible or not.

To ensure that the maximum number of DU sources is determined under the constraint that all of them are supported by non-dedicated XPUs, the following objective function

is defined for some $g > 1$.

$$\max \left\{ g \cdot \sum_{l} \mathbb{I}^{l} - \sum_{r} \mathbb{I}_{r}^{XPU} \right\} \tag{2.1}$$

Notice that the above objective function prescribes the following gains or penalties: (a) maximizes the number of DUs in the network, (b) minimizes the number of XPUs that are used for those DUs, and (c) if a source is an eNB source, it is not associated with any XPU and, thus, it does not contribute to the objective function (its gain is zero). Notice that, based on the above, an eNB source is preferable over a DU source supported by a dedicated XPU, because in Equation (2.10) we impose that the XPUs cannot be dedicated to one DU. Similarly, a DU supported by a non-dedicated XPU is preferable over (yields a higher gain than) an eNB source. Consequently, the objective in Equation (2.1) ensures that the solution to the maximization will not contain any DU source that is supported by a dedicated XPU, the number of DUs will be maximized and the number of XPUs will be minimized. The latter is the case since it can be easily shown that the resulting gain associated with M_1 non-dedicated XPUs is higher than that associated with M_2 non-dedicated XPUs for $M_1 < M_2$, for the same number of DUs.

In order to accommodate the various resource constraints and other specific requirements, various equations and constraints are introduced and summarized in Table 2.4.

Constraint	**Function**
Source Constraints Eqs. (2.2) and (2.3)	To determine if a source is an eNB or a DU
Link Capacity Equation (2.4)	To assure the traffic that uses a link does not surpass its capacity
Destination Constraint Equation (2.5)	To assure all the traffic reaches its destination
XPU Constraints Eqs. (2.6) - (2.17)	To assure the fronthaul flows are processed in a XPU
Node Constraints Eqs. (2.20) - (2.22)	To assure non loss of traffic
Single Path Eqs. (2.23) - (2.25)	To assure single path for all the flows in the network
Delay Constraints Eqs. (2.26) - (2.39)	To compute the delay of the traffic

Table 2.4: Summary of main constraints of the optimization.

The requirement for single path routing implies that all the traffic of any source leaves the source through a single link, as captured by Equations (2.2) and (2.3).

$$\mathbb{I}^l = \sum_j \mathbb{I}^l_{lj} \quad \forall\ l \in \mathcal{F} \quad \forall (l,j) \in \mathcal{E} \tag{2.2}$$

$$(1 - \mathbb{I}^l) = \sum_j \mathbb{I}^l_{ljk} \quad \forall\ flow\ k\ of\ source\ l \in \mathcal{F} \tag{2.3}$$

The link capacity constraints are captured by Equation (2.4).

$$\sum_l f^l \cdot \mathbb{I}^l_{ij} + \sum_l f^{l+} \cdot \mathbb{I}^{l+}_{ij} + \sum_{l,k} b^l_k \cdot \mathbb{I}^l_{ijk} \leq c_{ij} \quad \forall (i,j) \in \mathcal{E} \tag{2.4}$$

As the design space in this chapter is the Edge/Cloud network and the deployed C-RAN, it is assumed that the destination of each flow is beyond this network and is referred to as "the Internet"; let the superindex in j^{Int} mark a (fictitious) node representing that Internet destination of the flow. Notice also that a fronthaul flow always reaches its (Internet) destination as a backhaul flow. Equation (2.5) captures the balance of flows entering and exiting the edge/access network.

$$\sum_{l,k} \mathbb{I}^l_k + \sum_l \mathbb{I}^l = \sum_{i,j^{Int},l,k} \mathbb{I}^l_{ij^{Int}k} + \sum_{i,j^{Int},l} \mathbb{I}^{l+}_{ij^{Int}} \tag{2.5}$$

Equation (2.6) captures the requirement that a fronthaul flow must be routed through a node hosting an XPU.

$$\mathbb{I}^l = \sum_r \mathbb{I}^{XPU,l}_r \quad \forall\ l \in \mathcal{F} \tag{2.6}$$

Equation (2.7) imposes the requirement that a fronthaul flow l, $l \in \mathcal{F}$, that uses XPU r, $r \in \mathcal{X}$, must use one (and only one) of the incoming links attached to the node hosting the XPU (referred to as j^{XPU_r}); if flow l does not use this XPU, it may still use one of its incoming links.

$$\mathbb{I}^{XPU,l}_r \leq \sum_i \mathbb{I}^l_{ij^{XPU_r}} \quad \forall r \in \mathcal{X}, \quad \forall\ l \in \mathcal{F} \tag{2.7}$$

Equation (2.8) captures the capacity constraint of an XPU r

$$\sum_l \mathbb{I}^{XPU,l}_r \leq N_r \quad \forall r \in \mathcal{X} \tag{2.8}$$

An XPU is considered to be utilized as long as at least one fronthaul source uses it. On the other hand, an XPU has to be used by at least two fronthaul sources. These constraints are captured by Equations (2.9) and (2.10).

$$\mathbb{I}^{XPU,l}_r \leq \mathbb{I}^{XPU}_r \quad \forall r \in \mathcal{X} \tag{2.9}$$

$$2 \cdot \mathbb{I}_r^{XPU} \leq \sum_l \mathbb{I}_r^{XPU,l} \quad \forall r \in \mathcal{X} \tag{2.10}$$

A fronthaul flow l entering node j^{XPU_r} hosting XPU r, will appear at an outgoing link as either a transformed backhaul flow if it is processed by XPU r, or as the same fronthaul flow otherwise; this is captured by Equation (2.11). A fronthaul flow l that has been transformed into a backhaul flow (having been processed by another XPU) entering node j^{XPU_r} hosting XPU r, will appear unmodified at an outgoing link; this is captured by Equation (2.12).

$$\sum_i \mathbb{I}_{ij^{XPU_r}}^{l} = \left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l+}\right) \cdot \mathbb{I}_r^{XPU,l} + \left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l}\right) \cdot \left(1 - \mathbb{I}_r^{XPU,l}\right) \tag{2.11}$$

$$\sum_i \mathbb{I}_{ij^{XPU_r}}^{l+} = \left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l+}\right) \cdot \left(1 - \mathbb{I}_r^{XPU,l}\right) \tag{2.12}$$

Notice that the above constraints are non-linear and would increase the complexity of the optimization machinery to be employed. Thus, in order to keep the computational complexity low and use linear programming tools, we linearize these constraints as described in Appendix A.1. To facilitate the presentation, we use the notation shown below for the two terms in the right hand side of Equation (2.11), which is rewritten as in Equation (2.13)

$$\sum_i \mathbb{I}_{ij^{XPU_r}}^{l} = \alpha_{rf^{l+}} + \alpha_{rf^{l}} \tag{2.13}$$

where $\alpha_{rf^{l+}}$ is bounded from above and below by the linear expressions shown in Equations (2.14) and (2.15) and $\alpha_{rf^{l}}$ is bounded from above and below by Equations (2.16) and (2.17). Notice that it is possible to linearize $\alpha_{rf^{l+}}$ and $\alpha_{rf^{l}}$ because all the variables involved in the above bounds are binary. The exact values of $\alpha_{rf^{l}}$ and $\alpha_{rf^{l+}}$ are completely determined from the bounds shown in Equations (2.14), (2.15), (2.16) and (2.17), all involving binary variables.

$$\alpha_{rf^{l+}} \leq \left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l+} + \mathbb{I}_r^{XPU,l}\right)/2 \tag{2.14}$$

$$\left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l+} + \mathbb{I}_r^{XPU,l} - 1\right)/2 \leq \alpha_{rf^{l+}} \tag{2.15}$$

$$\alpha_{rf^{l}} \leq \left(\sum_i \mathbb{I}_{j^{XPU_r}i}^{l} + 1 - \mathbb{I}_r^{XPU,l}\right)/2 \tag{2.16}$$

$$\left(\sum_i \mathbb{I}^l_{jXPU_r i} + 1 - \mathbb{I}^{XPU,l}_r - 1\right)/2 \leq \alpha_{rf^l} \tag{2.17}$$

The additional constraints (2.18) and (2.19) linearize the constraint (2.12) involving binary variables.

$$\sum_i \mathbb{I}^{l+}_{ijXPU_r} \leq \left(\sum_i \mathbb{I}^{l+}_{jXPU_r i} + 1 - \mathbb{I}^{XPU,l}_r\right)/2 \tag{2.18}$$

$$\left(\sum_i \mathbb{I}^{l+}_{jXPU_r i} + 1 - \mathbb{I}^{XPU,l}_r - 1\right)/2 \leq \sum_i \mathbb{I}^{l+}_{ijXPU_r} \tag{2.19}$$

Equations (2.20) to (2.22) capture the requirement of flow continuity in the intermediate nodes of the access/edge network.

$$\sum_i \mathbb{I}^l_{ij} = \sum_i \mathbb{I}^l_{ji} \tag{2.20}$$

$$\sum_i \mathbb{I}^{l+}_{ij} = \sum_i \mathbb{I}^{l+}_{ji} \tag{2.21}$$

$$\sum_i \mathbb{I}^l_{ijk} = \sum_i \mathbb{I}^l_{jik} \tag{2.22}$$

Equations (2.23), (2.24) and (2.25) capture the requirement of single path routing assumed in this chapter.

$$\sum_j \mathbb{I}^l_{ij} \leq \mathbb{I}^l \quad \forall i\ node, \quad \forall\ fronthaul\ flow\ l \tag{2.23}$$

$$\sum_j \mathbb{I}^{l+}_{ij} \leq \mathbb{I}^l \quad \forall i\ node, \quad \forall\ fronthaul\ flow\ l \tag{2.24}$$

$$\sum_j \mathbb{I}^l_{ijk} \leq 1 - \mathbb{I}^l \quad \forall i\ node, \quad \forall\ backhaul\ flow\ k\ of\ source\ l \tag{2.25}$$

Finally, we also impose that some of the variables of the model are binary,

$$\begin{aligned}
\mathbb{I}^l &\in \{0,1\} \quad \forall f^l\ flow \\
\mathbb{I}^l_k &\in \{0,1\} \quad \forall b^l_k\ flow \\
\mathbb{I}^l_{ij}, \mathbb{I}^{l+}_{ij} &\in \{0,1\} \quad \forall (i,j)\ link,\ \forall f^l\ flow \\
\mathbb{I}^l_{ijk} &\in \{0,1\} \quad \forall (i,j)\ link,\ \forall b^l_k\ flow \\
\mathbb{I}^{XPU,l}_r &\in \{0,1\} \quad \forall r\ XPU,\ \forall f^l\ flow \\
\mathbb{I}^{XPU}_r &\in \{0,1\} \quad \forall r\ XPU
\end{aligned}$$

A major challenge in the general C-RAN deployment problem considered in this chapter is to accommodate the stringent delay requirements of the fronthaul traffic (see Table 2.2). Consequently, the delay constraints should also be incorporated in the optimization. The following subsection presents the formulation of the (non-linear) delay constraints and the derivation of linear approximations to allow for employing linear programming solution tools.

2.2.1. Incorporation of Delay constraints

There are 3 delay components that every packet experiences between the completion of its arrival to a network node, say i, and that to the next node along its path, say node j. These delays will be attributed to link/port (i, j) and are determined by the transmission capacity (referred to as the transmission delay), the length (referred to as the propagation delay) and the queuing phenomena (referred to as the queuing delay) of link (i, j). Other sources of additional delay, such as that of processing time at the nodes or an XPU, will be considered to be relatively small and will be ignored. The consideration of the aforementioned 3 components will establish the impact of distances, capacities and traffic loads in a C-RAN environment, which is of utmost interest to the network operators. The transmission and propagation delays are easily derived, as the packet sizes, link distances and transmission capacities are assumed to be known. The challenge here is to derive the queuing delay in a way that is easily incorporated in the optimization formulation.

As the challenge in the C-RAN deployment is to ensure that the stringent delay constraints of the fronthaul flows are satisfied, a priority queuing scheme will be adopted giving non-preemptive priority to fronthaul packets over the backhaul ones. Although the sizes of the packets are considered to be fixed, we will adopt a queuing model with general service time, to keep the treatment more general. On the other hand, the arrival process will be considered to be Poisson, which is considered to be a reasonable model capturing the superposition of independent packet streams arriving over different input links to an outgoing link. Thus, we will consider an M/G/1 queuing model with 2 priority classes [50]. The packet arrival rates of priority n, λ_{ij}^n, can be expressed by Equation (2.26).

$$\lambda_{ij}^1 = \sum_l f^l/p^l \cdot \mathbb{I}_{ij}^l, \quad \lambda_{ij}^2 = \sum_l f^{l+}/p^l \cdot \mathbb{I}_{ij}^{l+} + \sum_{k,l} b_k^l/p_k^l \cdot \mathbb{I}_{ijk}^l \tag{2.26}$$

Let $\rho_{ij}^n = \lambda_{ij}^n/\mu_{ij}$ denote the traffic intensity at link (i, j) due to the incoming flows of priority class n. The classical queuing results provide for the mean queuing delay of packets of priority n, denoted by W_{ij}^n, described in Equations (2.27) and (2.28), where R_{ij} describes the mean remaining time till the completion of the transmission of the packet being transmitted upon a packet's arrival to node i; notice that since the packet size and link capacities are fixed, the second moment of the service time in Equation (2.28) is equal

to and has been replaced by $1/\mu_{ij}^2$.

$$W_{ij}^n = \frac{R_{ij}}{(1-\rho_{ij}^1-\ldots-\rho_{ij}^n)(1-\rho_{ij}^1-\ldots-\rho_{ij}^{n-1})} \tag{2.27}$$

$$R_{ij} = \frac{\sum_n \lambda_{ij}^n/\mu_{ij}^2}{2} \tag{2.28}$$

Considering the packet transmission, queuing and propagation delay components over all links traversed by the flow (given by Equation (2.27)), the delay of a fronthaul packet in reaching its XPU is derived and given by Equation (2.29). This delay is subject to the most stringent constraint, as shown in Table 2.1. Considering the corresponding delay components similarly, the delay of a packet generated by a fronthaul source in reaching its destination is given by Equation (2.30), considering also its path (as a backhaul packet) from its XPU to its destination. Similarly, the delay experienced by a packet generated by a backhaul source is derived and given by Equation (2.31). Notice that a fronthaul packet has priority $n = 1$ while a backhaul packet has priority $n = 2$.

$$d^{l-} = \sum_{i,j} \frac{\mathbb{I}_{ij}^l}{\mu_{ij}} + \sum_{i,j} W_{ij}^1 \cdot \mathbb{I}_{ij}^l + \sum_{i,j} \frac{L_{ij}}{vl} \cdot \mathbb{I}_{ij}^l \tag{2.29}$$

$$d^{l} = d^{l-} + \sum_{i,j} \frac{\mathbb{I}_{ij}^{l+}}{\mu_{ij}} + \sum_{i,j} W_{ij}^2 \cdot \mathbb{I}_{ij}^{l+} + \sum_{i,j} \frac{L_{ij}}{vl} \cdot \mathbb{I}_{ij}^{l+} \tag{2.30}$$

$$d_k^l = \sum_{i,j} \frac{\mathbb{I}_{ljk}^l}{\mu_{ij}} + \sum_{i,j} W_{ij}^2 \cdot \mathbb{I}_{ijk}^l + \sum_{i,j} \frac{L_{ij}}{vl} \cdot \mathbb{I}_{ijk}^l \tag{2.31}$$

Notice that the delay expressions above include the non-linear functions W_{ij}^n (with respect to ρ_{ij}^n or λ_{ij}^n) which would not allow for the incorporation of linear programming tools for the solution of our optimization problem, even if most of the variables involved are binary. To address this problem, a linear approximation based on Taylor expansion along with an iterative procedure are adopted and are described next.

By considering the first terms of a Taylor expansion of W_{ij}^n around some point $(\rho_{ij}^{1,0}, \rho_{ij}^{2,0})$ we get the approximation $\tilde{W}_{ij}^n$ shown in Equation (2.32).

$$\tilde{W}_{ij}^n = \frac{1}{2\mu_{ij}} \cdot \left\{ a_{n0} + a_{n1} \cdot (\rho_{ij}^1 - \rho_{ij}^{1,0}) + a_{n2} \cdot (\rho_{ij}^2 - \rho_{ij}^{2,0}) \right\} \tag{2.32}$$

By substituting W_{ij}^n by $\tilde{W}_{ij}^n$ in Equations (2.29), (2.30) and (2.31) we end up with some products of variables. Since one of them is bounded (ρ_{ij}^n), and the other one is binary ($\mathbb{I}_{ij}^l$ or $\mathbb{I}_{ij}^{l+}$ or $\mathbb{I}_{ijk}^l$) we can linearize such products by introducing some additional variables, as expressed in Appendix A.2, and shown for the case of the product in Equation (2.29) next.

$$y_{ij}^{l,n} = \mathbb{I}_{ij}^{l} \cdot \rho_{ij}^{n} \tag{2.33}$$

$$y_{ij}^{l,n} \leq \mathbb{I}_{ij}^{l} \tag{2.34}$$

$$y_{ij}^{l,n} \leq \rho_{ij}^{n} = \frac{\lambda_{ij}^{n}}{\mu_{ij}} \tag{2.35}$$

$$\mathbb{I}_{ij}^{l} + \rho_{ij}^{n} - 1 \leq y_{ij}^{l,n} \tag{2.36}$$

Finally, the following constraints are imposed on the delay of the fronthaul packets in reaching their XPU and their destination and the backhaul packets in reaching their destination.

$$d^{l-} \leq D^{l-} \cdot \mathbb{I}^{l} \tag{2.37}$$

$$d^{l} \leq D^{l} \cdot \mathbb{I}^{l} \tag{2.38}$$

$$d_k^{l} \leq D_k^{l} \cdot \left(1 - \mathbb{I}^{l}\right) \tag{2.39}$$

Since the linearized formula for the queuing delays shown in Equation (2.32) requires some (arbitrary) initial input for the class 1 and 2 traffic, some discussion on the impact of the particular approximation on the accuracy of the solution derived through the optimization framework is in order. A (first) solution to the optimization problem is obtained by considering an arbitrary initial value for the loads $(\rho_{ij}^{1,0}, \rho_{ij}^{2,0})$ in Equation (2.32). This solution determines also the loads and delays associated with all links. In the sequel, these loads are used for the calculation of the link delays based on the exact formula in Equation (2.27) and the result is compared with that returned by the solution to the optimization problem. If the deviation exceeds some threshold, then the new loads are considered as the initial values in Equation (2.32) and a new solution to the optimization problem is obtained yielding new loads and delays. The procedure continues until the aforementioned delay deviation is below some accuracy threshold and the solution regarding the determined DUs and XPUs remains unchanged. A specific application of this approach is reported in Section 2.4.1.

The developed optimization framework suffers from an exponential explosion of variables with respect to network size and the number of flows. By relating it to the multi-commodity flow problem with integer constraints (known to be NP-complete), it is shown in Appendix B.1 to be NP-complete.

2.3. Heuristic Algorithm

As a consequence of the NP-completeness, the computational complexity would be very high when large scale environments are considered. For such environments, an

efficient heuristic of low computational complexity is proposed for solving the optimization problem prescribed in Equation (2.1). The efficiency of the heuristic, which may yield the optimal or a suboptimal solution, is assessed in Section 2.4. The heuristic algorithm is described in detail in Algorithm 1 and it is outlined next.

Algorithm 1 Heuristic 1: Heuristic algorithm for flexible RAN elements

```
1: procedure HeuristicFlexibleRAN
2:     All sources ← DUs
3:     DUsNotUsed ← DUs
4:     while (UsedXPUs < MaxXPUs)&&(maxDUhit > 1)&&(DUsNotUsed > NumberSources) do
5:         maxDUhit ← 0
6:         for all r ∈ XPUPlacement do
7:             maxDUXPU ← 0
8:             for all l ∈ DUsNotUsed do
9:                 Path1_{f^l} ← ShortestPath(DU_l, XPU_r)
10:                while (Capacity(link)+f^l > MaxCapacity(link), link ∈ Path1) and (Not All Links Removed) do
11:                    Remove links that cannot transport f^l
12:                    Path1_{f^l} ← ShortestPath(DU_{f^l}, XPU_r)
13:                Path2_{f^l} ← ShortestPath(XPU_r, Destination)
14:                while (Capacity(link) + f^{l+} > MaxCapacity(link), link ∈ Path2) and (Not All Links Removed) do
15:                    Remove links that cannot transport f^l
16:                    Path2_{f^l} ← ShortestPath(XPU_r, Destination)
17:                Recompute delays for flows already routed
18:                if Recomputed delays satisfy their maximum delay then
19:                    Keep the paths and the DUs that are placed for the current XPU
20:                    maxDUXPU ← maxDUXPU + 1
21:            if maxDUXPU > maxDUXPUsaved then
22:                maxDUXPUsaved ← maxDUXPU
23:                Save the information for all the DUs that uses this XPU
24:        if maxDUXPUsaved > 1 then
25:            maxDUXPUit ← maxDUXPUsaved
26:            Save the information for all the DUs that uses this XPU
27:            Update DUsNotUsed removing the ones that uses the selected XPU
28:    flag ← 1
29:    while flag == 1 do
30:        flag ← 0
31:        for all l ∈ DUsNotUsed do
32:            for all k ∈ BackhaulFlowsOfSource(l) do
33:                Path_{b^l_k} ← ShortestPath(source_l, Destination)
34:                while (Capacity(link)+b^l_k > MaxCapacity(link), link ∈ Path) and (Not All Links Removed) do
35:                    Remove links that cannot transport b^l_k
36:                    Path_{b^l_k} ← ShortestPath(source_l, Destination)
37:                Recompute delays for flows already routed
38:                if Recomputed delays satisfy their maximum delay then
39:                    Keep the path of the new backhaul flow and update the loads in the links
40:                else
41:                    flag ← 1
42:                    Remove one DU from the XPU that accommodates more DUs
43:                    if The XPU selected contains only 2 DUs then
44:                        Remove the two DUs
45:                    Add the selected DUs to DUsNotUsed
46:                    Update all the information saved for those DUs
```

The algorithm aims at determining the best placements for the XPUs (supporting 2 or more DUs) while trying to accommodate as many DUs as possible. The algorithm starts

by trying to accommodate the largest possible number of DUs that can be supported by one only XPU and determine the (best) placement of that one XPU. To accomplish this, the algorithm starts assuming that all the sources are DUs and the algorithm computes the paths and the associated loads/delays from the sources to each candidate XPU placement. The placement determined and the supported DUs are kept as the baseline for the next round of the algorithm. In the next round, the placement of one XPU that can accommodate the largest number of the remaining DUs is determined. Following this, the new loads and delays are recalculated and the latest solution is accepted only as long as previous solutions are not invalidated; that is, the delay requirements of the flows whose paths were determined previously are not violated due to the new loads of the paths determined by the latest round. These rounds are repeated until the sources are exhausted or no more XPUs can be placed without invalidating previous placements.

At this point the XPUs and the supported DUs have been determined, including the paths from the DUs to the supporting XPU (fronthaul flow) and the path from the XPU to the destination (backhaul flow). The backhaul flows from the remaining sources (which are eNBs) are then routed to their destination; notice that the delay constraints are not as stringent for backhaul flows and that their loads are substantially less than that of the DU sources (fronthaul). Shortest path routing is considered for these backhaul flows, taking into account the remaining capacity of the links after having accommodated the flows of the DU sources. If the delay requirement of previously routed flows is not violated due to the shortest path routing of a backhaul flow, the determined path is accepted for the current flow. If the delay requirement of a previously routed flow is violated, the responsible links are "removed" (i.e., cannot be part of the route for the current flow) and the shortest path algorithm is reapplied until a path is found.

If no path is found for at least one (eNB) flow, we consider the XPU that accommodates the largest number of DUs and we switch one of those DUs to an eNB. The flow of that DU is removed, as well as the flows of all the eNBs routed before. If that XPU accommodates only two DUs, the XPU is removed and both DUs are removed since an XPU cannot support only one DU. The procedure for routing the eNB flows is then started again and is repeated until all such flows are routed.

At the end of this heuristic algorithm we obtain the largest possible number of DUs that can be accommodated, the number and placement of the supporting XPUs and the routes for all flows. This solution (regarding the number of DUs and XPUs) will be compared for some network topologies and scenarios against that returned by the optimal one obtained with a much higher computational complexity.

2.4. Validation and Application of the Approaches

The optimization framework and the heuristic approach developed in this work are validated and evaluated by applying them over some topologies of practical interest in a Matlab environment.

During the research to find the best topology for the testing of our algorithms, we found that there is no single common topology used for the transport network of operators. Based on the available fiber deployment and geographical characterization, the operator may choose one topology or another. The only common rule followed is the use of aggregation of ring topologies, in which rings with lower bandwidths are aggregated in larger rings with higher capacity. Hence, following this spirit, we have selected a ring topology that is computationally feasible to validate the heuristic results compared with the linearized problem results. Moreover, to test the algorithm proposed we have selected a synthetic topology that follows the same principles given by the operators and the works [51] and [52], and can be characterized by a set of parameters which can be modified to assemble an operator deployment. Thus, the topology used for the experiments can be perfectly an example of a real operator deployment or can be parametrized to be similar to a real one.

2.4.1. Small-Scale Topology

First, a relatively small scale environment is considered in order to derive results under the optimization framework in reasonable computational time. That is, to determine the maximum number of DUs that can be accommodated with the minimum number of necessary XPUs each of which supporting two or more DUs. The derived solution is compared against that obtained under the heuristic approach introduced in Section 2.3 to assess the potential effectiveness of the heuristic. The network topology considered consists of a ring of 7 nodes connected with 10Gbps (per direction) bi-directional links and each of these nodes is connected to 3 traffic sources via a 1Gbps access link to each one of them, as shown in Figure 2.2.

A quick back-of-the-envelop calculation easily reveals that for the capacities and topology shown in Figure 2.2, the maximum number of DUs is 21 (all of the sources can be DUs) and the minimum number of XPUs is 1 (supporting all 21 DUs). Due to the symmetry in the topology, this XPU may be placed in any of the 7 nodes of the ring. The XPU will receive 9 of the non-local fronthaul flows (of a rate of 0.9Gbps each) over the clock-wise 10Gbps ring and the other 9 non-local fronthaul flows over the counter-clock-wise ring of 10Gbps; the 3 local DUs are supported by the XPU without creating fronthaul traffic over the ring. Without loss of generality it is assumed that the 21 backhaul flows exiting the XPU facility are forwarded to destinations outside the shown network topology.

The aforementioned back-of-the-envelop result is just a way of explaining the results obtained by simulation and are the same as the ones obtained by solving the optimization problem and by applying the heuristic approach, validating both approaches. The back-of-the-envelop results are shown in Figure 2.3 and correspond to the value of $\rho = 1$ (denoting that the full capacity of the 10Gbps links is available, see discussion below). Notice that the number of XPUs is 1 and the total Air Bandwidth is equal to 4200 Mbps under both the optimization and the heuristic approaches. Assuming a cell Air Bandwidth of approximately 150 Mbps (Long Term Evolution (LTE) eNB node using 2x2 MIMO and 20MHz channel) and that a 33% Air Bandwidth gain is achieved when the eNB is replaced by a DU (which benefits from coordinated processing of its signals with those from at least one more cell [53]), then the total Air Bandwidth achieved by the 21 DUs is 21x200=4200Mbps, as shown in Figure 2.3 for $\rho = 1$.

As an iterative approach is needed for obtaining the optimization solution due to the queuing delay approximation (see discussion at the end of Section 2.2.1), the following may be reported for the solution obtained under the aforementioned experiment. The initial value for the loads are set to $(\rho_{ij}^{1,0}, \rho_{ij}^{2,0}) = (0.25, 0.25)$. Then, the loads in all links are determined. Their average values over all ρ_{ij} tuples were equal to $(\rho_{ij}^{1,1}, \rho_{ij}^{2,1}) = (0.2280, 0.0101)$ and the maximum value for $\rho_{ij}^{1,1}$ appeared in the tuple of $(\rho_{ij}^{1,1}, \rho_{ij}^{2,1}) = (0.9900, 0.0750)$; notice the lower values of load for the backhaul traffic (class 2) as this traffic imposes a lighter load and the solution determines that all the sources become DUs (generating fronthaul traffic till their CUs). With the new load values we iterate one more time and the final solution is reached and remains there after unchanged.

In order to test the performance of the developed approaches further, we expand the scenario considered in Figure 2.2 by considering that only ρ, $0.5 \leq \rho \leq 1$, of the ring capacity is available. As observed in Figure 2.3, the results obtained under both approaches coincide, demonstrating again the effectiveness of the heuristic approach. It may be noted that all 21 sources can be DUs, as the achieved Air Bandwidth remains equal to 4200Mbps, for $0.5 \leq \rho \leq 1$. This fact, indicates that the same level of DU aggregation results are obtained by the application of the heuristic approach and the optimization framework. On the other hand the number of XPUs (i.e. XPU locations) required increases from 1 to 3, as the ring capacity decreases and the resulting fronthaul traffic cannot be forwarded to a single node any more. Note that although the numbers of XPUs and Air Bandwidth are the same for both approaches, some result details such as the location of the XPUs differ in both solutions. In addition, the resulting graphs in Figure 2.3 completely overlap due to the designed topology, which allows the deployment of all base stations as DUs.

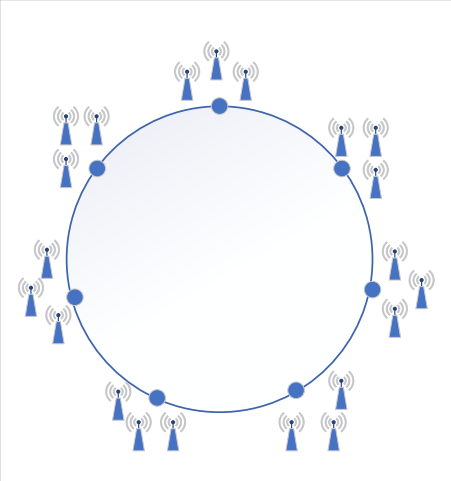


Figure 2.2: Small Scale Validation Environment.

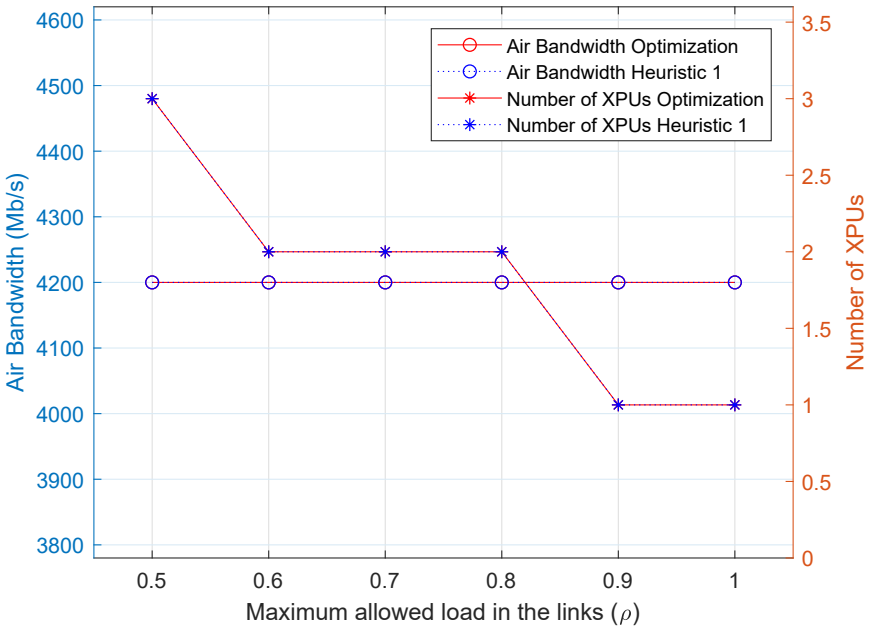


Figure 2.3: Optimization vs Heuristic 1 cite.

2.4.2. Large-Scale Topology / Practical Crosshaul Transport Network

In this subsection a large-scale network topology - that is likely to encounter in real environments - and some scenarios of potential interest to operators are considered. Due to the high computational complexity of the optimization framework, results are obtained by employing the heuristic approach of Section 2.3. These results turn out to provide for efficient deployment of C-RAN (that is, improved placement of the XPUs and accommodation of a large number of DUs). To this end, the Crosshaul transport network

depicted in Figure 2.4 is considered that represents a real production transport network deployed in the north of Italy. This network was provided by an operator involved in the 5G-Crosshaul project [10]. It is based on a number of optical rings where the base stations are connected to. Each blue point in the rings of Figure 2.4 corresponds to an Edge data center (potential host of an XPU facility). The length of each ring varies depending on the geographical area, ranging from 3Km to 100Km. This is the reference topology considered in this subsection.

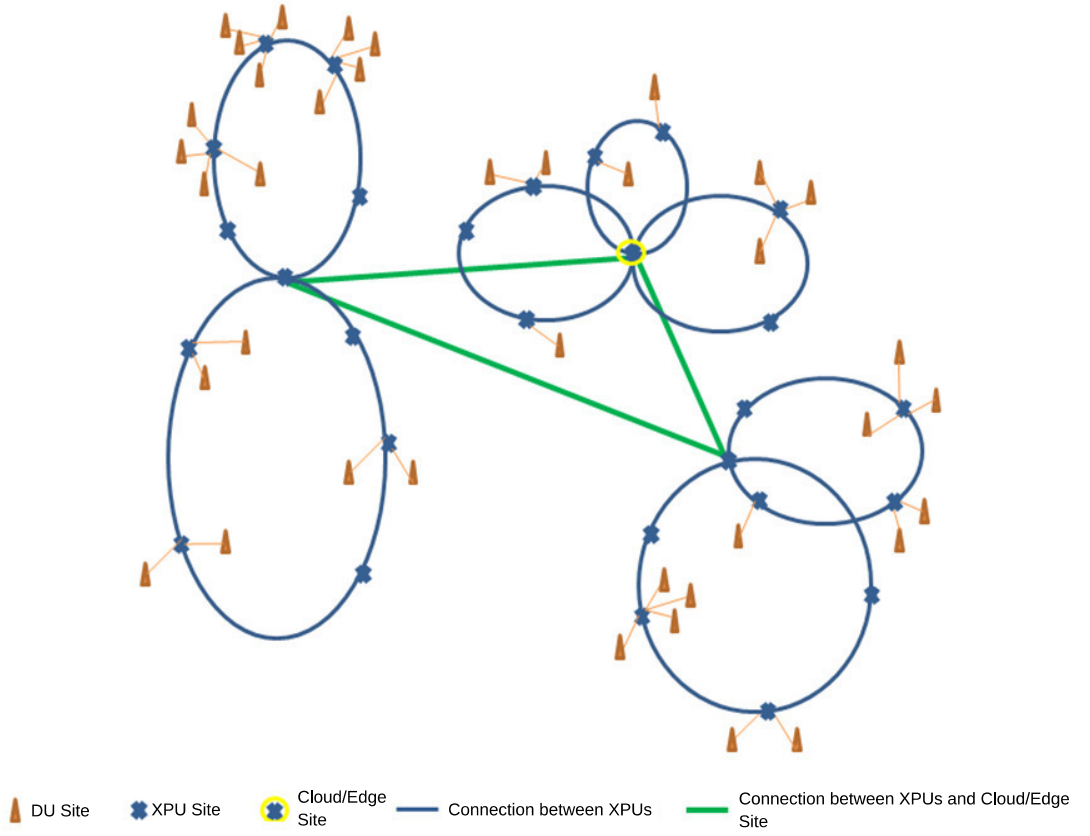


Figure 2.4: Reference topology.

Based on the scenario depicted in Figure 2.4, we have generated synthetic Ring-Tree based topologies as shown in Figure 2.5. Their configuration parameters (number of base stations, possible location for Edge data centers, radius of the links, etc.) are generated randomly. The generation process begins by forming hexagonal cells that form groups of size $A1$. Each of these hexagonal cells are supported by either a complete eNB node or by a DU.

Each of these cells is connected via a 1 Gbps link to one of the $A2$ nodes that reside on a ring of capacity of 10Gbps, which node is common to all the cells belonging to the same $A1$ group. $A3$ of those rings of $A2$ nodes are connected via a ring of capacity of

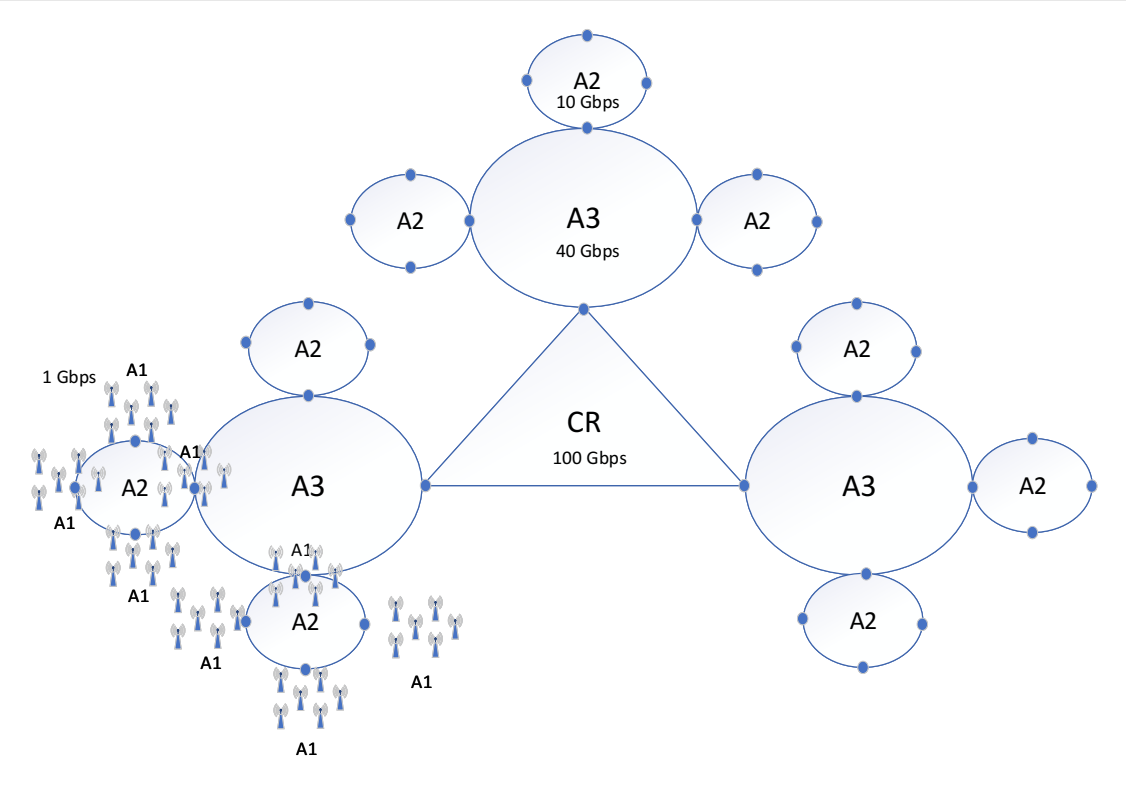


Figure 2.5: Synthetic Ring-Tree based topology.

40 Gbps. Finally, CR of those rings of $A3$ nodes are connected via a single ring CR (of capacity of 100 Gbps). Any of these CR nodes of the central ring would be considered to be the exit to the Internet where the destination of any flow generated within this topology would reside. Finally, any of the nodes residing in any of the rings is a potential host of an XPU. For the rest of the section we will assume a topology with 339 nodes in total ($CR = 3$, $A3 = 5$, $A2 = 4$, $A1 = 6$). The objective of this section is to evaluate how a large scale operator network can be optimized based on our approach. In order to do so, we will stress the network based on 2 scaling parameters: i) the maximum end to end propagation delay in the network and ii) the maximum allowed capacity used in the links of the network, ρ, as we also did in the small scale environment case. With the first of these parameters we control the diameter of the network. This way, we can possibly have all XPUs placed in the central ring if the propagation delay is below the stringent delay constraint and the available capacities permit it. With the second parameter we control the available capacities in the network, which would also affect the placement of XPUs, depending on the induced queuing delays.

In the first of the experiments of this section, we derive and present results by applying Heuristic 1 (see Section 2.3) to the large scale network described above. Heuristic 1 determines the deployment mix of eNBs and DUs (and their placement) aiming at

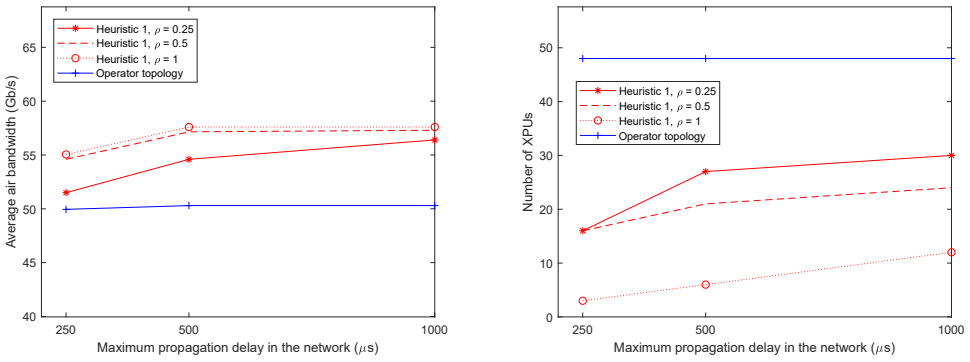
maximizing the Air Bandwidth (or number of DUs deployed), while minimizing the number of deployed XPUs. Figure 2.6 presents the results of Heuristic 1 for the Air Bandwidth (Figure 2.6(a)) and the Number of XPUs required (Figure 2.6(b)). Considering the Number of XPUs, Figure 2.6(b) shows how the number of XPUs deployed increases with the maximum propagation delay. The main reason for this behavior is that due to the delay increase, the aggregation of the flows of a high number of DUs in the higher aggregation rings (A3 and CR, in Figure 2.5) is not possible, requiring more XPUs and distributing them over the lower aggregation rings (A2 in Figure 2.5) to meet the fronthaul delay constraints. To illustrate this, consider the curve for $\rho = 1$ and compare the result corresponding to a propagation delay of $250\mu s$ with that of $1ms$. For the case of $1ms$, Heuristic 1 results in 12 XPUs: 3 XPUs placed in rings A3 and 9 in A2. For the case of $250\mu s$, Heuristic 1 places a total of 3 XPUs, placed 1 in the central ring and 2 in the A3 rings. As explained earlier, the reason for this difference is that when delay constraints are met, the best solution is to aggregate in the higher aggregation rings.

Following a similar line of reasoning, when the maximum capacity of the links is reduced, from $\rho = 1$ to $\rho = 0.25$, the fronthaul traffic cannot be pushed deeper into the network, due to the saturation of the links in the aggregation rings. For this reason, the number of XPUs required increases while ρ decreases. To illustrate this, consider the result under a propagation delay of $500\mu s$ for $\rho = 0.25$ and $\rho = 1$. For the case of $\rho = 1$, the total number of XPUs is 6, placing 1 in a A3 ring and 5 of them in the A2 rings. For the case of $\rho = 0.25$, Heuristic 1 results in 27 XPUs, placing 1 in the central ring, 5 in A3 rings and 21 in A2 rings. As explained, the lower the bandwidth available (lower ρ), the more the XPUs required and the less the aggregation.

Figure 2.6(a) shows the resulting total Air Bandwidth of all DUs and eNBs, whose numbers are determined by Heuristic 1. As expected, Heuristic 1 achieves a higher level of aggregation under lower maximum propagation delay, determining a lower number of XPUs placed deeper in the network. For instance, under $500\mu s$ and $1ms$ maximum propagation delays, the number of XPUs increases (and they are pushed towards the edge of the network), compared with that under $250\mu s$. As a result, the number of DUs deployed will be decreased under low maximum propagation delay, since the fronthaul flows will share the bandwidth with backhaul flows for longer paths deeper into the network and the total Air Bandwidth will decrease, for all values of ρ. In addition, as expected, as ρ decreases, the resulting Air Bandwidth decreases accordingly. Note that this seems a different behavior from the one in Figure 2.3 where the air bandwidth does not decrease, it remains the same (4200 Mbps) for every value of ρ. It is not a different behavior, but in Figure 2.3 for the values of ρ selected the transport capacity still allows that all the sources are DUs, but due to the lack of transport bandwidth (lower ρ means lower link capacity) when ρ decreases the number of XPUs required to maintain the same air bandwidth has to increase. Here, in Figure 2.6(a) the lack of bandwidth in the links

affects to the number of sources that can be DUs also and this is the reason it modifies the Air Bandwidth.

Finally, in Figure 2.6, we also provide a base-line to compare to. The line corresponding to the Operator topology represents the results that will be obtained by an operator deploying the same networks as used for Heuristic 1, but considering that the operator deploys just 1 XPU in the first point aggregating the DUs (point corresponding to A1 in Figure 2.5), resulting in a higher number of XPUs (see Figure 2.6(a) and Figure 2.6(b)) compared to the case under Heuristic 1. In addition, since the operator does not try to aggregate DUs, the pooling gain and Air Bandwidth gains based of cooperative signal processing cannot be obtained and the total Air Bandwidth is lower.



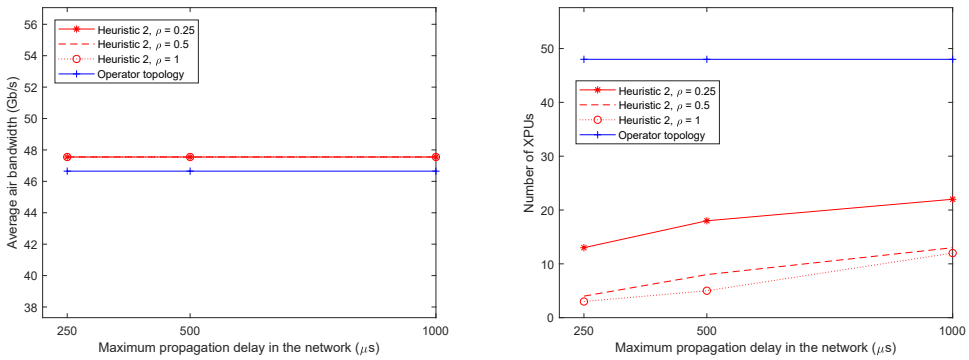
(a) Air Bandwidth

(b) Number of XPUs required

Figure 2.6: Cite of Heuristic 1 and a generic Operator deployment.

The proposed Heuristic 1 tries to optimize the network both in terms of Air Bandwidth (by choosing if a RAN element must be deployed as an eNB or a DU) and reduced number of XPUs. This is possible only if the RAN has not been already deployed or if the deployed RAN elements can be flexibly configured as eNBs or DUs. Since this is not always possible, as part of this work we have also developed a modification of Heuristic 1, called Heuristic 2 (see Appendix C.1), which takes as input a given topology with fixed RAN elements (i.e., whether they are eNB or DUs and their positions) and computes the minimum number of required XPUs. Results for Heuristic 2 are presented in Figure 2.7. As in Figure 2.6, we derive and present the Number of XPUs deployed in Figure 2.7(b) and the achievable total Air Bandwidth in Figure 2.7(a), for different values of the maximum propagation delay and different values of ρ. In order to build the simulated topologies we use the same ones as in Figure 2.6, but considering a probability of choosing eNB or DU $p_{DU} = 0,5$, resulting in an average of 144 DUs. The results in Figure 2.7(b) show that the Number of XPUs can be significantly reduced by applying the solution obtained by Heuristic 2, compared with the generic Operator deployment. Notice also a similar trend and for the same reasons as for Heuristic 1: the number of required XPUs increases with

the maximum propagation delay and ρ. Regarding the Air Bandwidth, since all RAN elements are fixed, the bandwidth obtained by Heuristic 2 is similar to the Operator deployment, with a small gain due to the higher aggregation of DUs achieved. This small gain is already obtained with the lower value of ρ, thus the only value that changes when we increase the ρ is the number of XPUs required.



(a) Air bandwidth

(b) Number of XPUs required

Figure 2.7: Cite of Heuristic 2 and a generic Operator deployment.

2.5. Conclusion

This chapter has focused on the development of a framework for the joint optimization of an integrated networking and edge/cloud environment supporting two diverse classes of flows (fronthaul/backhaul) under path and delay constraints. This framework is directly applicable to the optimal design or dynamic management of a mixed RAN and C-RAN environments, foreseen on the road to 5G networking. These mixed environments emerge as operators attempt to maximize their adoption of the C-RAN technology in the most effective way, subject to the constraints imposed by the available supporting infrastructure. Or, such environments may emerge in a more dynamic (operation-level) case, where operators may switch on/off DUs, such as Remote Radio Heads (RRH), or aggregate them in a lower number of CUs, such as Base Band Units, according to the demand to reduce OPEX, necessitating the re-optimization of the resulting mixed RAN / C-RAN environment. The 5G networks incorporating a mixed RAN and C-RAN environment (where some nodes are split while others are not), will face planning and deployment challenges, requiring mechanisms to decide on the most appropriate RAN element to split and the placement of the supporting CUs in the edge/cloud. It is also important to highlight that the use of split RAN elements requires the transport of the generated fronthaul flows characterized by more stringent throughput and delay requirements (than the RAN-generated backhaul flows) all the way to their CUs.

This chapter provides an optimization framework and computationally more efficient heuristics to tackle exactly the aforementioned problems. The main contributions of this work are: i) an optimization framework for joint routing and resource placement is developed, taking into account delay, capacity and path constraints, maximizing the degree of DU deployment while minimizing the supporting CUs, ii) an efficient heuristic approach for solving the optimization problem in large scale environments, allowing the operator to derive solutions aiming at maximizing the Air Bandwidth (that is boosted by properly splitting a RAN element) while minimizing the number of XPUs (edge/cloud nodes hosting an array of CUs) by determining the placement of XPUs and the RAN elements that can be split into DUs and iii) a heuristic allowing the operator to compute the minimum number of XPUs and their placement for a given mixed RAN/C-RAN deployment. The approaches have been applied to both small scale and large scale/production level environments, demonstrating the effectiveness of the heuristics and the optimization approach and yielding potentially large gains in terms of reduced number of required Edge data-centers and increased Air Bandwidth.

3 Virtual Network Function (VNF) Placement on Mobile Cloud/Edge Environments

5th generation of telecommunications networks (5G) (& beyond) systems have been promising several appealing sometimes unbelievable future use cases and applications which could reshape our society. The vast number of Internet of Things (IoT) devices, autonomous vehicles and different types of robots collaborating with each other and with humans are expected to be part of our lives. These devices usually require coordination or fine granular, dynamically programmable control from a reliable and permanently available platform. Coordination of collaborating robots, drone swarms, self-driving cars or any types of unmanned vehicles are good examples with several fields of application from industry to agriculture and from logistics to emergency management. The envisioned use cases typically pose serious challenges on the underlying networks and cloud platforms in terms of latency and reliability. For example, 3rd Generation Partnership Project (3GPP) specified a dedicated set of features for mission critical applications referred to as Ultra-Reliable Low-Latency Communication (URLLC) [54].

Edge and fog computing, Multi-access Edge Computing (MEC) are key enablers of these applications. The main concept is to extend traditional cloud computing by deploying compute resources closer to customers and end devices. By these means, both end devices and central cloud servers can offload computational tasks to resources at the edge or the fog resulting in lower delays and in reduced network load. In order to meet the strict delay and reliability requirements of mission critical applications, a distributed and heterogeneous infrastructure and the encompassed compute and network resources should be managed carefully. The underlying infrastructure includes both public and private cloud/edge resources [55] providing execution environments for VNF interconnected by public 5G networks and privately operated domains. In this environment, resource orchestration is a challenging task which aims at always finding the proper placement of software components realizing the service. Moreover, robots or different vehicles equipped with sensors, actuators and local computation environments, provide capabilities which

can or must be consumed by certain applications. More exactly, now we can run VNFs on these continuously moving mobile devices, and the uninterrupted communication to other service components should also be guaranteed. Beside mobility, the limited battery capacity and the VNFs power consumption are novel aspects to be considered in the placement decision.

Due to the widespread of virtualization technologies, the problem of allocating VNFs on top of physical resources has been of interest in recent years. In most of the existing research the allocation of VNFs is envisioned as an optimization problem, that is generally $\mathcal{NP}$-hard [56].

A common technique is to solve the VNF allocation problem as a variation of the bin packing problem, taking the VNFs as items, and the bins as servers. Particularly, the first steps of this chapter's proposed heuristic are build upon the basis defined in the algorithm of [57], which minimizes a data center energy consumption using a generalized bin packing problem. Works as [58] solve the VNF allocation using the variable size bin packing problem [59], which provides an efficient solution to minimize both response time, and resource utilization. Other research projects have studied different and relevant generalizations for variable sized bin-dependent costs [60].

A recent survey categorizes bin packing problem generalizations which might be relevant to VNF placement solutions [61]. In general, algorithms for bin packing problems do not consider delays on the sequence of items, nor any topological constraint among the bins, so using their results for VNF placement problems is not trivial; our heuristic builds on such results.

Solutions of the VNF allocation problem must reshape with the new 5G networks, which bring computational capabilities closer to the user thanks to MEC [62], and fog computing [63]. Indeed, servers are way closer to antennas, or even co-located with them in the edge, and IoT devices are becoming part of a dense network. Thus, 5G comes with the urge of a more dense radio coverage, and the possibility of sharing public/private network infrastructure [55] [64] can help to achieve it. Orchestration in the edge of 5G has motivated solutions [65] that benefit from edge servers to asses the mapping and migration of VNF resources upon users' mobility. Additionally, edge computing has popped up the quest of deploying Network Service (NS) with very strict latency requirements, and recent research as [66], [67], [68], [69], and [70] study solutions about how to allocate VNFs to meet low latency requirements. [66] uses a genetic algorithm to obtain a fixed allocation that minimizes/maximizes latency/availability, [70] provides a stopping theory solution that migrates the allocated VNFs as time passes, such that latency restrictions are not violated. [67] formulates an optimization problem to allocate VNFs demanded by end-users attached to antennas, so as to maximize/minimize resources re/usage, by imposing latency constraints. [68] proposes a deep learning agent that assigns VNFs to servers maximizing the requests' throughput, while they meet latency constraints. [69]

presents a solution that maximizes the throughput of services in 5G slices, while meeting latency requirements of each slice. The solution idea relies on preventing the performance interference caused by co-locating multiple VNFs in the same server.

There is some research that focus on VNF allocation in fog environments. [71] presents an allocation model accounting for the computational overhead of fog devices, based on the assigned workload; and [72] studies how to satisfy End-to-End (E2E) delay by reducing the distance of the deployed service, to the users consuming the service (envisioned as traffic generators). Another approach to allocate VNFs is to deploy them jointly using cloud and fog devices, as [73] does. In that work, service providers derive a wireless and resource sharing model of fog devices, and the allocation is done using a student project allocation algorithm. There are other results [74] related to low energy IoT devices, that study the trade-off between the energy requirement for computation, and transmitting data, as a computation task outsourcing pipeline is proposed.

Although the literature already provides solutions to perform the VNF allocation on edge and fog scenarios, the work exposed in this chapter contributes to the state-of-the-art by tackling all at once the (i) radio coverage; (ii) battery consumption; and (iii) E2E delay restrictions present in 5G use cases with mobile compute nodes.

The rest of the chapter is organized as follows. Section 3.1 introduces a future use case motivating the work. Sections 3.2 and 3.3 are devoted to the detailed description of the model and the optimization problem. Section 3.4 describes the proposed heuristic algorithm. Section 3.5 presents the algorithms' evaluation from different aspects based on extensive simulations.

3.1. Mobile Robotics Use Case

This work tackles the mobile robotics use case [75, Table 5.3.1.1-1], as a warehousing solution for future factories [76, Section 3.1.2]. In particular, it deals with the transport of goods from boats to specific locations of Valencia city haven.

The use case considers a cluster of robots, that move in a *master-slave* fashion to deliver goods arriving to the haven. Each of the robots carries containers from a pick up point (S in Figure 3.1) to a drop off point (D1 and D2). In particular, the *master* robot is followed by the other *slave* robots (represented in Figure 3.1) of the cluster along its way towards the drop off point. Robots communicate among themselves to report position status, or other context information useful for the *master-slave* coordination. Thus, robots have device-to-device communication between them, and computational capabilities so they can execute lightweight VNFs [77] as the *driving* and *follow* VNFs represented in Figure 3.1. The *driving VNF* runs in the *master* robot to drive it towards the drop off point, and the *follow VNFs* run in the *slave* robots to follow the master robot movements until it reaches a drop off point. The *driving VNF* receives driving instructions

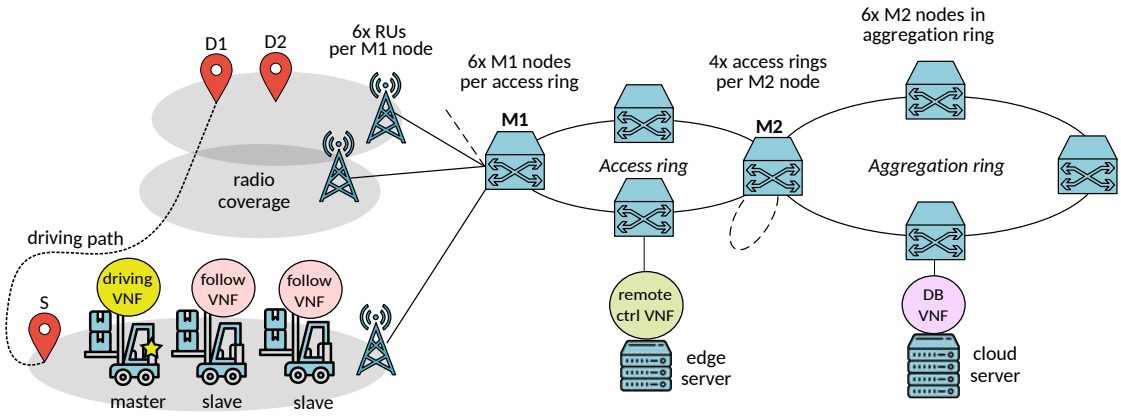


Figure 3.1: Deployment of a cloud robotics warehousing NS.

from the *remote ctrl VNF* running in the edge server in Figure 3.1, and reports sensor data like the speed to the *Database (DB) VNF* running in the cloud.

To enhance the robots' remote driving, the communication between the *remote ctrl VNF* and the *driving VNF* is crucial, indeed, resources proximity is needed as Mobile robotics demand communications with cycle times between 1ms and 100ms (for machine control, and video operated remote control cases) [75]. Thus, the placement of both *driving* and *remote ctrl VNFs* should satisfy latencies below 100ms.

While moving, robots may run out of battery or switch between Radio Unit (RU) coverage area (see Figure 3.1). Whenever the *master* robot enters a new coverage area, it attaches to a new RU to keep the connectivity with the servers running the *remote driving VNF*, and *DB VNF* (edge, and cloud servers in Figure 3.1). Therefore, it is important to take into account that a robot is not selected for goods delivery if (i) it may run out of battery; or (ii) it may lose RU connectivity as it moves towards the drop off point.

To increase the RUs coverage and improve the E2E delay, the use case presented in this section considers that the haven is covered by Long Term Evolution (LTE) RUs managed by a network operator, and New Radio (NR) RUs belonging to its Non Public Network (NPN), which is called an NPN deployment in a public network [55]. That is, Valencia city haven only owns the NR RUs, and its management (subscription, gateways, control plane) is done by the public network, i.e., a network operator.

For the public network infrastructure, a 5G transport network is assumed based on [78] and [79]. All the RUs present in the use case transmit their traffic up to an access ring composed of several switches connected in a ring fashion. The traffic of the access rings is latter gathered by the aggregation rings which forward traffic up to the core of the public infrastructure. The presented use case, assumes that cloud servers are in the core of the public network, edge servers are co-located next to the access ring and the aggregation ring switches. Regarding computational resources (i.e., Central Processing Unit (CPU), memory and disk), edge servers in access rings are less powerful than edge servers in

aggregation rings, and cloud servers are more powerful than edge servers.

It is worth highlighting that the problem formulation presented in this work holds for public and private deployments, being the only consideration the cost of connection, that may vary depending on the type of management. The mention here is for a better understanding on the real situation in the mentioned city haven.

3.2. System Model

This section presents the architecture, elements and dependencies considered for the problem formulation in Section 3.3. The network infrastructure is represented by a graph G_I, where the nodes $V(G_I)$ contain NR and LTE RUs, generally referred to as Access Point (AP) $V_{AP}(G_I)$, server nodes (representing edge or cloud servers) $V_S(G_I)$, and mobile nodes $V_M(G_I)$. Hence, the vertex set of the graph is built up as $V(G_I) = V_{AP}(G_I) \cup V_S(G_I) \cup V_M(G_I)$. Host nodes N_i with computation capacities $\overline{C}_{N_i}$ are stored in $V_H(G_I) = V_S(G_I) \cup V_M(G_I)$, and their corresponding unitary price is represented by p_{N_i}. As a realistic generalization to the mobile robotics use case, the concurrent management of multiple robot clusters is assumed. The subsets of $V_M(G_I)$ define the clusters of robots $V_{RC_q}(G_I) \subseteq V_M(G_I), 1 \leq q \leq Q$, where Q refers to the number of clusters. Moreover, graph edges $E(G_I)$ represent the connections between the infrastructure nodes, which are annotated by their transmission delays.

Due to the mobile clusters' mobility, their connections to the static part of the infrastructure are not represented by edges in G_I.

The mobile nodes $V_M(G_I)$ are connected to access points $V_{AP}(G_I)$ in order to communicate with other nodes of the infrastructure. However, the nodes are moving and may encounter areas with overlapping access point coverage or areas where handover between different access points is needed to guarantee the connection to the servers deeper in the infrastructure. Thus, this work assumes that each AP has an associated coverage area AP_k, and the mobility pattern of robot cluster q is modeled by the probability distribution of being in the AP coverage areas $\mathbb{P}_{AP_k^q}(t)$, referred as *coverage probability* throughout the chapter. Notice that a cluster can be in an area where several access points have coverage, with a different probability for each of them. Each value models the probability of a robot cluster q to fall inside the coverage area of each AP in each moment t. This model is able to compute the placement of NSs with guarantees of communication between the mobile and fixed parts of the infrastructure, while considering any model of coverage areas, such as [80] or a linear model, by using precomputed values of the coverage.

The parameter t is a time instant within an interval (t_0, t_1) in which the network service will be running. For the sake of simplicity in the model, the time interval is discretized in subintervals, thus continuous time $t \in (t_0, t_1)$ becomes discrete time $t_u \in \{t_0, t_a, t_b \ldots, t_1\}$

with $t_0 \leq t_a \leq t_b \leq \ldots \leq t_1$. Subintervals help to identify the moments when handovers may occur during the service time. The time division guarantees the communication between robots and APs selected in each subinterval. Note that VNFs are deployed on the same servers during all the service time, thus, they must have communication with the APs selected in each subinterval.

The cost of using an AP for a single subinterval t_u by any single cluster is p_{AP_k}. The energy consumption of the mobile nodes is modeled by the distribution $\mathbb{P}_{bat}(N_i, C_{N_i})$ depending on the allocated load to node N_i, which represents the probability of having a not depleted battery for the whole interval (t_0, t_1).

Both $\mathbb{P}_{AP_k^q}(t)$ and $\mathbb{P}_{bat}(N_i, C_{N_i})$, are used in the optimization problem to ensure robots' radio coverage, and battery needs are met during the interval (t_0, t_1).

The requested NSs are represented with a NS graph G_S, with the nodes being VNFs $v \in V(G_S)$ and their capacity requirements C_v. Each Service Function Chain (SFC) is a subgraph $G_s \subseteq G_S$ with its own set of VNFs and path, as the one depicted in the NS graph of Figure 3.1, and expressed in Equation (3.1).

$$\mathcal{C}_{SFC} = \{(G_s, \Delta_{G_s}) \mid V(G_s) \subseteq V(G_S),\ E(G_s) \in \mathcal{P}(G_S), \Delta_{G_s} \in \mathbb{R}^+\} \tag{3.1}$$

where $\mathcal{C}_{SFC}$ represents the set of SFCs in NS G_S, and $\mathcal{P}(G_S)$ represents the paths of the NS graph G_S. Each SFC has a corresponding delay requirement Δ_{G_s} which defines an upper bound of the total delay of the SFC path $E(G_s)$.

For a better understanding of the model, all the notations used for the mathematical formulation of the optimization problem are gathered in Tables 3.1 and 3.2.

3.3. Problem Formulation

This section presents the formulation of the use case to tackle the VNF allocation as an optimization problem. The problem is solved using an integer program solver to gain optimality and scalability insights. The optimization must decide which infrastructure node $N_i \in V(G_I)$ should host which VNF $v \in V(G_S)$, which is represented by the binary decision variable $x(v, N_i)$ and constraints Equation (3.2) and Equation (3.3).

$$x(v, N_i) \in \{0, 1\} \qquad \forall v \in V(G_S), \forall N_i \in V(G_I) \tag{3.2}$$

$$\sum_{N_i \in V(G_I)} x(v, N_i) = 1 \quad \forall v \in V(G_S) \tag{3.3}$$

The resource capacities $\overline{C}_{N_i}$ must be respected by the load allocation on each node N_i. This requirement is gathered in Equation (3.4), where C_{N_i} stands for the allocated

Parameters		Definition
G_I		Network infrastructure graph
G_I	$V(G_I)$	All infrastructure nodes
	$V_*(G_I)$	Nodes of type $*$ in the infrastructure $* \in \{AP,\ S,\ M,\ H,\ RC_q\}$
	$E(G_I)$	Edges of the infrastructure
G_S		NS graph
G_S	$V(G_S)$	All VNFs of the network service
	$\mathcal{P}(G_S)$	All paths of the service graph
	G_s	Graph of SFC G_s
	$V(G_s)$	VNFs of SFC G_s
	$E(G_s)$	Edges of SFC path G_s
	Δ_{G_s}	Delay requirement of SFC G_s
	th^s_{bat}	Battery threshold for SFC G_s
	$\mathcal{C}_{SFC}$	Set of all SFCs
VNF v		VNF v
VNF v	C_v	Capacity demand of VNF v
	L	Locality matrix $V(G_S) \times V(G_I)$
N_i	$\overline{C}_{N_i}$	Total resource capacity of node N_i
	p_{N_i}	Cost per resource unit used of node N_i
	$D_{AP,S}(N_i, N_j)$	Delay between N_i and $N_j \in V_{AP}(G_I) \cup V_S(G_I)$
	$D_{M_q}(N_i, N_j)$	Delay between N_i and $N_j \in V_{RC_q}(G_I)$
	$\mathbb{P}_{bat}(N_i, C_{N_i})$	Probability of having battery for the whole time interval using C_{N_i} resources
AP_k	d_{AP_k}	Delay for the coverage area of AP_k
	$\mathbb{P}_{AP^q_k}(t_u)$	Probability of cluster q to be in the coverage area of AP_k in time subinterval t_u
	p_{AP_k}	Cost of usage of AP_k
	κ_q	Coverage probability threshold for cluster q

Table 3.1: Parameters of the optimization formulation.

Variables	Definition
$d_{G_s}(t_u)$	Delay of SFC G_s in time t_u
$d(N_i, N_j, t_u)$	Delay between nodes N_i and N_j in time t_u
$x(v, N_i)$	Placement of VNF v in node N_i
C_{N_i}	Resource usage in node N_i
$AP^q_k(t_u)$	Usage of AP_k by cluster q time t_u
$\mu : V(G_S) \mapsto V_H(G_I)$	VNF to host node mapping structure
$\alpha : \{t_u\} \times \{q\} \mapsto V_{AP}(G_I)$	AP selection structure for all clusters

Table 3.2: Variables of the optimization formulation.

resources in infrastructure node N_i as presented in Equation (3.5).

$$C_{N_i} \leq \overline{C}_{N_i}, \quad \forall N_i \in V_H(G_I) \tag{3.4}$$

$$C_{N_i} = \sum_{v \in V(G_S)} x(v, N_i) C_v, \quad \forall N_i \in V_H(G_I) \tag{3.5}$$

Furthermore, there may be a necessity of applying placement policies and VNF functional types. In order to include those policies in the model, the matrix $L(v, N_i)$ expresses locality constraints between the VNFs $v \in V(G_S)$ and infrastructure node

$N_i \in V_H(G_I)$. Each element of the matrix is a binary constant, identifying whether the VNF can be located in an infrastructure node, as expressed in Equation (3.6).

$$x(v, N_i) \leq L(v, N_i), \quad \forall v \in V(G_S), \forall N_i \in V_H(G_I) \tag{3.6}$$

In the use case presented in Section 3.1, $L(v, N_i)$ enforces the deployment of the *driving* and *follow VNFs* in the robots (i.e., mobile nodes). This requirement may be useful for other use cases, such as Unmanned Aerial Vehicles (UAVs) running virtual access points that forward traffic to the cloud (see [77,81]). Under such scenarios, $L(v, N_i)$ can be used to enforce virtual access points to run on top of UAVs.

3.3.1. Radio Coverage Constraints

The deployment must also decide at each time interval to which access point each cluster of robots is attached to, that is, $AP_k^q(t_u) = 1$ in case robot cluster RC_q is connected to access point AP_k at time t_u. Equation (3.7) reflects the assumption that each cluster can only be attached to one AP at each interval.

$$\sum_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u) = 1, \ \forall 1 \leq q \leq Q, \forall t_u \in (t_0, t_1) \tag{3.7}$$

The deployment decision must also ensure that the coverage probability is above the imposed threshold κ_q for mobile cluster q, representing the requirements each cluster needs to guarantee connectivity during the time interval, as stated in Equation (3.8).

$$\sum_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u) \cdot \mathbb{P}_{AP_k^q}(t_u) \geq \kappa_q, \quad \substack{\forall 1 \leq q \leq Q \\ t_u \in (t_0, t_1)} \tag{3.8}$$

Notice that this optimization problem only needs to know whether cluster q has radio coverage of AP_k at time t_u, which makes it agnostic about how $\mathbb{P}_{AP_k^q}(t_u)$ is obtained, and the values could be derived from any radio access model. For instance, Section 3.5 obtains $\mathbb{P}_{AP_k^q}(t_u)$ with a linear function directly proportional to the distance between q and AP_k.

3.3.2. Delay Constraints

In order to measure the distances between infrastructure nodes, the metric used is the delay, which in the case of the static nodes is given in a matrix containing the precomputed and the time-independent delays, $D_{AP,S}(N_i, N_i)\ \forall N_i, N_j \in V_S(G_I) \cup V_{AP}(G_I)$.

Similarly, the distances inside each mobile cluster are time invariant, precalculated and stored in matrix $D_{M_q}(M_i, M_j)\ \forall M_i, M_j \in V_{RC_q}(G_I), 1 \leq q \leq Q$.

Each access point $AP_k \in V_{AP}(G_I)$ provides a time- and distance-independent delay to its whole coverage area, its value is denoted by d_{AP_k}, while delay between APs is

given with the value $D_{AP,S}(AP_{k_1}, AP_{k_2})$. The delay value between a mobile cluster and the static part of the infrastructure and between mobile nodes belonging to two different clusters might vary according to the assigned APs during the time interval (t_0, t_1). A mobile cluster q has an appointed relay node $N^{(r_q)}$ (in our case the *master* robot), which is connected to the APs, and all the traffic of other mobile nodes of the same cluster towards the fixed part of the infrastructure goes through the corresponding relay mobile node. Thus, the orchestration system can execute the handover of the cluster by only connecting the relay node to a different AP. This way the delay of device-to-device communication is accounted in a different variable than the AP delays.

Hence, the general delay function which covers any pair of infrastructure node types is expressed in Equation (3.9).

$$d(N_i, N_j, t_u) = \begin{cases} \sum\limits_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u)[D_{M_q}(N_i, N^{(r_q)}) + d_{AP_k} + D_{AP,S}(AP_k, N_j)], & \text{if } N_i \in V_{RC_q}(G_I) \wedge N_j \in V_S(G_I); \\ d(N_j, N_i, t_u), & \text{if } N_j \in V_{RC_q}(G_I) \wedge N_i \in V_S(G_I); \\ D_{AP,S}(N_i, N_j), & \text{if } N_i, N_j \in V_S(G_I) \cup V_{AP}(G_I); \\ D_{M_q}(N_i, N_j), & \text{if } N_i, N_j \in V_{RC_q}(G_I); \\ \sum\limits_{\substack{AP_{k_1}, AP_{k_2} \\ \in V_{AP}(G_I)}} AP_{k_1}^{q_i}(t_u) AP_{k_2}^{q_j}(t_u) \left(D_{AP,S}(AP_{k_1}, AP_{k_2}) + d_{AP_{k_1}} + d_{AP_{k_2}} + \sum\limits_{n \in \{i,j\}} D_{M_{q_n}}(N_n, N^{(r_{q_n})}) \right), & \text{if } N_i \in V_{RC_{q_i}}(G_I) \wedge \ N_j \in V_{RC_{q_j}}(G_I) \end{cases} \tag{3.9}$$

Equation (3.9) is a piece-wise function that depends on the type of node hosting the different VNFs of the NS. The delay between two mobile nodes of the same cluster is accounted in $D_{M_q}(N_i, N_j)$. Delay between two nodes of the fixed infrastructure is $D_{AP,S}(N_i, N_j)$, while delay between a node of the fixed infrastructure and a mobile cluster depends on the AP that the cluster uses in that moment, which is gathered in $\sum\limits_{AP_k \in V_{AP}(G_I)} AP_k^q(t_u)[D_{M_q}(N_i, N^{(r_q)}) + d_{AP_k} + D_{AP,S}(AP_k, N_j)]$. Thus, the delay of a service is composed by the different delays between the nodes that host the different VNFs and the order in which they must be performed.

The overall delay of a SFC $G_s \in \mathcal{C}_{SFC}$ in time t_u is formulated in Equation (3.10), where the delays between the hosts of each SFC edge are summed.

$$d_{G_s}(t_u) = \sum_{\substack{(v_i, v_j) \in E(G_s) \\ N_i, N_j \in V(G_I)}} x(v_i, N_i) x(v_j, N_j) d(N_i, N_j, t_u) \tag{3.10}$$

The upper bound of the SFCs' total permitted delay Δ_{G_s} for the whole optimization interval is expressed in constraint Equation (3.11).

$$d_{G_s}(t_u) \leq \Delta_{G_s}, \ \forall (G_s, \Delta_{G_s}) \in \mathcal{C}_{SFC}, \forall t_u \in (t_0, t_1) \tag{3.11}$$

3.3.3. Battery Constraints

In order to place VNFs in mobile nodes it is necessary to ensure the mobile node will not run out of battery during the time interval (t_0, t_1). This is introduced in the problem formulation, in Equation (3.12), as the probability of having battery for the whole time interval considered, based on the resources used in the node.

$$\mathbb{P}_{bat}(N_i, C_{N_i}) = \mathbb{P}_{bat}(N_i, 0) - \frac{C_{N_i}}{\overline{C}_{N_i}} \left(\mathbb{P}_{bat}(N_i, 0) - \mathbb{P}_{bat}(N_i, \overline{C}_{N_i}) \right), \forall N_i \in V_M(G_I) \quad (3.12)$$

C_{N_i} is the consumed capacity of mobile node N_i, and $\mathbb{P}_{bat}(N_i, C_{N_i})$ is the probability of having battery on N_i by the end of time interval (t_0, t_1) when using C_{N_i} resources as allocated capacity. Note that the optimization problem is agnostic of the used battery consumption model, as $\mathbb{P}_{bat}(N_i, C_{N_i})$ values could be derived by any battery consumption model. For example, Section 3.5 derives $\mathbb{P}_{bat}(N_i, C_{N_i})$ as a linear function between the empty $C_{N_i} = 0$ and the fully loaded states $C_{N_i} = \overline{C}_{N_i}$. To ensure the proper performance of the mobile nodes, the battery life is guaranteed in Equation (3.13) by a threshold th^s_{bat} given per SFC G_s, for all nodes hosting VNFs.

$$\mathbb{P}_{bat}(N_i, C_{N_i}) \geq th^s_{bat} x(v, Ni),\ \forall N_i \in V_M(G_I),\ \forall v \in V(G_s),\ \forall G_s \in \mathcal{C}_{SFC} \quad (3.13)$$

This threshold takes into account the battery of all the mobile nodes hosting the VNFs of the service and guarantees each of the nodes hosting a VNF will have battery during the whole time interval with a probability higher than the threshold, for example a $th^s_{bat} = 0.9$.

3.3.4. Cost Minimization

Finally, the problem minimizes the total cost of allocating all the services demanded, and gathered in G_S, as well as the cost of the APs used by all of the mobile clusters along the service duration. Hence, the objective function is shown in Equation (3.14).

$$\min \sum_{N_i \in V(G_I)} C_{N_i} \cdot p_{N_i} + \sum_{t_u, q, k} AP^q_k(t_u) \cdot p_{AP_k} \quad (3.14)$$

The VNF mapping μ and AP selection structures α are defined by the variables $x(v, N_i)$ and $AP^q_k(t_u)$ of a solution to the optimization problem. This model is not linear in some equations as the one representing the delay in Equation (3.10), but each product of two variables can be easily linearized due to the fact that all the variables involved are binary variables. Thus, the linearization is performed by substituting each product of two binary variables by one extra binary variable, as expressed in Property 5 in Appendix A.1.

3.4. Heuristic Algorithm

This section details the design of the heuristic which exploits the peculiarities of the system model to design an efficient and practical algorithm. The core idea of the heuristic algorithm proposed to solve the optimization problem is to use the fractional optimal solution of a bin packing problem of the VNFs and host nodes, which is deterministically rounded to an invalid integer solution. Next, the algorithm iteratively resolves the capacity, delay, battery and coverage constraint violations by changing the mapping location of VNFs in the initial invalid integer solution until a feasible mapping is found.

First, Formulation 1 introduces the bin packing problem variation with variable bin and item sizes supporting linear usage costs [57].

Formulation 1 Bin Packing with Usage Cost [57]
Input: VNFs $V(G_S)$ as items with weight, host nodes $V_H(G_I)$ as bins with capacity
Output: VNF placement respecting only capacity constraints

$$\sum_{N_i \in V_H(G_I)} x(v, N_i) = 1 \qquad \forall v \in V(G_S) \tag{3.15}$$

$$\sum_{v \in V(G_S)} x(v, N_i) C_v \leq \overline{C}_{N_i} \qquad \forall N_i \in V_H(G_I) \tag{3.16}$$

$$x(v, N_i) \in \{0, 1\} \qquad \forall v \in V(G_S), N_i \in V_H(G_I) \tag{3.17}$$

$$min \sum_{N_i \in V_H(G_I)} C_{N_i} \cdot p_{N_i} \tag{3.18}$$

Lemma 1 states how to construct a fractional optimal solution for this bin packing variant, relaxing the integrality constraint. The proof of Lemma 1 can be found in the original source [57].

Lemma 1 (Fractional optimal solution of Formulation 1 [57])**.** *Let $\{a_i\}$ be a permutation of all host infrastructure nodes $N_i \in V_H(G_I)$ in ascending order by their unit costs of computation capacity $p_{a_1} \leq p_{a_2} \leq \cdots \leq p_{a_{|V_H(G_I)|}}$. Let $W_C = \sum_{v \in V(G_S)} C_v$ be the sum of all VNF capacities. Let b be the minimum number of host nodes in order $\{a_i\}$ where $\sum_{i=1}^{b} \overline{C}_{N_{a_i}} \geq W_C$. The fractional optimal solution (discarding the integrality constraint (3.17)) of Formulation 1 is*

$$\tilde{x}(v, N_{a_i}) = \begin{cases} \frac{\overline{C}_{N_{a_i}}}{W_C} & if \quad i < b, \\ \frac{W_C - \sum_{i=1}^{b-1} \overline{C}_{N_{a_i}}}{W_C} & if \quad i = b, \\ 0 & if \quad i > b; \end{cases} \qquad \forall v \in V(G_S).$$

The pseudo-code of the proposed heuristic for this problem is shown in Algorithm 2.

Algorithm 2
Input: service graph G_S, infrastructure G_I, improvement score limit Υ, and all constraints from Section 3.3
Output: VNF placement $\mu : V(G_S) \mapsto V_H(G_I)$ and AP selection $\alpha : \{t_u\} \times \{1 \dots Q\} \mapsto V_{AP}(G_I)$ satisfying all constraints

1: **procedure** PLACEVNFSSELECTAPS(G_S, G_I, Υ)
2: $\tilde{x}(v, N_i), b, \{a_i\} \leftarrow$ fractional solution based on Lemma 1 for host nodes $V_H(G_I)$ and VNFs $V(G_S)$
3: **for** $v \in V(G_S)$ **do** ▷ Round initial solution
4: $\mu(v) \leftarrow argmax_{N_i \in V_H(G_I)} \tilde{x}(v, N_i)$ which obeys locality constraints (3.6)
5: **for** $b' \in \{b \dots |V_H(G_I)|\}$ **do** ▷ In order of $\{a_i\}$
6: $\mathcal{V}, \mathcal{R} \leftarrow$ VIOLATINGVNFMAPPINS(μ)
7: **while** $\mathcal{V} \neq \emptyset$ **do**
8: $\mathcal{I} \leftarrow \emptyset$ ▷ Allowed improving VNF moves
9: **for** $v \in \mathcal{V}$, $i \in \{1 \dots b'\}$ **do**
10: **if** $\mu(v) \neq N_{a_i}$ **and** $\mu(v) = N_{a_i}$ obeys
11: locality constraints (3.6) **then**
12: **if** $\Upsilon \leq$ IMPROVESCORE(μ, v, N_{a_i}) **then**
13: $impr\_cost \leftarrow C_v(p_{N_{a_i}} - p_{\mu(v)})$
14: $\mathcal{I} \leftarrow \mathcal{I} \cup \{(v, N_{a_i}, impr\_cost)\}$
15: **if** $\mathcal{I} \neq \emptyset$ **then**
16: $\mu(v) \leftarrow N_{a_i} \mid (v, N_{a_i}, impr\_cost) \in \mathcal{I}$ **and** $impr\_cost$ is minimal
17: $\mathcal{V}, \mathcal{R} \leftarrow$ VIOLATINGVNFMAPPINS(μ)
18: $\alpha \leftarrow$ retrieve AP selection from violation record $\mathcal{R}$
19: **else break**
20: **if** AP selection α is valid **and**
21: VNF placement μ is valid **then**
22: **return** μ, α ▷ Solution found
23: **return** $\emptyset, \emptyset$ ▷ Solution not found

Intuitively, the heuristic reallocates VNFs that violate any constraint, and measures the goodness of the reallocation with the *improvement score* (see Algorithm 4). The higher the *improvement score*, the better the VNF reallocation. Initially, the fractional optimal solution is retrieved and rounded to initial constraint-violating VNF placement, obeying only the locality constraints (3.6) as shown in lines 2-4. The cost increasing order $\{a_i\}$ of mobile and server nodes are used from Lemma 1 to involve additional hosts to the VNF placement pool, starting only from the first b cheapest hosts. In each iteration a set of violating items, respecting all constraints is calculated based on the temporary decisions stored in the current VNF placement function μ. Next, the iteration in lines 9-14 collects improvement scores for moving a VNF which is involved in any constraint

violation to any currently considered host node (i.e. until index b'). Line 12 heuristically filters only the VNF relocations whose improvement score is higher than a configured improvement score limit Υ. The improvement cost is calculated by the cost difference of VNF v on the current host $\mu(v)$ and the possible new host N_{a_i}. If any allowed VNF replacement is found, update actions are taken and a current AP selection α is retrieved as shown in lines 16-18. Otherwise, the algorithm exits the improvement operations, and the next cheapest mobile or server node is included in the search by increasing b'. If a feasible solution is found after any inner iteration (see Line 22), the procedure returns the current VNF placement function μ and AP selection structure α. The presented algorithm could be easily extended to continue searching for better quality solutions at the price of increased running time.

All subsequently presented subroutines take the input of Algorithm 2, but these are omitted from the pseudo-codes for readability. VIOLATINGVNFMAPPINGS takes as input the current VNF placement function μ and returns a set of violating VNFs $\mathcal{V}$ and an information storage of the actual constraint violations $\mathcal{R}$. Based on the current VNF placement μ, the feasibility of AP selection for each robot cluster $q \in \{1 \dots Q\}$ is checked using the subroutine CHOOSEAPS. If the AP selection is not possible, all VNFs of the causing SFC G_s are added to $\mathcal{V}$ and the violation information is stored in constraint violation record $\mathcal{R}$.

Algorithm 3

Input: Current VNF placemnet μ, current (possibly incomplete or invalid) AP selection α, robot cluster index q

Output: Extended and/or ivalidated AP selection α, AP selection violation record $\mathcal{R}^{AP}$

1: **procedure** CHOOSEAPS(μ, α, q)
2: **for** $t_u \in (t_0, t_1)$ **do**
3: **if** $\exists G_s \in \mathcal{C}_{SFC}, \exists v \in V(G_s)$,
4: where $\mu(v) \in V_{RC_q}(G_I)$ **then**
5: $d^{G_s}, AP_l \leftarrow$ DELAYDISTWITHCOVERAGEANDAPSELECTION($E(G_s), \mu, t_u, q, \kappa_q$)
6: **if** $d^{G_s} \leq \Delta_{G_s}$ **and** $\exists AP_l$ **then**
7: Let $\alpha(t_u, q) = AP_l$ ▷ Same AP for all SFCs
8: **else**
9: Let $\alpha(t_u, q) =$ ⌈
10: Add result d^{G_s} and SFC G_s to $\mathcal{R}^{AP}$
11: **else**
12: Let $\alpha(t_u, q) = AP_l$, where $AP_l \in V_{AP}(G_I)$ and obeys coverage constraint Equation (3.8) and p_{AP_l} is minimal
13: **return** α, $\mathcal{R}^{AP}$

Algorithm 3 shows the details of how the AP selection and its feasibility based on the placement function μ are derived for a given robot cluster $q \in \{1 \dots Q\}$ for all temporal subintervals. Line 4 chooses the affected SFCs G_s, which have any VNF mapped to the mobile nodes of the robot cluster q. Given the current VNF placement μ, the total delay used by the path of the whole SFC $E(G_s)$ can be calculated using the delay expression (3.9). Access points are chosen by discarding the ones which do not meet the coverage requirement κ_q and finding the one with minimal delay among the remaining ones:

$$AP_l = \text{argmin}_{AP_k \in V_{AP}(G_I) \cap \{AP_\phi : \mathbb{P}_{AP_\phi^q}(t_u) \geq \kappa_q\}}(d_{AP_k}) \tag{3.19}$$

These operations are done by the function DELAYDISTWITHCOVERAGEANDAPSELECTION, which also ensures that the same AP is chosen for a given input robot cluster $q \in \{1 \dots Q\}$ in subinterval t_u, no matter which input SFC it gets. The algorithm discards the impractical option of placing the VNFs of a single SFC to distinct mobile clusters. This simplification is only applied for the delay bounded VNFs, not to the other VNFs of the network service G_S. If an access point AP_l is found for subinterval t_u with the given requirements, the selection is saved in AP selection function α, otherwise the structure is invalidated and the reason is saved in $\mathcal{R}^{AP}$, as shown by the logical structure starting at Line 6. In case the computation capacities of a robot cluster are not used by any VNFs of any SFC, an access point still needs to be selected for the cluster, which is done by minimizing the cost instead of the unbounded delay and similarly filtering to the coverage probability (see Line 12).

Finally, the improvement score calculation is shown in Algorithm 4, which takes the current VNF placement μ and a possible relocation of VNF v to N_{a_i} as input, and outputs an integer whose higher value represents a more significant improvement. The IMPROVESCORE procedure uses the previously presented VIOLATINGVNFMAPPINGS function to evaluate how the mapping would change by the VNF mapping modification. The mapping structure μ with less violating constraints is considered better, as shown in lines modifying the improvement score y. In case of capacity constraints, total improvement score y would decrease, keep unchanged or increase if the number of hosts with more than their max capacity allocated would increase, stay or decrease by the VNF movement, respectively (see Line 5). A similar score modification is done for each SFC, using the change in the number of temporal subintervals t_u where the coverage or delay constraints are violated as shown by the iteration starting at Line 6. In the case of the battery constraints, the number of VNFs mapped to mobile nodes with violated battery thresholds are used.

Algorithm 4
Input: Current VNF placement μ, movement of VNF v to host N_{a_i}
Output: Integer in interval $[-|\mathcal{C}_{SFC}|-2, |\mathcal{C}_{SFC}|+2]$, the improvement score of the VNF movement

1: **procedure** IMPROVESCORE(μ, v, N_{a_i})
2: $y \leftarrow 0$ ▷ Init. improvement score of moving v to N_{a_i}
3: $\mathcal{V}, \mathcal{R} \leftarrow$VIOLATINGVNFMAPPINGS($\mu$)
4: $\mathcal{V}', \mathcal{R}' \leftarrow$VIOLATINGVNFMAPPINGS($\mu \mid \mu(v) = N_{a_i}$)
5: $y \leftarrow y - 1/ + 0/ + 1$ if *number of hosts* N_i with violated constraint (3.4) increases/stays/decreases in $\mathcal{R}'$ compared to $\mathcal{R}$
6: **for** $G_s \in \mathcal{C}_{SFC}$ **do**
7: $y \leftarrow y - 1/ + 0/ + 1$ if *number of subintervals* t_u with any invalid mappings (i.e. where $\exists t_u, q : \alpha(t_u, q) = \lceil\ \rceil$ increases/stays/decreases in $\mathcal{R}'$ compared to $\mathcal{R}$
8: $y \leftarrow y - 1/ + 0/ + 1$ if *number of VNFs* v which are mapped to any mobile node $V_M(G_I)$ with violated battery constraint (3.13) increases/stays/decreases in $\mathcal{V}'$ compared to $\mathcal{V}$
9: **return** y

3.4.1. Complexity Analysis

A brief analysis on the heuristic's complexity and its termination is presented in Theorem 1 and its corresponding proof.

Theorem 1 (Complexity of heuristic)**.** *The overall complexity of the heuristic with positive improvement score limit* $\Upsilon > 0$ *is:*

$$\mathcal{O}\Big(|V(G_S)|^4 |V(G_I)|^3 |\mathcal{C}_{\mathcal{SFC}}| QT\Big) \tag{3.20}$$

where Q *and* T *are the number of clusters and the number of subintervals* t_u *in the optimization time frame* (t_0, t_1)*, respectively.*

Proof: Looking at Algorithm 2, the fractional solution construction and its rounding are dominated by the iteration starting at Line 5, which is executed at most $|V(G_I)|$ times. Assuming a positive improvement score limit Υ, the violating VNFs set $\mathcal{V} = \mathcal{O}(|V(G_S)|)$ decreases at least by one element in each iteration of the *while* cycle. At most every iteration runs VIOLATINGVNFMAPPINGS. Filtering for the allowed VNF movements in Line 11 is done at most $\mathcal{O}(|V(G_S)||V(G_I)|)$ times, and in worst case for each of them we execute a IMPROVESCORE subroutine call. These observations make Algorithm 2's complexity to be $\mathcal{O}\Big(|V(G_S)||V(G_I)|\{|V(G_S)||V(G_I)|$IMPROVSCORE

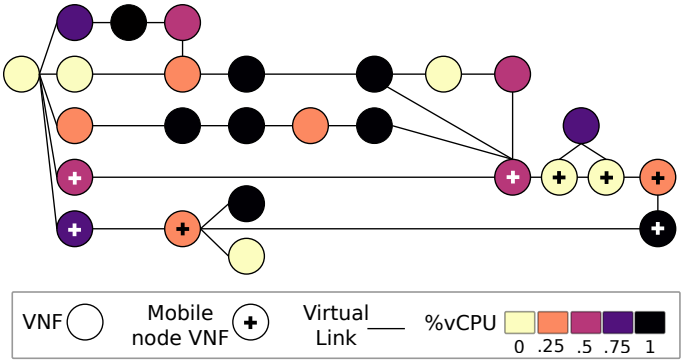


Figure 3.2: A service graph generated with a series-parallel graph. This instance contains x8 VNFs bounded to mobile nodes, and is used in the battery experiment (Figure 3.4).

+ violatingVNFMappings$\}\big)$. The violatingVNFMappings's complexity is dominated by $Q\mathcal{O}$(chooseAPs), because the other constraints can be checked in $\mathcal{O}(|V(G_I)||V(G_S)|)$ time. Access point filtering for sufficient coverage in a longest SFC can be done in $\mathcal{O}(|V(G_I)||V(G_S)|)$ time, which is done for all SFCs $|\mathcal{C}_{SFC}|$, all SFC edges $\mathcal{O}(|V(G_S)|)$ for all time subintervals T. Which gives $\mathcal{O}$(violatingVNFMappings) $= \mathcal{O}(QT|V(G_S)^2||V(G_I)||\mathcal{C}_{SFC}|)$. Similarly, improvScore is dominated by violatingVNFMappings's complexity. Finally, a Floyd-Warshall algorithm is used to pre-calculate the all the delay matrices $D_{AP,S}$ and D_{M_q} with complexity $\mathcal{O}(|V(G_I)|^3)$, which is dominated by the previous operations. Substituting and ordering the $\mathcal{O}(\cdot)$ notations, the statement follows. ■

3.5. Validation and Application of the Approaches

This section compares the performance of Section 3.4 heuristic, with the optimal solution of Section 3.3 formulation from various aspects. As integer programs are generally impractical due to the hardness of the problem, our heuristic is extensively evaluated to demonstrate its applicability. The heuristic solutions are compared to the optimal solution obtained with Gurobi which finds a solution within a gap optimality of 3%. Such comparison is done for the mobile robotics use case of Section 3.1, where scalability is a critical issue due to the size of the infrastructure and service graphs.

Additionally, this section compares Section 3.4 heuristic against "Follow Me Chain" (FMC) [65], a heuristic that tackles mobility by triggering VNF migrations upon AP handovers, but does not consider battery constraints. Our implementation of FMC (i) replaces [65, Algorithm 1] VNF-based Breadth-First Search (BFS) with a virtual-link-based BFS, so as to ensure the mapping of every virtual link; (ii) uses a k-shortest paths in [65, Algorithm 2:line 1] to avoid getting stucked in the search of all paths between two

nodes[1]; (iii) considers mobile compute nodes as well as edge servers; and (iv) can map service graphs with unconnected components.

3.5.1. Experiment Setup

The presented evaluation scenario scales up the mobile robotics use case of Valencia city haven. A realistic 5G network infrastructure topology is considered with multiple types of wireless access points, while the service graph instances are random graphs. Many parameters of the experiment setting are examined during the presented simulations, varying the size of the input, SFC delay requirements, coverage probabilities and battery thresholds.

In order to generalize the service graphs and gain confidence in our simulations, series-parallel graphs are used to generate the network service topology G_S. Figure 3.2 shows an example of such graph. This graph class covers the structure of many data streaming applications, such as map-reduce topologies, and have been used in other realistic, industrial case studies for fog application allocation [82]. The round-trip time experienced by every robot running a VNF must stay below the delay restriction, therefore, SFCs correspond to loops starting and ending in mobile node VNFs, and each SFC must satisfy the delay restriction. Among all the VNFs of the SFC, some of them are forced to run on top of the mobile robots' hardware $V_M(G_I)$ (denoted as *Mobile node VNF* in Figure 3.2), and the rest can run on top of any server $V_S(G_I)$ or robot $V_M(G_I)$. It is up to the heuristic and the optimization formulation, to decide where to deploy them.

Every experiment uses the 5G infrastructure characterization of [79] and [78], which considers URLLC. Table 3.3 shows every infrastructure element considered in the experiments, and Figure 3.1 illustrates the interconnection of the network infrastructure. Each M1 switch is located in the access ring of the network, and it gathers the traffic of up to x6 LTE or NR RUs. Access rings have x6 M1 switches and x1 M2 switch, all of them interconnected in a ring fashion. Every M2 switch belongs to x4 access rings, and it steers the traffic up to the aggregation ring, where it is connected in a ring fashion with another x5 M2 switches. Experiments consider that edge and cloud servers are reachable using M1 and M2 switches, respectively.

Each point in the operation area of the robot cluster is covered at least by one LTE RU and at least by one NR RU. The coverage probabilities for each time instance are derived by a function which maps the distance of the RU and the cluster to the coverage probability. The probability slightly decreases until the end of the RU coverage area, and steeply drops to 0 at 120% of the RU reach. If a NR RU and the mobile cluster are not in LoS, the coverage probability is 0, independent of their distance. To achieve E2E delays demanded by the mobile robotics use case (between 1ms and 100ms), the

[1]FMC builds a full-mesh servers' graph, and even the proposed range-based Depth-Fist Search (DFS) incurs into a $\mathcal{O}(|V(G_i)|!)$ search space

#	Element	Characteristics
x2	LTE RU [83]	8km radio coverage, 5ms one way delay [84], 5.5 cost units OPEX [85]
x36	NR RU [86]	700m LoS coverage [87], 1ms one way delay, 11 cost units OPEX
x10	robots	x2 CPUs, 15.27 cost units/CPU [88]
x6	edge server	x12 CPUs, 5.83 cost units/CPU [88]
x2	cloud rack	200 CPUs, 2.46 cost units/CPU [88]
x8	M1 switch	x8 dedicated CPUs
x6	M2 switch	x238 dedicated CPUs
x2	access rings	fiber ring connection, $\leq$6 M1 switches
x1	aggregation ring	fiber ring connection, x6 M2 switches

Table 3.3: Infrastructure used for experimentation.

experiment infrastructure assumes that aggregation and access ring switches introduce packet processing delays between 1ms and 10ms, under the same characterisation as performed in [79].

To derive this section's results, a network infrastructure with just one cluster of robots has been generated with the 5GEN R package [89]. Then, a Python script generates series-parallel NS graphs G_S from which loop SFCs are selected. Robot cluster paths are encoded by coordinates which are used to calculate RUs coverage probabilities as robots move along the path. Next, the Python script runs Section 3.4 heuristic to decide each VNF mapping on top of the infrastructure graph. Section 3.3 formulation is encoded in AMPL [90], and the Python script invokes Gurobi 8.1 solver [91] through the *amplpy* API to obtain the optimal mapping. All the experiments have been executed on two identical Virtual Machine (VM) with x4 virtual Central Processing Unit (vCPU), 32GB of memory, and 132GB of disk.

3.5.2. Simulation Results

This section presents the results of the extensive simulations performed with Algorithm 4, AMPL solver, and the state of the art FMC solution; denoted as *impr-*Υ, *AMPL*, and *FMC*; respectively. The details of the simulation parameters are shown in Table 3.4 for each of the experiments. The cluster paths in the Valencia haven are represented by their source and target locations.

All evaluation figures present boxplots, where the middle line shows the median (a.k.a. second quartile) of the dataset, while the body of the boxplots show the first and third quartiles (a.k.a. the medians of the first half and the second half of the dataset separated by its median). The whiskers of the boxplots represent the datum which deviates from the boxplot body at most by 1.5 times the inter-quartile range, while outliers are individually plotted by circles which fall beyond the whiskers.

An input VNF placement problem with all previously presented constraints is deemed feasible, if the AMPL implementation finds a valid solution that respects all constraints in

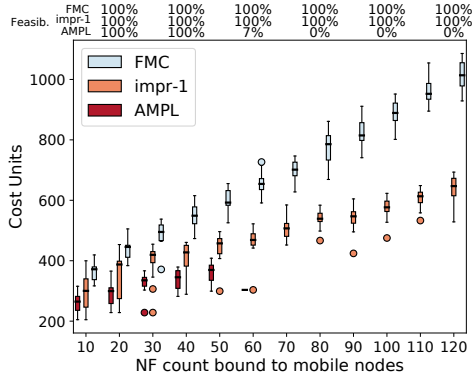
Parameter name/explanation	Value/range			
Experiment name	Scalability	Coverage	Delay	Battery
Robot cluster path, see Figure 3.1	S → D1	S → D2	S → D2	S → D1
Path total distance [meters]	678	488	488	678
Time interval count $t_u \in (t_0, t_1)$	24	24	24	24
Unloaded battery probability $\mathbb{P}_{bat}(N_i, 0), \forall N_i \in V_M(G_I)$	99%	99%	99%	99%
Full loaded battery probability $\mathbb{P}_{bat}(N_i, \overline{C}_{N_i}), \forall N_i \in V_M(G_I)$	50%	80%	80%	50%
Battery probability threshold th^s_{bat}	40%	70%	70%	72% & 75%
Infrastructure delay sample count	1	4	4	1
SFC delay [ms] Δ_{G_s}	1000	5	Varies	1000
Randomized VNF vCPU requirement $C_v \forall v \in V(G_S)$	0.5 x {0, ... 4}	0.5 x {0, ... 4}	0.5 x {0, ... 4}	0.25 x {0, ... 4}
hline VNF count $\lvert V(G_S)\rvert$	Varies	10	10	26
VNF count bound to robots	6	4	1	Varies
Coverage probability threshold κ_q	94%	Varies	94%	70%
Scenario repetition with different randomization seed	14	24	20	14

Table 3.4: Experiment parameters.

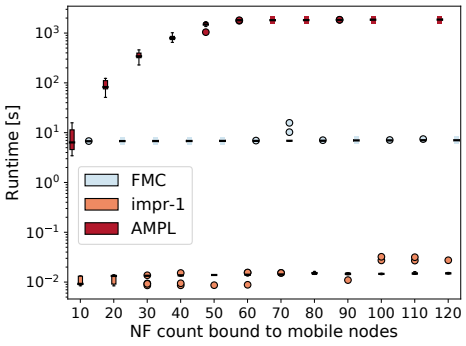
30 minutes (measured in wall-clock time). In case of the heuristic, the timeout is reduced to 20 minutes. All experiments were executed with 3% optimality gap for AMPL, various improvement score limit values for the heuristic, and $k = 10$ for FMC.

First of all, the scalability of the algorithms are compared depending on the number of VNFs to be placed; results in terms of cost and runtime are shown in Figure 3.3(a) and Figure 3.3(b), respectively. The time-bound feasibility is shown on top of the figures for each randomized scenario repetition corresponding to the dependant value on the horizontal axis. The scalability experiment is repeated multiple times for each input size, varying the distribution of VNF capacity requirements, the service graph's concrete topology, and the selection of the VNFs bound to the mobile cluster (see Table 3.4). The scenario parameters allow a solution to be found in any randomized generation, due to loose SFC delay, coverage and battery probability thresholds; though the $30mins$ time limit may not be enough in all cases. Figure 3.3(a) shows the time-bound feasibility ratio calculated on the randomized repetitions. A steep drop of feasibility of the AMPL implementation occurs at the VNF count of 60, which is due to reaching the computation timeout in each case. The reason behind the timeouts is the exponential runtime of the AMPL solution, which is shown by Figure 3.3(b) in logarithmic time scale. On the contrary, both heuristics find feasible solutions in every possible setup. In terms of cost, our heuristic with $\Upsilon = 1$ outperforms FMC, with the former staying between 15% and 30% away of the optimal costs, and the latter increasing the cost gap with respect to our solution as the number of Network Function (NF) bound to mobile nodes grows. Furthermore, our heuristic always find solutions below $100ms$ for all tests, whilst FMC takes around $10s$.

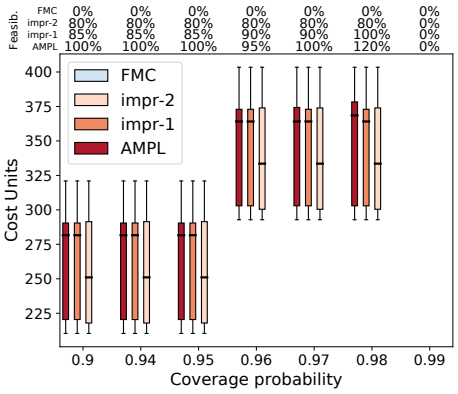
Second, the effect of the coverage probability threshold κ_q is studied. Figure 3.3(c) shows how the cost varies by increasing the threshold, i.e. making the AP selection more strict. As the coverage probability requirement increases, deployment costs become more expensive, because the solutions impose the selection of the closer and more expensive NR antennas, rather than the cheap LTE antennas. Figure 3.3(c) depicts as well the feasibility,



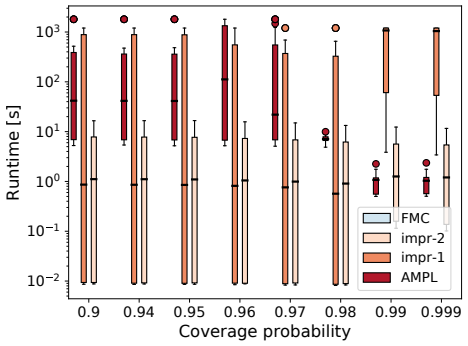
(a) Scalability test: costs



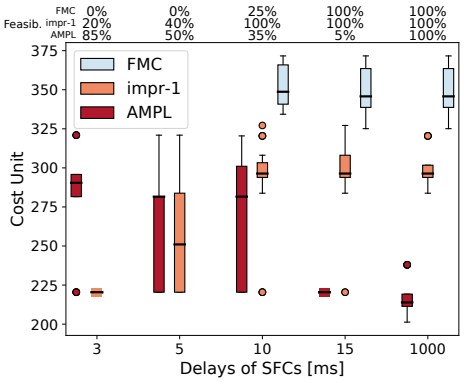
(b) Scalability test: runtimes



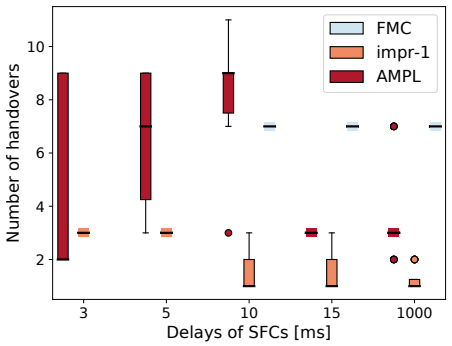
(c) Coverage threshold variation test: costs



(d) Coverage threshold variation test: runtimes



(e) SFC delay variation test: costs



(f) SFC delay variation test: handover counts

Figure 3.3: Results of scalability, coverage probability and SFC delay experiments.

and shows that for $\kappa_q = 0.99$ all scenarios are infeasible, because there exists at least one subinterval in which the cluster is not covered by any antennas with such high probability. Regarding the impact of the improvement score Υ, Figure 3.3(c) and Figure 3.3(d) show that $\Upsilon = 2$ (*impr-2* time series) finds cheaper solutions faster. This is due to the heuristic design, which goes faster by shrinking the solution space and considering only VNF relocations with higher improvement score. The heuristic finds cheaper deployments faster, because they require less steps to make the rounded fractional solution feasible. Additionally, Figure 3.3(c) shows that FMC cannot find feasible solutions with the studied coverage thresholds $\kappa_q \geq 0.9$, since one or more migrations failed during the experienced handovers.

Next, the results of simulations varying the SFC delay are shown in Figure 3.3(e) and Figure 3.3(f). FMC cannot find feasible solutions for $3ms$ scenarios, as it is designed to try to map one VNF per compute node, and therefore, its mappings have to traverse more network links. The heuristic *impr-1* struggles with finding feasible solutions in the allocated time for the $3ms$ scenarios, while AMPL manages to prove the existence of valid solutions as shown by the feasibility percentages of Figure 3.3(e). This could be easily addressed by introducing a search space pruning step in addition to the locality constraints. In the $3ms$ scenarios the usage of the cheap and high capacity cloud nodes is not an option because their RTTs from all APs are above this value. Excluding these compute nodes from the allocation options for the VNFs contained in the strict SFCs would dramatically decrease the running time and thus increase the time-bound feasibility of the heuristic. Although, additional pruning steps decrease solution quality in the cases of more permissive delay requirements. On the other hand, the heuristic greatly outperforms the optimal solution search in the 10-15ms scenarios, where the AMPL algorithm fails to find any feasible solution before the 30 minutes timeout. This is due to the growth of the search space as the delay restriction is relaxed. Additionally, *impr-1* finds cheaper deployments than FMC, since the latter tries to use one compute node per VNF, and does not account for cloud nodes by design. Note that cloud nodes are cheap and strong candidates used by *impr-1* when the delay requirement relaxes (see the SFC delay case of $1000ms$).

Another interesting aspect of the solutions is the number of required handovers needed for the whole optimization time interval. A lower handover count requires less management operations and results in a more stable service. Handover comparison between the cost-optimal and the heuristic solutions are shown in Figure 3.3(f). The heuristic outperforms the optimal solution, which is especially relevant when the scenario could be solved by a few handovers as shown by the $10ms$ experiment scenarios with 100% heuristic feasibility. The AMPL algorithm could be modified to minimize the number of handovers, but it would further worsen its scalability, while the heuristic performs well by design. Furthermore, *impr-1* required less handovers than FMC in all the simulated

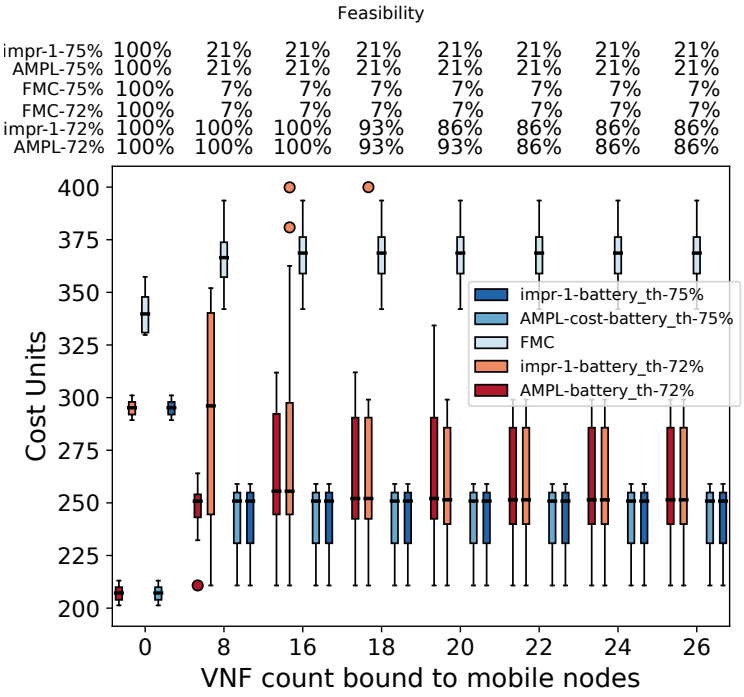


Figure 3.4: Impact of battery probability threshold on cost and feasibility.

scenarios.

Last, the results of the conducted experiments to examine the battery threshold parameter's effects are shown in Figure 3.4. The figure depicts cost values for both algorithms in cases of 72% and 75% battery alive probability requirements, as the number of VNFs to be placed on the mobile cluster increases. Note that FMC is agnostic of battery constraints and it reports the same solution, no matter the imposed battery alive probability. However, the feasibility of the FMC solution is depicted for battery alive cases. These scenarios challenge constraint (3.13), discovering the critical battery threshold to be around 72%-75%. In the 72% case the scenarios are vastly feasible with a slight decrease as the VNF bound to mobile nodes increase. The heuristic finds close to optimal allocations in almost all scenarios, except in the extreme case of much freedom. In the more strict case of 75%, besides the no location-bound VNF experiment which is essentially the same as the 72% case, the heuristic always finds all optimal solutions where it exists. Last of all, Figure 3.4 shows that FMC only finds solutions in the 7% of the simulations with more than 8 VNFs bound to mobile nodes. Indeed, it reports same feasibility ratios for both $72\%, 75\%$ battery thresholds, as it could only find deployments with $\mathbb{P}_{bat}(N_i, C_{N_i}) \geq 0.75$. As in previous results, FMC reports higher deployment costs because it tries to map each VNF to a different compute node.

3.6. Conclusion

This chapter has analyzed the notoriously hard problem of VNF placement in a realistic use case based scenario: mobile robotics for warehousing solution in the Valencia city haven, where an NPN deployment in a public network is assumed. In this scenario, mobile compute nodes act as an extension of the cloud and edge computing infrastructure, which triggers the need for VNF placement solutions with strict delay bounds and reliability constraints, while taking into account radio coverage, mobility and battery conditions.

The chapter has introduced a system model and a mathematical formulation of the problem, to then propose an efficient heuristic building on the fractional optimal solution of a bin packing variant. The heuristic has been extensively evaluated via simulations in terms of scalability and the strictness of constraints which are relevant to the use case. Results show that the proposed heuristic outperforms a state of the art mobility-aware algorithm, and achieves close to optimal deployments' in terms of cost, while improving the convergence speed to the solution (therefore the number of time-feasible solutions is increased) and minimizing the number of required handovers.

4 Network Optimization for Distributed Machine Learning

Owing to the ever-increasing scale and complexity of the learning tasks to perform, Machine Learning (ML) algorithms have swiftly been extended to work in a distributed fashion, with the purpose of leveraging the computational capability of multiple nodes, possibly across multiple datacenters [92–95] and/or allowing nodes belonging to different parties to cooperate in a learning task without sharing sensitive data [96–98].

More recently, distributed ML has emerged also as an excellent match for new generation (5G-and-beyond) networks. It can be used for the management of the network (as envisioned by such initiatives as ETSI ZSM [99], ENI [100], and O-RAN [101]), as well as to enable user services within the so-called *intelligent edge* [102]. In general, new generation networks can (i) integrate a wide number of *heterogeneous* nodes, including those that can provide the data used for ML tasks, (ii) provide a distributed computational infrastructure needed to run the ML algorithms (see e.g., [103]), and (iii) be dynamically reconfigured so as to perform the ML task at hand with the required performance.

However, implementing a ML task in a 5G-and-beyond network also poses important challenges such as defining the *logical topology* of the nodes that cooperate towards the ML task. Specifically, it requires to decide on: (i) which computing nodes in the different locations of the network edge should interact during the learning process and; (ii) how many (and which) data sources to exploit, and which computing nodes should receive their data.

The above decisions influence each other, often in counterintuitive ways: as an example, seeking information from too many nodes may result in longer learning times, due to the additional waiting. Furthermore, a given target learning error (e.g., classification accuracy) may be reached through alternative, completely different approaches, e.g., collecting a significant quantity of information *or* performing more iterations to process a smaller set of data.

In spite of the wide usage of ML in mobile networks and the considerable attention devoted to it, most of the works aim at exploiting the network more efficiently, e.g., reducing the overhead [92,104] or dealing with straggling nodes [105]. Just a small number

of recent works [96, 106] have characterized the impact of the network topology on the performance of distributed ML, providing interesting insights on, e.g., the optimal network connectivity. However, *none* of such works tackles the problem of defining the logical network topology *around* the ML task to perform.

Furthermore, the nature of the task to solve can be very diverse and approach considered to solve it will be very different, affecting the needs of the system to gather and process the data. Thus, the distributed learning approach has been studied from simple to complex cases. In the simplest cases [107], all training data is known before the training itself starts, and the purpose of performing distributed learning is simply to leverage more computational power. A more complex variation is represented by *active learning* where new information arrives during the learning process, and is combined with the offline training set [108, 109]. Applications include drone planning [93] and network management [110, 111].

A more recent trend is *Federated learning*, which tackles scenarios where participating devices are not required to share potentially sensitive data [98, 112]. Depending upon the specific scenario, new data may or may not arrive during the training process.

Several works propose generic methodologies to mitigate common hurdles of distributed ML, including scaling the parameter servers [92], dealing with slower nodes [105], and trading learning efficiency for convergence speed [104]. All these works propose novel algorithms and/or approaches to *adapt to* the existing network structure, e.g., by limiting the overhead, to perform the learning task at hand as efficiently as possible. Again, *none* of them envisions to do the opposite, i.e., adapting the nodes interaction to the learning task.

Some works seek to theoretically characterize the convergence of supervised ML and how it is influenced by the cooperation among learning nodes. The study in [95] characterizes the convergence of a wide class of multi-agent algorithms. Using tools from spectral graph analysis, it establishes a relation between the topology formed by pairs of cooperating nodes and the convergence of the algorithm they run. [106] focuses on distributed ML over regular topologies, and seeks to establish the graph degree associated with the shortest convergence *time* – as opposed to the lowest number of iterations –, finding that such a degree depends on the distribution of the nodes' computing time. Through similar steps and targeting a resource-constrained edge-computing scenario, [96] searches for the optimal trade-off between local computation and global parameter exchange in federated learning scenarios.

Hence, the goal of the work exposed in this chapter focuses on distributed, supervised learning, and aims at filling the gap and extending the literature by (i) adapting the logical network topology to the learning task, and (ii) considering not only learning nodes (in charge of processing information), but also information nodes, where data comes from. The latter is especially critical, as it allows to characterize and study the trade-off between

gathering information and extracting knowledge from it.

The rest of the chapter is organized as follows. The system model and how it can represent different supervised ML tasks is described in Section 4.1. Section 4.2 presents the formulation of the problem tackled and discusses its complexity. Section 4.3 characterizes the learning performance, while important properties of the problem are proven in Section 4.4. Then, the DoubleClimb algorithm is developed in Section 4.5 and its complexity is analyzed before evaluating its performance in Section 4.6.

4.1. System Model

The system model of this work addresses a generic distributed, supervised ML task where multiple nodes cooperatively seek to minimize a *loss function*, via gradient descent approaches such as the *stochastic gradient descent* (SGD) algorithm [94,96,106,113]. The behavior of individual nodes and their interactions are the most important characteristics described by this system model, with reference to different real-world ML approaches. A unique feature of our model is its ability to capture the presence of two different types of nodes:

- *learning nodes*, or L-nodes for short, that, having computational capabilities, run the ML algorithm and can exchange gradient data during learning; we denote their set by $\mathcal{L}$;
- *information nodes*, or I-nodes for short, which can provide information to the L-nodes; we denote their set by $\mathcal{I}$.

Real-world counterparts of L-nodes include physical servers and virtual machines running at the intelligent network edge [102] or in the cloud. I-nodes, on the other hand, represent such entities as monitoring platforms, network nodes, and sensors.

In our system model, L-nodes behave in a similar way to their equivalents in [96, 106]. Their high-level goal is to cooperatively train a ML model network, and do so by minimizing a loss function via distributed optimization. The computation time at each iteration of the learning process at a generic node $l \in \mathcal{L}$ follows an arbitrary distribution with Probabiliy Density Function (pdf) $\tau_l^k(t)$. Note that, in the most general case, such a pdf depends on the current iteration (k) of the learning process, since the amount of samples used for learning may vary from an iteration to the next one. This reflects the need to exploit all the available data as soon as it becomes available [109, 114], as opposed to training on a fixed number of samples as in more static scenarios. L-nodes are logically connected to form an arbitrary *logical topology*, i.e., a graph where vertices represent L-nodes and edges, hereinafter referred to as L-L edges, represent the logical links connecting them. As exemplified in Figure 4.1 (steps 3–4), after every iteration, each L-node sends its gradient data to its neighboring L-nodes on the logical topology, and waits for them to do the same before moving on. The logical topology, i.e., which

pairs of L-nodes are neighbors and exchange gradient data, is one of the main decision variables of this problem.

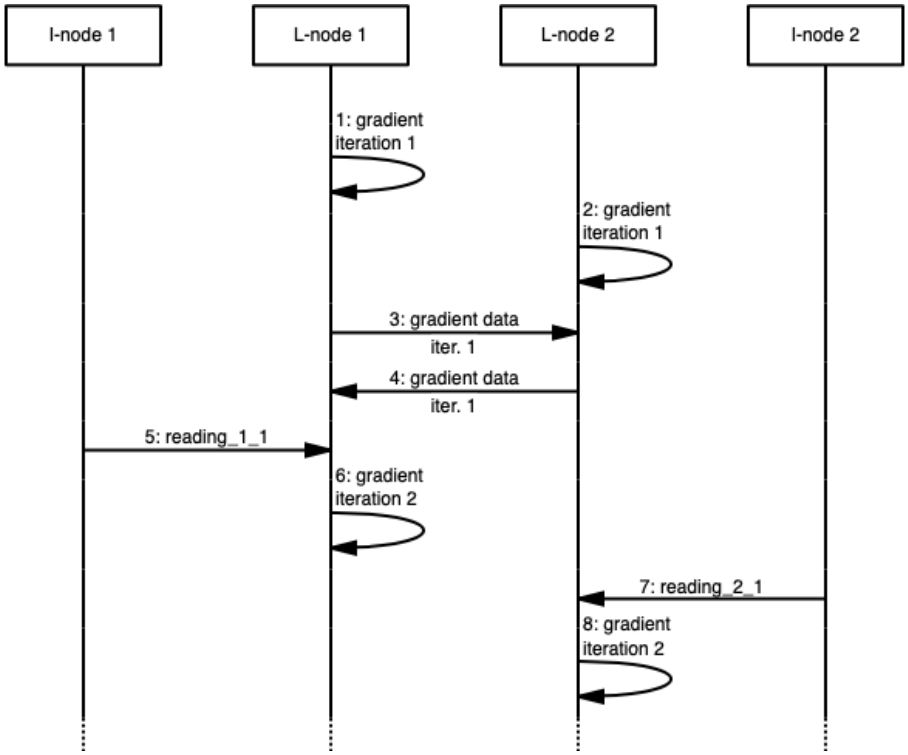


Figure 4.1: Scheme of the interactions between L- and I-nodes in a general case.

Each L-node can be logically connected to one or more I-nodes, through the so-called I-L edges. Only I-nodes that are connected to at least one L-node are added to the logical topology. After each iteration of the learning process, an L-node receives data from the I-nodes it is connected to (steps 5 and 7 in Figure 4.1). Each I-node i may provide new samples after a *sample generation time* since the end of the previous iteration, with r_i being the average number of provided samples and $\rho_i(t)$ the pdf describing the sample generation time. The received samples are used by an L-node l to perform the next learning iteration, in addition to the data it received in the previous iterations and the number X_l^0 of (offline) samples initially available at l. Note that this behavior is compatible with current, widely deployed applications (e.g., IoT) using publish/subscribe mechanisms, such as MQTT [115], or Zenoh [116], or even the notification mechanisms included in the 3GPP Service Based Architecture [117] of Release 15 and above.

Both L-nodes and I-nodes have per-iteration operational costs, denoted by c_l and c_i, respectively. Moreover, communication between nodes that are neighbors in the logical topology involve additional costs, denoted by $c_{l,l'}$ or $c_{i,l}$ depending on the type of nodes. The notation used to model the problem is gathered in Table 4.1 and Table 4.2.

Parameter	Meaning
$\mathcal{L}$, $\mathcal{I}$	L-nodes and I-nodes set (resp.)
$\rho_i(t)$	pdf of sample generation time at I-node $i \in \mathcal{I}$
r_i	ave. no. of samples per iteration by I-node i
X_l^k	amount of samples at the beginning of iteration k at L-node l
c_l, c_i	operational cost of L-node l and I-node i (resp.)
$c_{l,l'}$	communication cost between L-nodes l, l'
$c_{i,l}$	communication cost between I-node i and L-node l
$\epsilon^{\max}$	maximum learning error
$T^{\max}$	maximum duration of the learning process

Table 4.1: Main parameters of the model.

Variable	Meaning
$p(l,l')$	binary variable determining if L-nodes l and l' cooperate (matrix $\mathbf{P}$)
$q(i,l)$	binary variable determining if L-node node l obtains samples from I-node i (matrix $\mathbf{Q}$)
K	number of iterations to run
$\tau_l^k(t)$	pdf of the computation time at L-node l and iteration k
$\epsilon_l^k(\mathbf{P},\mathbf{Q})$	local error at L-node l and iteration k
$\epsilon^K(\mathbf{P},\mathbf{Q})$	global error at the end of the whole learning process
$T^K(\mathbf{P},\mathbf{Q})$	expected time to complete the whole learning process
$C^K(\mathbf{P},\mathbf{Q})$	global cost for running the whole learning process

Table 4.2: Main variables of the model.

4.1.1. Modeling Real-World Supervised ML tasks

As mentioned, the proposed model can describe a wide range of real-world ML tasks, falling in the category of *supervised learning*, for which a ground truth is available. The most prominent examples of supervised learning tasks are classification (where the quantity to predict is discrete, e.g., whether or not a given transaction is fraudulent) and regression (where the quantity to predict is continuous).

In a distributed setting, supervised learning can be performed in two main modes: (i) *distributed learning with static data*, where no new data arrive during the learning process. In this case there are no I-nodes (hence, no such steps as 5 and 7 in Figure 4.1), and each L-node learns from its X_l^0 initial samples, as well as the gradient data from the other

L-nodes; and (ii) *active learning* [108], where new samples can be collected from data sources (e.g., sensors) during the learning process so as to improve the learning quality. In this latter case, the network topology includes both L- and I-nodes.

Importantly, our model can also capture *federated learning* [96, 97, 118], an emerging paradigm whereby different devices (e.g., smartphones) cooperatively train a model without sharing (potentially sensitive) data. In this case, each device is modeled as an L-node; if, in the specific scenario at hand, devices collect or generate additional information while learning, an I-node per device is added, only connected to the corresponding L-node.

For all tasks and approaches, our model can capture the cases where the communication between nodes happens in a peer-to-peer fashion [95, 106], as well as those when it is mediated by a *parameter server*, also known as *broker* [96, 104, 118]. In the latter case, the logical topology created by the L-nodes is fully connected.

4.2. Problem Formulation

Our decisions concern which nodes' interactions should be enabled, and the number of iterations to execute during the learning process. We thus define the following decision variables:

- the set of binary variables $p(l, l') \in \{0, 1\}$, expressing whether L-nodes l and l' cooperate during learning;
- the set of binary variables $q(i, l) \in \{0, 1\}$, expressing whether L-node $l \in \mathcal{L}$ obtains samples from I-node $i \in \mathcal{I}$;
- the total number of iterations, K, to perform so that the learning task meets the desired learning quality and execution time.

For compactness of notation, we will collect the p- and q-variables in matrices $\mathbf{P} = \{p(l, l')\}$ and $\mathbf{Q} = \{q(i, l)\}$, respectively. Given the decisions $\mathbf{P}$, $\mathbf{Q}$, and K, we can compute the following system performance metrics:

- the expected time required to the system to complete the learning process, denoted by $T^K(\mathbf{P}, \mathbf{Q})$;
- the total cost $C^K(\mathbf{P}, \mathbf{Q})$, incurred by the system to complete the learning process;
- the (system-wide) learning error $\epsilon^K(\mathbf{P}, \mathbf{Q})$ at the end of the learning process (i.e., after K iterations).

It is important to point out that in general the concrete definition of error ϵ depends on the type of learning task being performed, e.g.,

- for classification tasks, $\epsilon \triangleq 1 - \alpha$, where α is the classification accuracy (i.e., the rate of correctly labeled items);
- for regression tasks, $\epsilon \triangleq 1 - R^2$, where R^2 is the coefficient of determination [119].

In both cases, $\epsilon = 0$ corresponds to perfect learning, while larger ϵ values identify worse learning quality, i.e., higher error. In the remainder of the chapter, we use *learning error* or *learning quality* when referring to generic machine learning, and more precise terms (e.g., *accuracy* for classification) when discussing specific learning tasks.

Our objective is to minimize the total cost, while ensuring that the final learning error does not exceed the limit $\epsilon^{\max}$, i.e., $\epsilon^K(\mathbf{P}, \mathbf{Q}) \leq \epsilon^{\max}$, and the learning is completed within the target time, i.e., $T^K(\mathbf{P}, \mathbf{Q}) \leq T^{\max}$. The problem can then be synthetically formulated as:

$$\min_{\mathbf{P},\mathbf{Q},K} C^K(\mathbf{P}, \mathbf{Q}), \tag{4.1}$$

$$\text{s.t.} \min \left\{ \frac{\epsilon^{\max}}{\epsilon^K(\mathbf{P}, \mathbf{Q})}, \frac{T^{\max}}{T^K(\mathbf{P}, \mathbf{Q})} \right\} \geq 1. \tag{4.2}$$

The problem is combinatorial in nature and includes a large number of binary variables (the elements of matrices $\mathbf{P}$ and $\mathbf{Q}$). This makes it very hard to solve, even without considering the complexity of computing the quantities $C^K(\mathbf{P}, \mathbf{Q})$, $\epsilon^K(\mathbf{P}, \mathbf{Q})$, and $T^K(\mathbf{P}, \mathbf{Q})$. Specifically, the problem is NP-hard and the proof can be found in Appendix B.2.

Remarkably, in spite of the problem complexity, we can design an efficient and provably effective solution strategy. It is done by first characterizing the system performance as functions of the problem decision variables (Section 4.3), and then showing that the problem in Equation (4.1) and Equation (4.2) is *submodular* (Section 4.4). Leveraging this result, we can devise the DoubleClimb algorithm (Section 4.5), which has cubic worst-case time complexity and proves to be $1 + 1/|\mathcal{I}|$ competitive.

4.3. Characterizing the Performance of the Learning Process

This section is devoted to characterize the learning error, execution time, cost, and number of iterations of the learning task at hand. We denote the number of samples available at L-node l at iteration k with X_l^k. Notice that such data is obtained by enhancing the amount of samples initially available at l, X_l^0, with the samples that l receives at each iteration from the I-nodes it is connected to.

To perform the characterization, we blend together results from the literature and our own experiments. Specifically, we performed and profiled the following learning tasks:

(i) a *classification* task on the famous MNIST digit database [120]; and (ii) a *regression* task on the dataset used for the ITU AI Challenge [121], with the goal of predicting the throughput of a set of Wi-Fi nodes leveraging their position and settings.

Through these two datasets, we can show how our methodology works for the two most common and relevant types of supervised learning. While the numerical results we obtain (e.g., the coefficient values) are specific to the concrete learning algorithm at hand, our approach is general and can be effortlessly extended to any supervised learning task.

Experiments have been performed using the Python language and the `sklearn` library, specifically, the `MLPClassifier` and `MLPRegressor` objects. The `sklearn` library does not support GPU, hence, only CPU is used for their training, which makes them easier to profile. All tests were run on a server based on a twenty-core Intel Xeon E5-2630V4 processor with 64 GByte of RAM.

4.3.1. Learning Time

In general, the learning error at each iteration depends on (i) the number of already performed iterations, and (ii) the number of available samples [96, 106]. To characterize such a dependence, we proceed in four steps:

1. we first focus on a single L-node, l, in a scenario where there are no I-nodes, and characterize the relation between the per-iteration error $\epsilon_l^k(\mathbf{P}, \mathbf{Q})$ and iteration k;
2. for the same scenario, we establish a relationship between the quantity X_l^0 of offline training data available at l and its final error $\epsilon_l^K(\mathbf{P}, \mathbf{Q})$;
3. we extend such a relation to account for I-nodes, i.e., the case where new samples arrive at each iteration;
4. we generalize the error to the case of multiple L-nodes.

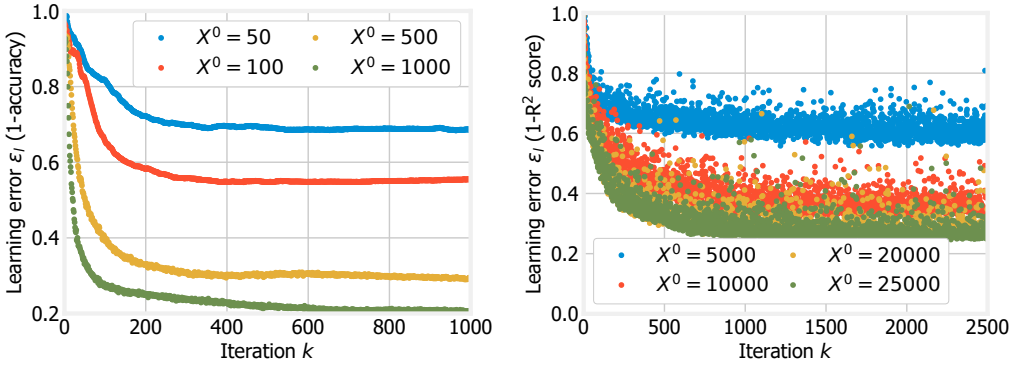
With regard to step 1), Figure 4.2 shows how the error (defined in terms of classification accuracy in Figure 4.2(a) and of coefficient of determination in the regression experiments in Figure 4.2(b)) evolves across iterations, when only X_l^0 offline data are used.

The evolution of the error as a function of iteration k is well captured by the following square-root relationship:

$$\epsilon_l^k(\mathbf{P}, \mathbf{Q}) = \epsilon_l^K + \frac{1}{\sqrt{k}}. \tag{4.3}$$

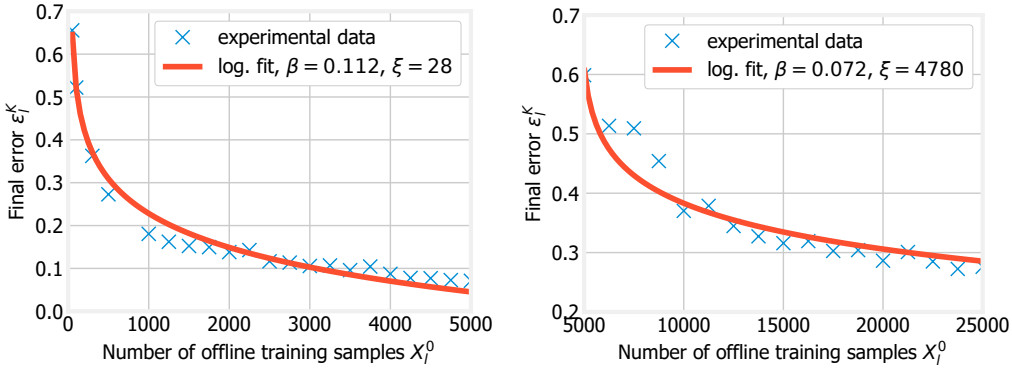
The relation in Equation (4.3) matches experimental data very well, with an Root Mean Square Error (RMSE) of 0.007 and 0.006 for the classification and regression tasks, respectively, in addition to conforming to the theoretical findings in [96, 106].

For step 2), we focus on the final value taken by the error at the end of the process, and on how this depends on the quantity X_l^0 of offline samples.



(a) Classification using MNIST dataset [120]. (b) Regression using ITU challenge dataset [121].

Figure 4.2: Evolution of the learning error for different values of X_l^0 when there are no I-nodes.



(a) Classification using MNIST dataset [120]. (b) Regression using ITU challenge dataset [121].

Figure 4.3: Values ϵ_l^K when there are no I-nodes and obtained fit.

As shown in Figure 4.3, the relationship between $\epsilon_l^K(\mathbf{P}, \mathbf{Q})$ and X_l^0 follows a logarithmic law:

$$\epsilon_l^K(\mathbf{P}, \mathbf{Q}) = 1 - \beta \log\left(X_l^0 - \xi\right). \tag{4.4}$$

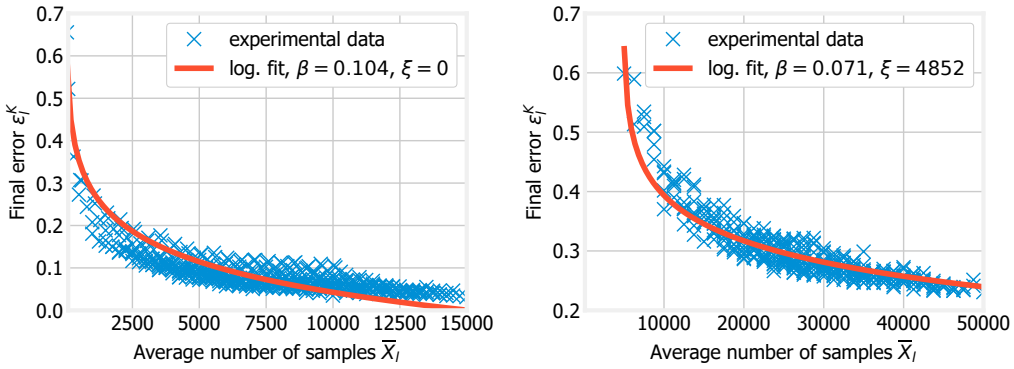
In our experiments, the best fit is obtained with $\beta = 0.112$, $\xi = 28$ for the classification task (RMSE 0.021) and $\beta = 0.072, \xi = 4780$ for the regression task (RMSE 0.029). Importantly, similar logarithmic laws can also be found in the literature [94, 108, 122].

In step 3), we move to a generic scenario where new data arrive at every iteration, i.e., $X_l^k \geq X_l^{k-1} \geq X_l^0$. We then update the above expression for $\epsilon_l^K(\mathbf{P}, \mathbf{Q})$ to account for the *average* number of samples available at the generic iteration at L-node l, $\overline{X}_l =$

$\frac{1}{K+1}\left(X_l^0 + \sum_{k=1}^{K} X_l^k\right)$, as:

$$\epsilon_l^K(\mathbf{P},\mathbf{Q}) = 1 - \beta \log\left(\overline{X}_l - \xi\right). \tag{4.5}$$

As summarized in Figure 4.4, using Equation (4.5) instead of Equation (4.4) also results in a very good fit, with RMSE of 0.022 for the classification task and of 0.023 for the regression one.



(a) Classification using MNIST dataset [120]. (b) Regression using ITU challenge dataset [121].

Figure 4.4: Values ϵ_l^K when I-nodes are present and obtained fit.

For step 4), we have to characterize the *aggregate* learning error of the whole set of L-nodes, each of which can have a different value of $\epsilon_l^K(\mathbf{P},\mathbf{Q})$. To this end, we leverage existing works [95, 106], linking the effectiveness of the distributed learning process with the graph formed by cooperating L-nodes, and more precisely with its spectral gap[1] γ. According to [95, 106], we can then write:

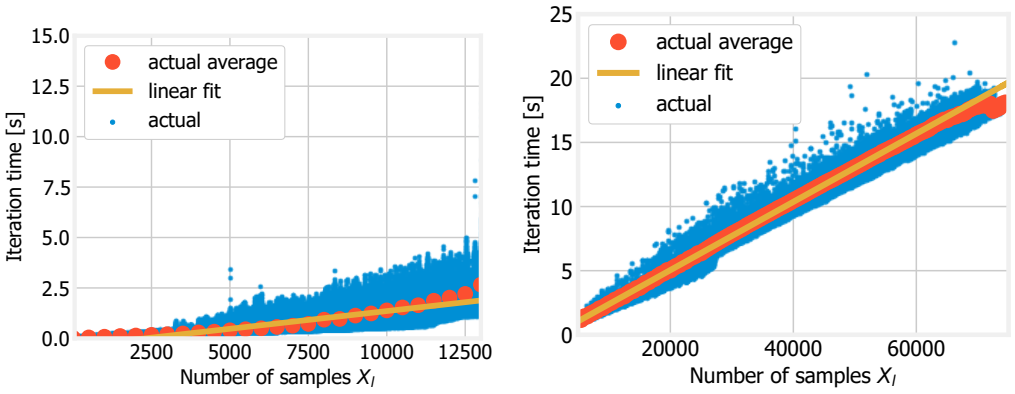
$$\epsilon^K(\mathbf{P},\mathbf{Q}) = \frac{1}{\gamma|\mathcal{L}|}\sum_{l\in\mathcal{L}} \epsilon_l^K(\mathbf{P},\mathbf{Q}). \tag{4.6}$$

4.3.2. Learning Time and Cost

We now consider that the total number of iterations K, the pdfs $\rho_i(t)$ of the sample generation time at each I-node i, and the pdfs $\tau_l^k(t)$ of the computation time of each L-node l at iteration k are given. Recall that $\tau_l^k(t)$ depends on k, as the presence of I-nodes in our system model implies that the computation time distribution must account for the quantity X_l^k of available data at L-node l and iteration k.

Backed by our experiments reported in Figure 4.5, as well as by the high efficiency and scalability of modern supervised ML algorithms, we consider the following relationship:

[1]The spectral gap of a graph is the difference between the moduli of the two largest eigenvalues of its adjacency matrix.



(a) Classification task using the MNIST dataset [120].

(b) Regression task using the ITU challenge dataset [121].

Figure 4.5: Duration of single iterations (each dot corresponds to one iteration) and linear fit.

$\tau_l^k(t) = \frac{X_l^k}{X^0}\tau_l^0(t)$. Also, we define the sets $\mathcal{I}_l = \{i \in \mathcal{I}\colon q(i,l) = 1\}$ and $\mathcal{L}_l = \{l' \in \mathcal{L}\colon p(l,l') = 1\}$ of I-nodes and L-nodes (resp.) each L-node is connected with. Our goal is to compute $T^K(\mathbf{P},\mathbf{Q})$, i.e., the total time required to complete the whole learning process.

As highlighted in Figure 4.1, at every iteration each L-node must perform the following steps: (i) wait for the information coming from the I-nodes $i \in \mathcal{I}_l$; (ii) perform its own gradient computation; and (iii) wait for the gradient data coming from the other L-nodes $l' \in \mathcal{L}_l$ it is cooperating with.

The first step is complete when *all* nodes in $\mathcal{I}_l$ send their samples. Recalling that each I-node has a sample generation time distributed with pdf $\rho_i(t)$, we can derive the Cumulative Distribution Function (CDF) of the maximum of a set of independent random variables as the product of individual CDFs $R_i(t)$, i.e., $\prod_{i\in\mathcal{I}_l} R_i(t)$. Once all data arrive, l can perform its own gradient computation, whose duration is distributed according to pdf $\tau_l^k(t)$. Recalling that the pdf of the sum of two independent random variables is the convolution of individual pdfs, we can write: $h_l^k(t) = \tau_l^k(t) * \frac{\mathrm{d}(\prod_{i\in\mathcal{I}_l} R_i(t))}{\mathrm{d}t}$.

For the system as a whole to move to the next iteration, all L-nodes must have received the gradient data they need. This, in turn, requires the slowest L-node to have obtained its information and have performed the computation. Working again with CDFs, the time taken by such a node is distributed according to: $H^k(t) = \prod_{l\in\mathcal{L}} H_l^k(t)$, where $H_l^k(t)$ denotes the CDF of the time to complete iteration k at L-node l. By letting $h^k(t) = \frac{\mathrm{d}H^k(t)}{\mathrm{d}t}$, the expected duration of the learning process is then given by:

$$T^K(\mathbf{P},\mathbf{Q}) = \sum_{k=1}^{K} \int_0^\infty x h^k(t) dt. \tag{4.7}$$

4.3.2.1. A numerical example.

Figure 4.6 exemplifies our methodology in a case where both the I-node sample generation times and the L-node computation times are uniformly distributed; specifically, $\rho_i(t) \sim \mathcal{U}(0.1, 1.9)$ and $\tau_l^k(t) \sim \mathcal{U}(1.35, 1.65)$. Furthermore, there are $|\mathcal{L}| = 10$ L-nodes, each connected to $|\mathcal{I}| = 5$ I-nodes.

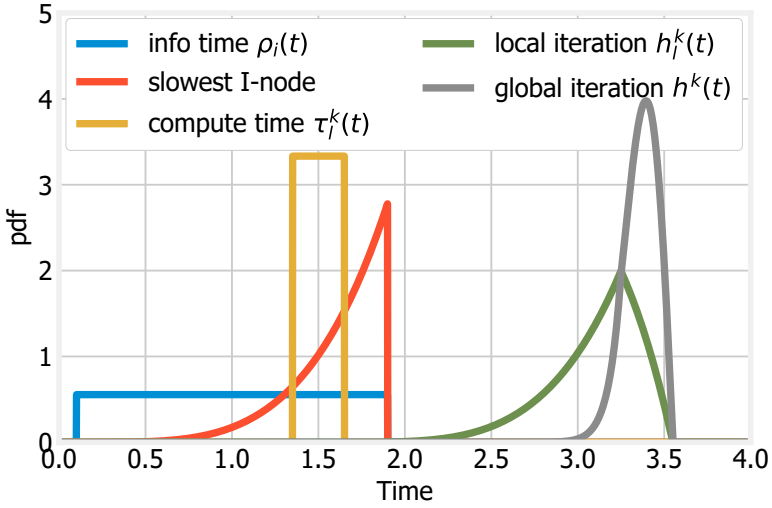


Figure 4.6: Toy scenario with $|\mathcal{L}| = 10$ and $|\mathcal{I}| = 5$ where both I-node sample generation times and L-node computation times are uniformly distributed. Left: pdfs of the I-node generation time $\rho_i(t)$ (blue), of the time required by the slowest I-node (red) and of the compute time $\tau_l^k(t)$ (yellow). Right: pdfs of the time taken by local (green) and global (gray) iterations.

We begin from the blue line in the plot, representing $\rho_i(t)$. To obtain the pdf of the sample generation time of the slowest I-node, we have to integrate $\rho_i(t)$ (obtaining $R_i(t)$, a ramp-like function), then raise it to the $|\mathcal{I}|$-th power (obtaining a 5th-degree polynomial), and finally derive it, obtaining the fourth-degree polynomial shown by the red line in Figure 4.6.

We next perform the convolution between the latter pdf and $\tau_l^k(t)$, represented by the yellow line in the plot. The result is $h_l^k(t)$, represented by the green line in Figure 4.6. The last step consists in computing the distribution of the time taken by the whole learning iteration, hence, by the slowest L-node. Integrating $h_l^k(t)$, we obtain $H_l^k(t)$, which we raise to the $|\mathcal{L}| = 10$-th power, and then derive it, obtaining the pdf $h^k(t)$ shown by the gray curve in Figure 4.6.

4.3.2.2. Closed-form Expression for Special Cases

The methodology outlined above does not require any assumption on the $\tau_l^k(t)$ and $\rho_i(t)$ distributions, nor on the logical links between nodes, and the computations

it requires can always be performed numerically. However, closed-form expressions are available in relevant special cases. As an example, when each L-node receives information from all I-nodes, the computation and the sample generation times are Independent and Identically Distributed (i.i.d.) and exponentially distributed with parameter λ_L^k and λ_I, respectively, we get:

$$T^K = -\sum_{k=1}^{K} \sum_{\substack{\mathcal{A}\subset\mathbb{N}:\\ |\mathcal{A}|=|\mathcal{I}|+2\\ \sum_{a\in\mathcal{A}} a=|\mathcal{L}|}} \binom{|\mathcal{L}|}{\mathcal{A}} \frac{\prod_{w=1}^{|\mathcal{I}|+2} (A^k(\mathcal{A},w))^{a_w}}{\lambda_I \sum_{w=1}^{|\mathcal{I}|} w a_w + \lambda_L^k a_{|\mathcal{I}|+2}}. \tag{4.8}$$

In the above expression, the sum over k accounts for all iterations, $k = 1, \ldots, K$. The inner sum comes from the multinomial expansion [123] of a sum of $|\mathcal{I}| + 2$ terms (one for each I-node, one for the L-node connected to them, and one representing the coefficient) raised to the $|\mathcal{L}|$-th power, where each term is a polynomial (see also the expression of $h_l^k(t)$). Therefore, the inner summation is over all sets $\mathcal{A}$ of natural numbers such that their size is $|\mathcal{I}|+2$ and their sum is $|\mathcal{L}|$, and $\binom{|\mathcal{L}|}{\mathcal{A}} = \frac{|\mathcal{L}|!}{\prod_{a\in\mathcal{A}} a!}$ is the multinomial coefficient. The term $A^k(\mathcal{A}, w)$ associated with the w-th element of each set $\mathcal{A}$ is:

$$A^k(\mathcal{A},w) = \begin{cases} \sum_{z=1}^{|\mathcal{I}|} \binom{|\mathcal{I}|}{z} (-1)^{z+1}, & \text{if } w{=}|\mathcal{I}|+1 \\ \sum_{z=1}^{|\mathcal{I}|} \binom{|\mathcal{I}|}{z} (-1)^{z+1} \frac{z\lambda_I}{\lambda_L^k - w\lambda_I}, & \text{if } w{=}|\mathcal{A}| \\ \binom{|\mathcal{I}|}{w} (-1)^{w+1} \frac{\lambda_L^k}{w\lambda_I - \lambda_L^k}, & \text{otherwise.} \end{cases}$$

A closed-form expression for the expected duration of the learning process can also be obtained when each L-node receives information from all I-nodes, and the I-nodes' sample generation times and the L-nodes's computation times are i.i.d. and uniformly distributed over (a_I, b_I) and (a_L^k, b_L^k), respectively. For simplicity and without loss of generality, let

us assume $a_L^k \leq a_I \leq b_I \leq b_L^k, \forall k$; then, we have:

$$T^K = \sum_{k=1}^{K} \sum_{\substack{\mathcal{A}\subset\mathbb{N}: \\ |\mathcal{A}|=|\mathcal{I}|+2 \\ \sum_{a\in\mathcal{A}} a=|\mathcal{L}|}} \binom{|\mathcal{L}|}{\mathcal{A}} \frac{\sum_{w=1}^{|\mathcal{I}|+1} w a_w}{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} \times \tag{4.9}$$

$$\times \Bigg[\prod_{w=1}^{|\mathcal{I}|+2} (A_1^k(\mathcal{A}, w))^{a_w} \left(Z_1^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} - Z_2^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} \right)$$

$$+ \prod_{w=1}^{|\mathcal{I}|+2} (A_2^k(\mathcal{A}, w))^{a_w} \left(Z_3^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} - Z_4^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} \right)$$

$$+ \prod_{w=1}^{|\mathcal{I}|+2} (A_3^k(\mathcal{A}, w))^{a_w} \left(Z_5^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} - Z_6^{\sum_{w=1}^{|\mathcal{I}|+1} w a_w + 1} \right) \Bigg]$$

where $Z_1 = a_L^k + b_I$, $Z_2 = a_L^k + a_I$, $Z_3 = b_L^k + a_I$, $Z_4 = a_L^k + b_I$, $Z_5 = b_L^k + b_I$, $Z_6 = b_L^k + a_I$. As in the previous case, the above expression comes from the multinomial expansion [123], and, after some algebra, one can obtain the terms $A_1^k(\mathcal{A}, w)$, $A_2^k(\mathcal{A}, w)$, and $A_3^k(\mathcal{A}, w)$ associated with the w-th element of each set $\mathcal{A}$, as:

$$A_1^k(\mathcal{A}, w) = \begin{cases} \frac{(-2a_I)^{|\mathcal{I}|} - \left(a_L^k - a_I\right)^{|\mathcal{I}|}}{(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)}, & \text{if } w{=}1 \\ \frac{\left(a_L^k - a_I\right)^{|\mathcal{I}|}(|\mathcal{I}|(a_L^k + a_I) + 2a_I) + (-2a_I)^{|\mathcal{I}|+1}}{(|\mathcal{I}|+1)(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)}, & \text{if } w{=}|\mathcal{A}| \\ \frac{\binom{|\mathcal{I}|+1}{w}(-2a_I)^{|\mathcal{I}|+1-w}}{(|\mathcal{I}|+1)(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)}, & \text{else.} \end{cases}$$

$$A_2^k(\mathcal{A}, w) = \begin{cases} A_1^k(\mathcal{A}, |\mathcal{A}|) + \sum_{z=1}^{|\mathcal{I}|+1} A_1^k(\mathcal{A}, z)(a_L^k + b_I)^z + \\ + \frac{\left(a_L^k - a_I\right)^{|\mathcal{I}|+1} - \left(a_L^k + b_I - 2a_I\right)^{|\mathcal{I}|+1}}{(|\mathcal{I}|+1)(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)} + \\ + \frac{(b_I + a_I)^{|\mathcal{I}|+1} - (2a_I)^{|\mathcal{I}|+1}}{(|\mathcal{I}|+1)(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)}, & \text{if } w{=}|\mathcal{A}| \\ \frac{-\binom{|\mathcal{I}|+1}{w}((-b_I - a_I)^{|\mathcal{I}|+1-w} - (-2a_I)^{|\mathcal{I}|+1-w})}{(|\mathcal{I}|+1)(b_I - a_I)^{|\mathcal{I}|}(b_L^k - a_L^k)}, & \text{else.} \end{cases}$$

$$A_3^k(\mathcal{A},w)=\begin{cases}\frac{(b_L^k-a_I)^{|\mathcal{I}|}-(b_I+a_I)^{|\mathcal{I}|}(-1)^{|\mathcal{I}|}}{(b_I-a_I)^{|\mathcal{I}|}(b_L^k-a_L^k)}, & \text{if } w{=}1\\ A_2^k(\mathcal{A},|\mathcal{A}|)+\sum_{z=1}^{|\mathcal{I}|+1}A_2^k(\mathcal{A},z)(b_L^k+a_I)^z- & \\ -\frac{(|\mathcal{I}|+1)\left(b_L^k-a_I\right)^{|\mathcal{I}|}(b_L^k+a_I)}{(|\mathcal{I}|+1)(b_I-a_I)^{|\mathcal{I}|}(b_L^k-a_L^k)}+ & \\ +\frac{\left(-b_L^k-a_I\right)^{|\mathcal{I}|+1}-\left(b_L^k-b_I\right)^{|\mathcal{I}|+1}}{(|\mathcal{I}|+1)(b_I-a_I)^{|\mathcal{I}|}(b_L^k-a_L^k)}, & \text{if } w{=}|\mathcal{A}|\\ \frac{\binom{|\mathcal{I}|+1}{w}(-b_I-a_I)^{|\mathcal{I}|+1-w}}{(|\mathcal{I}|+1)(b_I-a_I)^{|\mathcal{I}|}(b_L^k-a_L^k)}, & \text{else.}\end{cases}$$

Intuitively, the three different terms $A_*^k(\mathcal{A},w)$ are due to the convolution of the pdfs, which results in a piecewise function (see also the expression of $h_l^k(t)$). The support of the different pieces of the function are as follows: $\left[a_L^k+a_I, a_L^k+b_I\right)$ for the first piece where only one pdf is active, $\left[a_L^k+b_I, b_L^k+a_I\right]$ for the second piece where both pdfs are active and overlap, and $\left(b_L^k+a_I, b_L^k+b_I\right]$ for the third piece where only the other pdf is active.

4.3.3. Learning Cost

We define the per-iteration cost as the sum of operational and communication costs of the L- and I- nodes contributing to each iterations, i.e.,

$$C(\mathbf{P},\mathbf{Q})=\sum_{l\in\mathcal{L}}\left(c_l+\sum_{l'\in\mathcal{L}}c_{l,l'}p(l,l')+\sum_{i\in\mathcal{I}}c_{i,l}q(i,l)\right)+\sum_{i\in\mathcal{I}}c_i\mathbb{1}_{\exists q(i,l)>0}. \tag{4.10}$$

Then, we can write the total learning cost over the K iterations as $C^K(\mathbf{P},\mathbf{Q}) = K\cdot C(\mathbf{P},\mathbf{Q})$.

4.3.4. Number of Iterations

The number K of iterations needed to reach the target error $\epsilon^{\max}$ depends on two factors. The first is the quantity of available training data: the more data is available, the more the learning quality improves at each iteration. The second is the level of cooperation between L-nodes: the more nodes cooperate, the higher the quality achieved at each iteration. As shown in [106, Eq. (7)], we have $K\propto\frac{1}{\gamma}$, where γ is the spectral gap of the graph formed by L-nodes. Combining the two factors and denoting by $\overline{X}$ the average number of samples available at the generic iteration and L-node, as per Equation (4.5), we get:

$$K\propto-\frac{\log\overline{X}}{\gamma}. \tag{4.11}$$

As already noted in [106], on the one hand, a high degree of L-nodes makes the learning

process faster, as convergence requires fewer iterations; on the other hand, each iteration takes longer to complete as there are more nodes to wait for.

4.4. Problem Analysis

The problem at hand, formulated in Section 4.2, is NP-hard and the proof can be found in Appendix B.2. However, we also show that the problem objective function is submodular and non-decreasing, while the constraint is submodular and exhibits only one maximum (we prove the latter part separately for I-L and L-L edges).

In spite of its complexity, the problem of minimizing Equation (4.1) subject to constraint Equation (4.2) presents several features that can be exploited to solve it efficiently and effectively. Specifically, both the objective in Equation (4.1) and the constraint in Equation (4.2) are submodular (intuitively, the set-wise equivalent of convex [124]). Submodular optimization problems can often be solved with polynomial- or even linear-time greedy algorithms, with very good, even constant, competitive ratios [125].

Let us indicate with $f(\mathcal{Y})$ the *objective function* in Equation (4.1), and with $g(\mathcal{Y})$ the *constraint* in Equation (4.2). In our case, the set $\mathcal{X}$ of elements to choose from is given by $\mathcal{X} = \mathcal{L} \times \mathcal{L} \cup \mathcal{L} \times \mathcal{I}$, i.e., the set of possible I-L and L-L edges we can create, and $\mathcal{Y}$ is the subset of actually selected edges. The objective $f(\mathcal{Y})$ and constraint $g(\mathcal{Y})$ of our problem have several interesting and useful properties. Concerning the former, it is possible to prove the following result.

Property 1. *The objective function in Equation* (4.1) *is submodular and non-decreasing.*

Proof: Let $j = (a, b)$ be an edge in our logical topology graph, with $a \in \mathcal{L}$ and $b \in \mathcal{L} \cup \mathcal{I}$; let $\mathcal{S} \subset \mathcal{X}$ be the set of currently selected edges. By adding j, we incur the per-edge communication cost $c_{a,b}$; also, we may incur per-node operational costs c_a or c_b, depending on whether or not there are already edges in $\mathcal{S}$ with a or b as endpoints. Similar arguments hold for the cost of adding j to $\mathcal{T} \supset \mathcal{S}$. Thus,

$$\begin{aligned} f(\mathcal{S} \cup \{j\}) - f(\mathcal{S}) &= c_{a,b} + c_a \mathbb{1}_{a \notin \mathcal{S}} + c_b \mathbb{1}_{b \notin \mathcal{S}} \\ f(\mathcal{T} \cup \{j\}) - f(\mathcal{T}) &= c_{a,b} + c_a \mathbb{1}_{a \notin \mathcal{T}} + c_b \mathbb{1}_{b \notin \mathcal{T}}. \end{aligned}$$

Since $\mathcal{S}$ is a subset of $\mathcal{T}$, it also holds that $\mathbb{1}_{a \notin \mathcal{S}} \geq \mathbb{1}_{a \notin \mathcal{T}}$ and $\mathbb{1}_{b \notin \mathcal{S}} \geq \mathbb{1}_{b \notin \mathcal{T}}$, from which it follows that $f(\mathcal{S} \cup \{j\}) - f(\mathcal{S}) \geq f(\mathcal{T} \cup \{j\}) - f(\mathcal{T})$, i.e., the very definition of submodularity [124]. The fact that Equation (4.1) is non-decreasing trivially comes from the observation that, as more I-L or L-L edges are added, the cost always increases. ■

As for the constraint, the analysis is a little more complex, and we perform it separately for I-L and L-L edges. For simplicity of notation, we drop the dependency on $\mathbf{P}$ and $\mathbf{Q}$ while presenting our derivations.

Property 2. *When the choices are limited to I-L edges, i.e.,* $\mathcal{X} = \mathcal{L} \times \mathcal{I}$*, then the constraint in Equation* (4.2) *is submodular and has exactly one maximum.*

Proof: Let us study the two parts of the constraint Equation (4.2) separately, writing $g_1 = \frac{\epsilon^{\max}}{\epsilon^K}$, $g_2 = \frac{T^{\max}}{T^K}$, and $g(\mathcal{Y}) = \min\{g_1, g_2\}$, as exemplified in Figure 4.7. From Equation (4.5), $g_1 = \frac{\epsilon^{\max}}{1-\beta \log \overline{X}_l}$; also, adding an I-L edge increases $\overline{X}_l$ for at least an L-node l. Recalling that the logarithm is a concave function, the denominator of g_1 is convex, and g_1 itself is concave, which implies submodularity [124]. For analogous reasons, g_1 is also monotonically increasing.

The behavior of g_2 is more complex: we know from Equation (4.11) that the number of iterations decreases as $\overline{X}$ (hence, X_l^k) increases, according to an inverse-log law. Also, as shown in Section 4.3, $\tau_l^k(t)$ and $\frac{\mathrm{d}H^k(t)}{\mathrm{d}t}$ are proportional to X_l^k and $\prod_{l\in\mathcal{I}} X_l^k$, respectively. Thus, T^K is proportional to K and $\prod_{l\in\mathcal{I}} X_l^k$. Replacing K with Equation (4.11), we get that T^K behaves like $\frac{\prod_{l\in\mathcal{I}} X_l^k}{\log \overline{X}}$, i.e., it can be shown that it decreases until it reaches a minimum, and then increases. It follows that $g_2 = \frac{T^{\max}}{T^K}$ is concave, hence, submodular.

Looking now at $g(\mathcal{Y})$, the minimum of two submodular functions is not guaranteed to be submodular in general; however, since g_1 is not only submodular but also monotonically increasing, the submodularity of g_2 also implies that $g(\mathcal{Y})$ as a whole is submodular [124]. Next, consider the maximum of $g(\mathcal{Y})$, with the latter being equal to $\min\{g_1, g_2\}$. As exemplified in Figure 4.7(left), we know that g_1 starts from a value close to $\epsilon^{\max}$ and then monotonically increases towards infinity, while g_2 starts with a small value, increases until it has a global maximum, and then decreases again. If g_2 is always smaller than g_1, then $g(\mathcal{Y}) = g_2$ has exactly one global maximum, consistently with the hypothesis. If they cross (as in Figure 4.7)), they do so in exactly two points, say A and B, such that the maximum of g_2 is between A and B. Then, the following holds: (i) before A, $g(\mathcal{Y}) = g_2$, which is increasing before its maximum; (ii) between A and B, $g(\mathcal{Y}) = g_1$, which is always increasing; (iii) after B, $g(\mathcal{Y}) = g_2$ and, since we are after its maximum, $g(\mathcal{Y})$ is decreasing – hence, B is $g(\mathcal{Y})$'s only maximum. Therefore, in all cases $g(\mathcal{Y})$ is submodular and has exactly one maximum, and, until such a maximum is reached, $g(\mathcal{Y})$ is also monotonically non-decreasing. ■

As for L-L edges, their influence on the learning process can be quantified by studying the graph they form. Specifically, [95, 106] have shown that both the learning error and the learning time are inversely proportional to the *spectral gap* of such a graph, indicated by γ. Following the lead of [106] and restricting our attention to regular graphs, we can state the following result:

Proposition 1. *When the choices are limited to sets of L-L edges such that the graph created by L-nodes is uniform, then the constraint Equation* (4.2) *is submodular and has exactly one maximum.*

The arguments in support of Proposition 1 can be summarized as follows: 1) the error

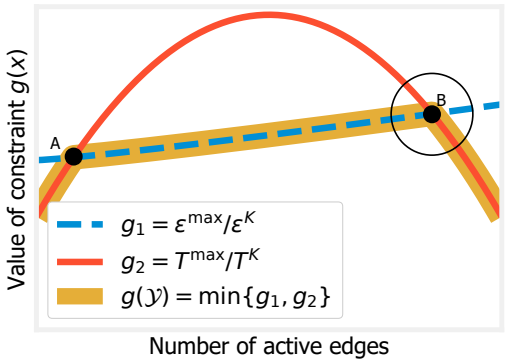


Figure 4.7: Qualitative example of the constraint in Equation (4.2) and its components.

reached after a given number K of iterations is proportional to $1/\gamma$ [106, Eq. (7)]; 2) the learning time is proportional to $1/\gamma$ [106, Eq. (18)]; 3) based on our own experiments, summarized in Figure 4.8, the link between the graph degree and the spectral gap γ is expressed by a concave function. Recalling that concavity is the continuous equivalent of submodularity, the first part of the proposition follows. The second part follows from the fact that, as exemplified in Figure 4.8, Equation (4.2) is the minimum between a monotonic function (as we add more L-L edges, the error decreases) and a function with at most one maximum (the inverse of the learning time, which decreases until an optimal degree is reached and then increases, as shown in [106]).

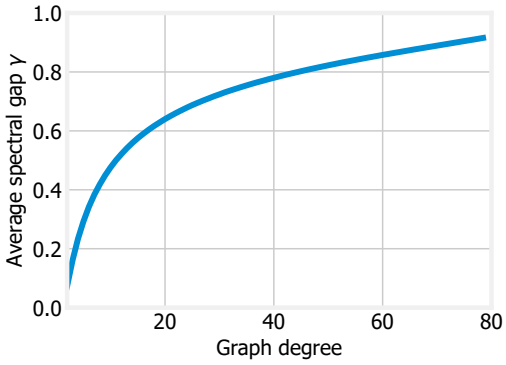


Figure 4.8: Experiments on the relation between the degree of a random graph with 100 vertices and uniform degree, and its spectral gap γ.

4.5. Heuristic Algorithm

In order to solve the problem stated in Section 4.2, i.e., determining the $\mathbf{P}$, $\mathbf{Q}$ and K resulting in the lowest cost Equation (4.1) subject to the constraint in Equation (4.2), in a practical and efficient way, we first extend existing results on the performance of greedy algorithms when optimizing submodular problems, in Section 4.5.1. Based on

such results, we present our own DoubleClimb algorithm in Section 4.5.2, and analyze its properties in Section 4.5.3.

4.5.1. Greedy Solutions to Submodular Problems

Algorithm 5 Greedy algorithm for submodular problems

1: $\mathcal{S} \leftarrow \emptyset$
2: **while** $g(\mathcal{S}) \geq c$ **do**
3: $\quad j^* \leftarrow \arg\min_{\mathcal{X}\setminus\mathcal{S}} \frac{c_j}{g(\mathcal{S}\cup\{j\})-g(\mathcal{S})}$
4: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
5: **return** $\mathcal{S}$

Let us consider Algorithm 5, which solves submodular problems with non-decreasing objective and constraints. More formally, it selects a subset $\mathcal{S} \subseteq \mathcal{X}$ of elements subject to a submodular non-decreasing constraint $g(\mathcal{S}) \geq 1$, while minimizing a submodular non-decreasing cost function $f(\mathcal{S})$. At every iteration, Line 3 selects the element minimizing the cost to benefit ratio $\frac{f(\mathcal{S}\cup\{j\})-f(\mathcal{S})}{g(\mathcal{S}\cup\{j\})-g(\mathcal{S})}$; such an element is then added to $\mathcal{S}$ (Line 4). As shown in [126, Thm. 4.7], Algorithm 5 is $1+\frac{1}{|\mathcal{X}|}$-competitive. However, the original proof requires both the objective and the constraint to be submodular and non-decreasing. In our case, Property 2 and Proposition 1 prove *weaker* properties, in that our constraint is not guaranteed to be non decreasing, as in Figure 4.7; therefore, the result in [126] cannot immediately be applied to our problem.

None the less, it is possible to prove that a less restrictive condition than being non-decreasing, namely, having only one maximum, is sufficient for the result to hold:

Property 3. *If $f(\mathcal{Y})$ is submodular non-decreasing and $g(\mathcal{Y})$ is submodular and has only one maximum, then the above algorithm minimizes $f(\mathcal{Y})$ s.t. $g(\mathcal{Y}) > 0$, with a competitive ratio of $1+\frac{1}{|\mathcal{X}|}$.*

Proof: The property generalizes the results in [126, Thm. 4.7]. The proof therein follows from analyzing the steps of the above algorithm until its convergence, and leveraging the fact that the sequences of marginal cost increases and constraint improvements are (resp.) monotonically non-decreasing and monotonically non-increasing. This is of course true if, as in the original hypotheses, $g(\mathcal{Y})$ is monotonically non-decreasing. However, this also holds if $g(\mathcal{Y})$ has only one maximum, as per the hypothesis of our property. This is because, if the algorithm cannot find a feasible solution before the maximum of $g(\mathcal{Y})$, i.e., as constraints become *closer* to being satisfied, it will also be impossible to find a feasible solution after the maximum, i.e., when constraints will get *farther* from being met. Thus, the sequences of marginal costs and improvements *of the selected elements* of $\mathcal{X}$ have the required behavior. Indeed, the behavior of $g(\mathcal{Y})$

for the non-selected items of $\mathcal{X}$ *has no impact* on the validity of [126, Thm. 4.7], nor of this property. ■

4.5.2. The DoubleClimb

Property 3 implies that the algorithm in Section 4.5.1 could efficiently select $\mathbf{P}$ and $\mathbf{Q}$, *if such decisions could be made independently.* However, they are clearly interlinked; thus, we propose a more complex solution strategy, called DoubleClimb, which operates as follows.

- First, based on the nodes capabilities defined in Section 4.1, DoubleClimb determines $\mathbf{P}$ and $\mathbf{Q}$. It does so by selecting I-L and L-L edges in two nested loops, with L-L edges resulting in a uniform graph [106]. It also selects the most appropriate value of K for each set of selected edges.

- Given such decisions, it computes the system performance characterized in Section 4.3, thus yielding the error $\epsilon^K(\mathbf{P}, \mathbf{Q})$, the learning time $T^K(\mathbf{P}, \mathbf{Q})$, as well as the cost $C^K(\mathbf{P}, \mathbf{Q})$.

- It then compares the obtained values for the learning time and error against the limits $\epsilon^{\max}$ and $T^{\max}$, and evaluates whether a sufficiently low cost has been achieved. If so, DoubleClimb returns the problem solution; otherwise, it tries to improve the decisions until the system constraints are met and the cost is further reduced.

Algorithm 6 The DoubleClimb algorithm

1: $d_{\mathrm{L}} \leftarrow 0$
2: $\texttt{best\_sol} \leftarrow \emptyset$
3: **while** $d_{\mathrm{L}} < |\mathcal{L}|$ **do**
4: $\quad d_{\mathrm{L}} \leftarrow d_{\mathrm{L}} + 1$
5: $\quad \texttt{ll} \leftarrow$ **cheapest_uniform**(d_{L})
6: $\quad \texttt{il} \leftarrow \emptyset$
7: $\quad$ **while** Equation (4.2) is not verified $\wedge \texttt{il} \not\equiv \mathcal{I} \times \mathcal{L}$ **do**
8: $\quad\quad i^*, j^* \leftarrow \arg\min_{i,l} \frac{c_{i,l}}{g(\texttt{il}) - g(\texttt{il} \cup \{(i,l)\})}$
9: $\quad\quad \texttt{il} \leftarrow \texttt{il} \cup \{(i^*, j^*)\}$
10: $\quad$ **if** $C^{\mathrm{curr}} < C^{\mathrm{best}}$ **then**
11: $\quad\quad \texttt{best\_sol} \leftarrow \texttt{ll} \cup \texttt{il}$
12: $\quad$ **else if** $C^{\mathrm{curr}}_{\mathrm{LL}} > C^{\mathrm{best}}_{\mathrm{LL}} \wedge C^{\mathrm{curr}}_{\mathrm{IL}} > C^{\mathrm{best}}_{\mathrm{IL}}$ **then**
13: $\quad\quad$ **break**
14: **return** $\texttt{best\_sol}$

The DoubleClimb algorithm is presented in Algorithm 6 and detailed below. It begins (Line 1) by setting to zero the degree d_{L} of the subgraph made of L-L edges, and to the

empty set the best solution `best_sol`. Then, while $d_{\mathrm{L}} < |\mathcal{L}|$, i.e., while such a subgraph is not a clique, d_{L} is first incremented by one (Line 4), and then the cheapest L-L uniform subgraph of degree d_{L} is chosen in Line 5.

Given such a choice of L-L edges, the algorithm selects the I-L edges essentially in the same way as described in Section 4.5.1: for all possible edges, the cost/benefit ratio – i.e., the ratio between the cost of adding the edge and how closer to feasibility the problem becomes by doing so – is computed in Line 8, and the edge associated with the lowest ratio is chosen. The loop continues until either all I-L edges are exhausted, or a feasible solution, satisfying constraint Equation (4.2), is found (Line 7). In the latter case, the cost of the current solution C^{curr}, computed as per Equation (4.10), is compared to the one of the best solution found so far (C^{best}); note that, by convention, the cost of the empty set is equal to ∞. If warranted, the best solution is updated (Line 11), otherwise we perform the check in Line 12 to assess whether other solutions should be explored. Indeed, as proven in Proposition 2 below, the submodularity of costs implies that trying higher values of d_{L} does not lead to cheaper solutions.

If neither happens, the next value of d_{L} is tried. After all values of d_{L} are exhausted, the best solution `best_sol` is returned in Line 14. If no feasible solution has been found, the problem instance is infeasible and the algorithm returns $\emptyset$.

4.5.3. Algorithm Analysis

We now prove that Algorithm 6 has an excellent competitive ratio as well as low complexity. As first step, we show that the stopping condition in Line 12 is valid, i.e., no solution better than `best_sol` is ignored by halting the algorithm when the condition is met.

Proposition 2. *If the condition specified in Line 12 of Algorithm 6 is met, then no solution cheaper than* `best_sol` *will be found for higher values of* d_L.

Proof: Let $d_{\mathrm{L}}^{\mathrm{best}}$ be the value of d_{L} for which the current best solution was found, and $C_{\mathrm{LL}}^{\mathrm{best}}$ and $C_{\mathrm{IL}}^{\mathrm{best}}$ the corresponding costs for L-L and I-L edges (resp.). At the current iteration, we have $d_{\mathrm{L}} = L^{\mathrm{curr}} > L^{\mathrm{best}}$, and the corresponding costs are $C_{\mathrm{LL}}^{\mathrm{curr}} > C_{\mathrm{LL}}^{\mathrm{best}}$ and $C_{\mathrm{IL}}^{\mathrm{curr}} > C_{\mathrm{IL}}^{\mathrm{best}}$. Let us now consider a future iteration where the value of d_{L} is $d_{\mathrm{L}}^{\mathrm{next}} > d_{\mathrm{L}}^{\mathrm{curr}} > d_{\mathrm{L}}^{\mathrm{best}}$. $C_{\mathrm{LL}}^{\mathrm{next}}$ depends on two effects: if we increase the number of L-L edges, the cost due to L-L edges will increase. However, more L-L edges also imply fewer iterations, thus they may lead to a reduced cost. Since similar observations hold for $C_{\mathrm{IL}}^{\mathrm{next}}$, which effect prevails depends on how strong the benefit of increasing d_{L} is. However, as per the submodularity property (Proposition 1), the benefit of adding L-L edges and I-L edges decreases as d_{L} increases: if moving from $d_{\mathrm{L}}^{\mathrm{best}}$ to $d_{\mathrm{L}}^{\mathrm{curr}}$ actually increased the cost of L-L and I-L edges, it is not possible that moving to $d_{\mathrm{L}}^{\mathrm{next}}$ will provide a better solution. ■

Thanks to Proposition 2 and Property 3, we can now prove our main result about Algorithm 6.

Theorem 2. *Algorithm 6 has* $1 + \frac{1}{|\mathcal{I}|}$ *competitive ratio.*

Proof: There are two possible sources of suboptimality, namely, the choice of d_{L} and that of the I-L edges to select. By Proposition 2 and considering that, if the condition in Line 12 is never triggered, all possible values of d_{L} are tried out, the choice of d_{L} is optimal. As for the I-L edges, Line 7–Line 9 of Algorithm 6 reflect exactly the same algorithmic steps reported in Section 4.5.1 which, as per Property 3, lead to a $1 + \frac{1}{|\mathcal{I}|}$ competitive ratio in our case. ■

Finally, we can prove that Algorithm 6 has a very low, namely, cubic *worst-case* computational complexity.

Property 4. *Algorithm 6 has a worst-case computational complexity of* $O(|\mathcal{L}|^2|\mathcal{I}|)$.

Proof: From inspection of the nested loops in Algorithm 6, one can see that the outer one is run at most once for each value of d_{L}, i.e., at most $|\mathcal{L}|$ times. The inner one is ran at most once for each possible I-L edge, i.e., at most $|\mathcal{L}||\mathcal{I}|$ times. As for the set of edges to activate for each value of d_L (function **cheapest_uniform** in Line 5), they can be pre-computed and thus do not influence the overall complexity. ■

It is also worth stressing that Property 4 concerns the *worst-case* complexity, but the actual one is often much lower. Indeed, in Line 7–Line 9 we are likely to compute the same costs in different iterations; if such costs are cached, *à la* dynamic programming, run time can be dramatically decreased, to be slightly more than linear in $|\mathcal{L}| + |\mathcal{I}|$.

4.6. Validation and Application of the Approaches

In the following, we describe the reference scenario and benchmark solutions we consider (Section 4.6.1), before studying the performance of DoubleClimb (Section 4.6).

4.6.1. Reference Scenario

We consider an Internet-of-things (IoT) environment similar to the one referred in [96], whereby:

- individual sensors produce samples, either periodically or as a reaction to an external event;
- *aggregators*, also known as gateways, collect and summarize the samples, before forwarding them in uplink;
- distributed ML algorithms, running at the edge of the network, leverage the samples to gather insights on the changes in external conditions.

In terms of our system model, aggregators correspond to I-nodes, and edge nodes running the ML algorithms correspond to L-nodes. New samples arrive every few seconds, and updating the gradient computations takes a comparable time. Note that similar approaches have been proposed for such applications as smart-city monitoring [127], support of connected vehicles [128], and attack/anomaly detection [129].

With reference to the taxonomy of approaches for supervised learning exposed at the begining of the chapter, we fall in the *active learning* case, as the data arrival and gradient computation are interleaved but not synchronized, e.g., new data can arrive both before and after a gradient computation is complete.

We refer to the real-world urban topology presented in [10] and shown in Figure 4.9, depicting the network of a major operator. Specifically, the network nodes represented in brown act as aggregators, hence, as I-nodes, while those in blue are edge nodes acting as L-nodes. As shown in Figure 4.9, all L-nodes can be connected with one another, while each I-node can only be connected to one L-node.

Normalized sample generation and gradient computation times are distributed exponentially with mean 1, while the I-L and L-L edges are randomly assigned a normalized cost between 0 and 1 units. I- and L-nodes have no operational cost, reflecting the fact that, in our reference environment, they cannot be switched off without discontinuing the service. In the *basic* version of the scenario, at every iteration each I-node generates between 10 and 100 samples; such a value is proportional to the traffic served by each node in the real-world topology [10]. In the *rich* scenario, representing applications where data is more plentiful, such a value is multiplied by five.

Benchmark solutions. We compare DoubleClimb against two benchmark solutions. The first, called Opt-Unif, follows the approach used (among others) by [106], and returns the cheapest solution among the feasible ones such that both the graphs formed by L-L and I-L edges have uniform degree.

The second benchmark, labeled as "Optimum/GA" in the plots, performs the selection of the I-L edges (i.e., the inner loop in Algorithm 6) leveraging a Genetic Algorithm (GA) approach with the following parameters:

- number of generations: 50;
- solutions per population 100;
- parents mating: 4;
- mutation probability: 15%;
- crossover type: single point;
- gene space: $\{0, 1\}$;
- number of genes: $|\mathcal{I}||\mathcal{L}|$.

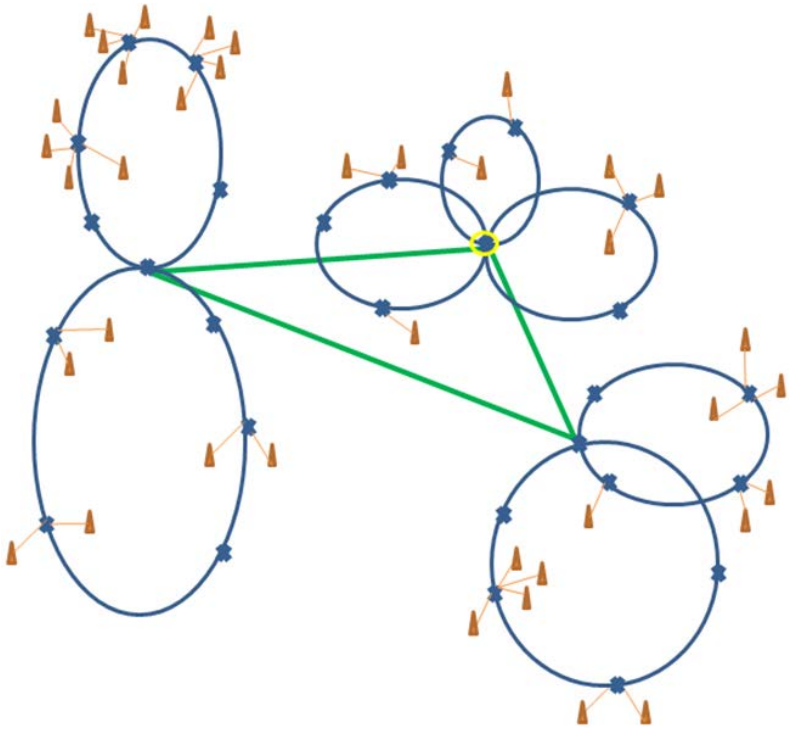


Figure 4.9: Our reference topology, depicting the network of a major operator (source: [10]).

Each solution corresponds to a string of binary values whose length equals the number of possible I-L edges: having a 1 in a given position means that the corresponding I-L edge is activated. The relatively large mutation probability reflects the importance of exploring multiple different solutions (i.e., exploration), given the combinatorial nature of the problem at hand and the fact that similar strings do not necessarily yield similar performance. When the size of the problem made it possible (i.e., $d_L \leq 6$), we have compared the performance of the genetic algorithm against the optimum obtained through brute force, and found that the two closely match.

4.6.2. Performance Comparison

The first plot in Figure 4.12 shows the cost of DoubleClimb and its benchmarks, for different numbers of L-nodes. As expected, the cost increases with $|\mathcal{L}|$ and decreases in the rich scenario, where the higher quantity of data results in faster convergence. Also, it is clear that DoubleClimb outperforms Opt-Unif and matches the performance of Optimum/GA. GA approaches are not, in general, guaranteed to yield optimal performance; therefore, we cannot conclude that DoubleClimb makes optimal decisions other than for $d_L \leq 6$, when the comparison with brute force was possible. However, GA approaches have long been known to be remarkably good at finding optimal or near-optimal solutions for combinatorial problems such as the one at hand, at the price of long

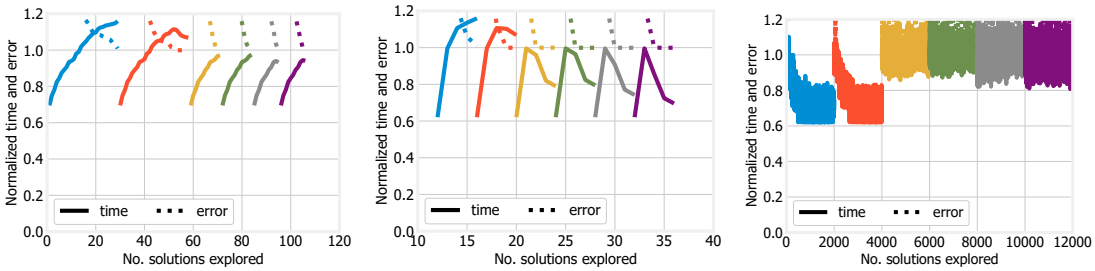


Figure 4.10: Normalized time and error of the solutions examined at each iteration by DoubleClimb (left), Opt-Unif (center), and GA (right), in the basic scenario. Different colors correspond to different values of d_{L}, as in Figure 4.13.

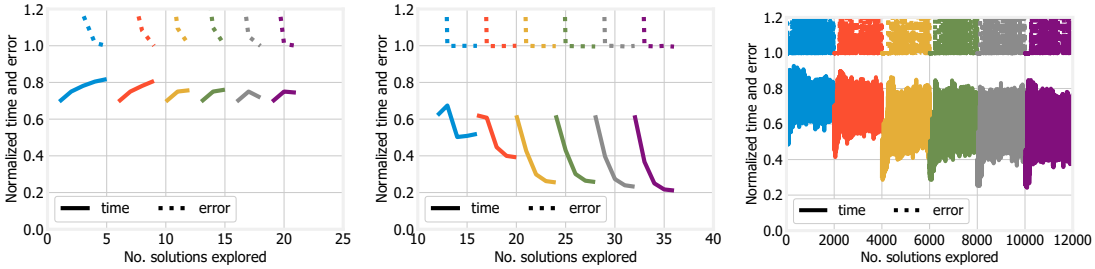
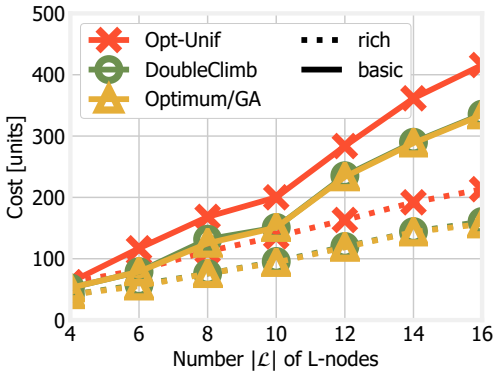


Figure 4.11: Normalized time and error of the solutions examined at each iteration by DoubleClimb (left), Opt-Unif (center), and GA (right), in the rich scenario. Different colors correspond to different values of d_{L}, as in Figure 4.13.

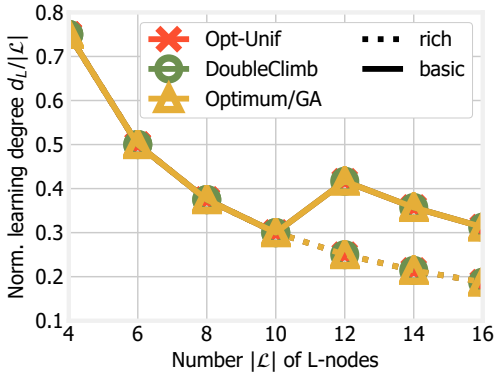
run times, as shown in Figure 4.10 and Figure 4.11 next. Observing that DoubleClimb matches Optimum/GA in all scenarios and for all values of d_L therefore boosts our confidence in the algorithm's effectiveness.

We now look deeper into the decisions made by each strategy. The second plot in Figure 4.12 depicts the selected value of d_{L}, normalized to $|\mathcal{L}|$. Interestingly, such a value is lower in the rich scenario, confirming our intuition that a tighter cooperation between L-nodes and more data coming from I-nodes are, to an extent, alternative solutions to achieve faster learning. DoubleClimb and Opt-Unif make exactly the same decisions in all cases, which suggests that the difference in cost shown in the first plot only comes from the choice of I-L edges. Accordingly, the third plot in Figure 4.12, depicting the fraction of I-L edges selected by each strategy, highlights how DoubleClimb uses substantially fewer edges than Opt-Unif. This highlights how the greater flexibility in the choice of I-L edges is an important asset of our approach, allowing us to beat state-of-the-art alternatives.

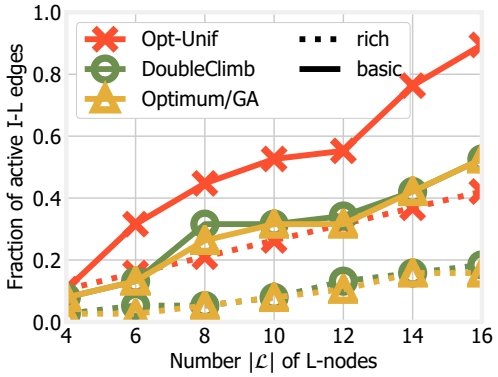
The fourth plot in Figure 4.12 shows how DoubleClimb not only uses fewer I-L edges, but also chooses the *right* ones. The plot depicts the number of new samples arriving at each iteration and highlights how, in spite of the substantially smaller number of selected I-L edges, DoubleClimb obtains a similar number of samples as Opt-Unif. Such an effect is especially evident for the basic scenario, where the number of samples provided by each I-node is smaller.



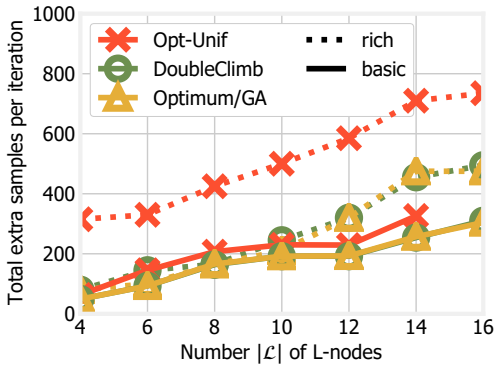
(a) Comparison experiment: total cost.



(b) Comparison experiment: selected value of d_L, normalized (to the maximum).



(c) Comparison experiment: fraction of selected I-L edges.



(d) Comparison experiment: total number of extra samples per iteration.

Figure 4.12: Comparison between DoubleClimb, Opt-Unif and the optimum (obtained via brute-force) in the basic and rich scenarios, for different values of $|\mathcal{L}|$.

Comparing the DoubleClimb and Optimum/GA curves, we can observe that in some cases Optimum/GA can activate slightly fewer I-L edges than the base scenario, e.g., for $d_L = 8$. This corresponds to solutions that DoubleClimb is unable to reach due to its hill-climbing nature; however, the impact on the overall cost (see the first plot in Figure 4.12) is negligible. Interestingly, DoubleClimb and Optimum/GA make the very same decisions in the rich scenario, confirming the somehow counterintuitive notion stated in Property 3, i.e., that, the solutions yielded by DoubleClimb tend to be closer to the optimum.

In Figure 4.13, we seek to better understand how DoubleClimb and Opt-Unif operate. Every marker in the plots corresponds to one solution examined by the algorithms; feasible solutions are denoted by a silver circle, the cheapest of such solutions is denoted by a black star. Note that Opt-Unif explores fewer solutions than DoubleClimb, as it is restricted to creating uniform logical topologies. Also, under the rich scenario it is easier for DoubleClimb to reach a high-quality solution, hence, the algorithm ends earlier.

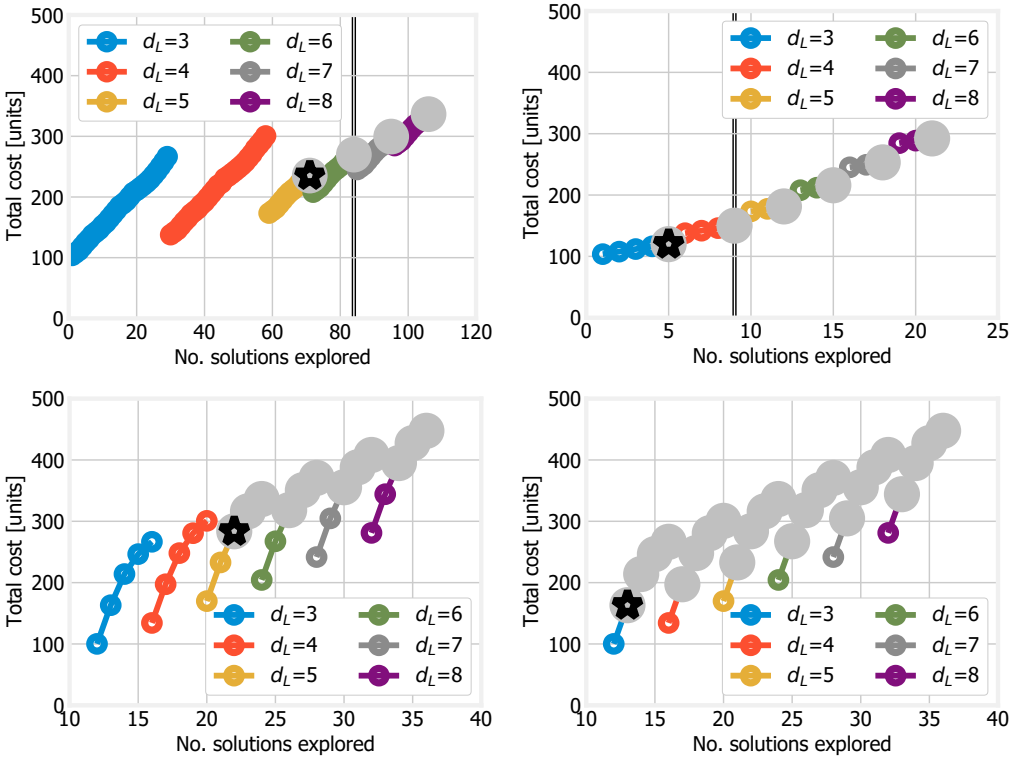


Figure 4.13: Cost of the solutions examined at each iteration by DoubleClimb (first two plots) and Opt-Unif (last two plots), in the basic (first and third plot) and rich (second and fourth plot) scenarios.

The first two plots, representing DoubleClimb in the basic and the rich scenario, respectively, clearly depict the behavior of Algorithm 6. The algorithm begins with the lowest possible value of d_{L} and no I-L edges, hence, with a low cost. Then, new edges are added until either a feasible solution is found, or all I-L edges are exhausted (as it happens in the first plot, representing the basic scenario). The double vertical lines in the first two plots correspond to the triggering of the condition in Line 12 of Algorithm 6; the plots confirm that enforcing such a condition does not result in ignoring cheaper feasible solutions.

The last two plots in Figure 4.13 represent Opt-Unif (again in the basic and the rich scenario, resp.), and clearly highlight its differences from DoubleClimb. As mentioned, Opt-Unif tries fewer solutions; also, multiple feasible solutions are tried out for the same value of d_{L}, since there is no stopping criterion analogous to the one in Line 13 in Algorithm 6. Importantly, the feasible solutions explored by Opt-Unif are more costly than those explored by DoubleClimb for the same value of d_{L}, a further confirmation of the importance of a flexible choice of I-L edges.

Last, in Figure 4.10 and Figure 4.11, we examine the error and learning time associated with each of the solutions examined by DoubleClimb and its benchmark solutions,

respectively in the basic and rich scenarios. Both quantities are normalized to their respective limits, thus both lines do not exceed 1 if the corresponding solution is feasible. It is interesting to note how adding I-L edges (moving from one solution to the next) affects error and time. The former (dotted lines) steadily decreases until its limit is reached, and then stays constant – recall that the learning process is interrupted upon reaching $\epsilon^{\max}$, so the normalized error never drops substantially below 1. The time (solid lines) increases at first, owing to the need to wait for more I-nodes; then, it decreases due to the fact that learning can be completed with fewer iterations. Importantly, both behaviors exactly match those described in Property 2 for g_1 and g_2. The third plots of both Figure 4.10 and Figure 4.11 highlight the behavior of GA approaches, which try multiple different solutions of varying quality and, in the interest of exploration, tend not to abandon low-quality solutions, on the grounds that they may mutate into high-quality solutions at some later stage.

Finally, the x-axis in those plots reminds us of the very high efficiency of DoubleClimb, where the number of examined solutions is orders of magnitude lower than in Optimal/GA. Recalling that GA algorithms themselves examine a number of solutions that is orders of magnitude lower than exact algorithms, the plots further highlight the efficiency of DoubleClimb, coupled with the effectiveness shown in Figure 4.12.

4.7. Conclusion

This chapter addressed the problem of defining an optimal level of cooperation among network nodes performing a supervised learning task. First, a system model was developed accounting for the presence of both learning nodes and information nodes interacting with each other. Then, it has been formulated the problem of choosing which learning nodes should cooperate to complete the learning task, and the information nodes that should provide them with data, as well as the number of iterations to perform. Although being NP hard, some important properties of the presented problem have been shown, most notably its submodularity, which allowed to define a solution algorithm that has cubic *worst-case* time complexity and is $1 + 1/|\mathcal{I}|$-competitive, with $\mathcal{I}$ being the set of information nodes. Numerical results also show that the proposed approach closely matches the optimum and outperforms state-of-the-art solutions, for both classification and regression tasks.

5 Conclusions

In this thesis, we have presented several works that aim at optimizing different aspects of the telecommunication networks in order to help the future 5th generation of telecommunications networks (5G) deployments to be more efficient both for the users and the operators. As defined in Chapter 1, the thesis is structured from the core of the network to the users matching with the timeline of the research and development of the 5G as well as with the moment of improving each part of the network. Thus, as a logical order, the first part that needed to be redefined was the core of the network to be flexible to address the continuous changes in the future demands. Once the most critical part was redefined, the challenge moved to reduce the overload in the central segment by offloading part of the load closer to the user, through a cloud/edge network able to host and perform the demands of end users. After reducing the load of the core and moved functionalities to the cloud/edge networks, new challenges appeared in those decentralized networks in which the processing of the services is split among different servers. The next challenge appreared on those edge networks, in particular to determine the connections of the distributed nodes to optimize the joint learning tasks of IoT devices. Thus, this thesis as a whole presents an optimization of the network, supporting the transition from the previous generation of networks to the new one in an optimized manner. Hence, the different chapters of this document have provided a solution to the different challenges exposed in Chapter 1.

First of all, Chapter 2 has addressed the first challenge exposed in Chapter 1 of optimizing the core of the networks by developing a framework for the joint optimization of an integrated networking and edge/cloud environment supporting two diverse classes of flows (fronthaul/backhaul) under path and delay constraints. This framework is directly applicable to the optimal design or dynamic management of a mixed Radio Access Network (RAN) and Cloud or Centralized Radio Access Network (C-RAN) environments. Thus, it has been tested over a synthetic small-scale network validating the utility of the framework. However, due to its computational complexity, computationally less intensive heuristics have been developed to tackle exactly the aforementioned problem.

The heuristics have been applied to both small scale and large scale/production level environments, demonstrating their effectiveness and yielding potentially large gains in terms of reduced number of required Edge data-centers and increased Air Bandwidth. In particular, the results show that the number of Crosshaul Processing Unit (XPU) can be significantly reduced by applying the solution obtained by the heurisitics, compared with a generic Operator deployment. It is also shown that the number of required XPUs increases with the maximum propagation delay and the available capacity of the links. Regarding the Air Bandwidth of all DUs and eNBs, Heuristic 1 achieves a higher level of aggregation under lower maximum propagation delay, determining a lower number of XPUs placed deeper in the network, since the fronthaul flows would share the bandwidth with backhaul flows for longer paths deeper into the network and the total Air Bandwidth would decrease.

Once, the most critical part of the network, the core, has been optimized, the focus moves to the second challenge exposed in Chapter 1, the service placement on cloud/edge environments. In particular, Chapter 3 has analyzed in scenarios with mobile actors the problem of Virtual Network Function (VNF) placement in a realistic use case based scenario: mobile robotics for warehousing solution in the Valencia city haven, where mobile compute nodes act as an extension of the cloud and edge computing infrastructure. Thus, the problem has been addressed by designing a system model and a mathematical formulation of the problem that considers strict delay bounds and reliability constraints, while taking into account radio coverage, mobility and battery conditions. The proposed solution includes the development of an efficient heuristic requiring less computational complexity. The heuristic has been extensively evaluated via simulations showing that it outperforms a state of the art mobility-aware algorithm. It achieves close to optimal deployments in terms of cost, staying between 15% and 30% away of the optimal costs, while the state of the art algorithm increases the cost gap with respect to our solution as the number of Network Function (NF) bound to mobile nodes grows. The proposed solution of this work was the first to tackle the VNF placement problem simultaneously respecting battery, coverage and delay constraints over a mobile and volatile 5G infrastructure, offering an efficient solution for providers dealing with these type of scenarios.

Finally, the optimization has focused on the part of the network closer to the end users and aplication environments. In Chapter 4 it has been addressed the problem of defining an optimal level of cooperation among network nodes performing a supervised learning task. In this chapter, a system model has been defined accounting for the presence of both learning nodes (with capability of perform ML models and learn from their result) and information nodes (providing the data to the learning nodes in order to train the models and extract the learning) interacting with each other. Then we formulated the problem of choosing which learning nodes should cooperate to complete the learning task, and the

information nodes that should provide them with data, as well as the number of iterations to perform. Although being NP hard, some important properties of our problem, most notably its submodularity, allowed to define a solution algorithm that has cubic *worst-case* time complexity and is $1 + 1/|\mathcal{I}|$-competitive, with $\mathcal{I}$ being the set of information nodes. Numerical results also has shown that our approach closely matches the optimum and outperforms state-of-the-art solutions, for both classification and regression tasks, the most important tasks of supervised learning.

Appendices

A Linearization of the product of two variables

A.1. Linearization of the product of two binary variables

Property 5 (Linearization of the product of two binary variables)**.** *Let $z = x \cdot y$, where z, x and y are binary, then the product can be linearized as follows:*

$$\begin{aligned} z &\leq x, \\ z &\leq y, \\ z &\geq x + y - 1 \end{aligned}$$

A.2. Linearization of the product of one binary variable and one real bounded variable

Property 6 (Linearization of the product of one binary variable and one real bounded variable)**.** *Let $z = x \cdot y$, where x is binary and $y, z \in \mathbb{R}$, $0 \leq y \leq B$, then the product can be linearized as follows:*

$$\begin{aligned} z &\leq B \cdot x, \\ z &\leq y, \\ z &\geq y - B \cdot (1 - x) \\ z &\geq 0 \end{aligned}$$

B NP-Hardness

B.1. NP-Completeness proof of the problem defined in Chapter 2

This section proves the NP-completeness of the problem exposed in Section 2.2.

A **multi-commodity flow problem** involves a collection of several networks whose flows must independently satisfy conservation of flow constraints, but are coupled through some other constraints or the cost function. Consider a directed graph $(\mathcal{N}, \mathcal{A})$, and a finite collection of flow vectors $x(m), m = 1, ..., M$, on that graph, where M is a given integer. Let $x(m)$ denote the flow vector of commodity m, and let $x = (x(1), ..., x(M))$ denote the collection of all commodity flow vectors. Each flow vector $x(m)$ must satisfy its own conservation of flow constraints $\forall i \in N,\ m = 1, ..., M$,

$$\sum_{\{j|(i,j)\in A\}} x_{ij}(m) - \sum_{\{j|(j,i)\in A\}} x_{ji}(m) = s_i(m) \tag{B.1}$$

where $s_i(m)$ are given supply scalars. Furthermore, the commodity flows must together satisfy $x = (x(1), ..., x(M)) \in X$, where X is a constraint set, which may impose additional restrictions on the various commodities. For example, to force a commodity m to avoid some arc (i, j), the constraint $x_{ij}(m) = 0$ may be introduced. In this way, one can model situations where each commodity is restricted to use only a subgraph of the given graph.

The feasible set is

$$F = \{x \in X | x\ satisfies\ Equation\ (B.1)\},$$

and the cost function is of the form $f(x) = f(x(1), ..., x(M))$.

The general *convex multi-commodity flow problem* is defined as

$$\begin{aligned} &minimize \quad f(x) \\ &subject\ to \quad x \in F \end{aligned}$$

where it is assumed that F is convex and f is convex over F.

Note that x may be viewed as a flow vector in an expanded graph consisting of M (disconnected) copies of the original graph $(\mathcal{N}, \mathcal{A})$. With this interpretation, it is seen that the only coupling between the commodities comes through the cost function and the constraint $x \in X$.

The version of the multi-commodity problem that is most amenable to analysis and algorithmic solution is the convex separable multi-commodity flow problem. In this problem the set X has the form

$$X = \{x | x_{ij}(m) \in X_{ij}(m), \forall (i,j) \in A, m = 1, ..., M\} \tag{B.2}$$

where $X_{ij}(m)$ are intervals of the real line and the cost function has the form

$$f(x) = \sum_{(i,j) \in A} f_{ij}(y_{ij}) \tag{B.3}$$

where y_{ij} is the total flow of arc (i,j), $y_{ij} = \sum_{m=1}^{M} x_{ij}(m)$ and each $f_{ij} : \mathbb{R} \to \mathbb{R}$ is a convex function of y_{ij}. Note here that the cost function is not separable with respect to the commodity flows $x_{ij}(m)$, but only with respect to the total flows y_{ij}. There is also a constraint-separable version of the multi-commodity flow problem, where the constraint set X has the form of Equation (B.2) but the cost function f does not have the separable form of Equation (B.3).

In the separable multi-commodity flow problem, commodities are coupled only through the total arc flows y_{ij} that appear in the separable cost function. Another type of commodity coupling in multi-commodity problems arises when the set X includes additional upper bounds on the total flows of the arcs:

$$X = \{x | x_{ij}(m) \in X_{ij}(m), y_{ij} \leq c_{ij}\},$$

for all $(i,j) \in A, m = 1, ..., M$, where $X_{ij}(m)$ are given intervals of the real line, and c_{ij} are given scalars representing arc "capacities". The convex separable version of the resulting problem is referred to as a convex separable multi-commodity flow problem with arc capacities. This problem may also be viewed as a special case of the convex network problem with side constraints, where the side constraints are the capacity constraints $y_{ij} \leq c_{ij}$. The described multi-commodity flow problem is NP-complete when integer binary constraints are imposed on the side constraints.

In the sequel, the optimization problem considered here will be reduced to the multi-commodity flow problem to establish its NP-completeness. To this end, we employ the definition of the problem we know is NP-complete, the definition we have just given from [130], and then we construct our problem from the definition of the multi-commodity flow problem. In fact, the variant of this problem we use is the un-splittable flow problem because our flows have to follow a single path; that is, each flow can leave a node only through one link and cannot be split to follow several links or paths. The definition is the same but adding the constraint that flows cannot be split.

On one hand, we consider three types of flows (a fronthaul flow before reaching a CU (1), after leaving a CU (2) and backhaul flow (3)), and we consider one source that mixes all our sources, one destination that mixes all our destinations and one XPU that mixes all the XPUs, so the collection of all commodity flow vectors will be $x = (x(1), x(2), X(3))$ $(x(1) = x(2))$, then each vector $\forall i \in N,\ m = 1, 2, 3$

$$\sum_{\{j|(i,j)\in A\}} x_{ij}(m) - \sum_{\{j|(j,i)\in A\}} x_{ji}(m) = s_i(m) \tag{B.4}$$

$$s_i(1) = \begin{cases} 0 & if\ i\ is\ an\ intermediate\ node \\ x(1) & if\ i\ is\ a\ source, \\ -x(1) & if\ i\ is\ a\ XPU \end{cases}$$

$$s_i(2) = \begin{cases} 0 & if\ i\ is\ an\ intermediate\ node \\ x(2) & if\ i\ is\ a\ XPU, \\ -x(2) & if\ i\ is\ a\ destination \end{cases}$$

$$s_i(3) = \begin{cases} 0 & if\ i\ is\ an\ intermediate\ node\ or\ XPU \\ x(3) & if\ i\ is\ a\ source, \\ -x(3) & if\ i\ is\ a\ destination \end{cases}$$

In addition, we need to add a constraint in the set of constraints X to prevent the backhaul flows from entering XPUs,

$$\sum_{\{j|(i,j)\in A\}} x_{ij}(m) = 0, \quad \forall i\ a\ XPU,\ m = 3 \tag{B.5}$$

On the other hand, we need to add the constraint for the capacities of the links as in the multi-commodity flow problem with arc capacities. The constraint will be introduced as follows: y_{ij} is the total flow of arc (i, j) $y_{ij} = \sum_{m=1}^{3} x_{ij}(m)$ and the side constraints are the capacity constraints $y_{ij} \leq c_{ij}$, where c_{ij} is the capacity of the link (i, j). Also, the rest of the constraints in our framework will be introduced in the set X as constraints of each type of commodity.

Furthermore, as with the multi-commodity flow problem, the commodity flows must together satisfy $x = (x(1), x(2), x(3)) \in X$, where X is a constraint set, which may impose

additional restrictions on the various commodities beyond those in Equation (B.5).

The feasible set is

$$F = \{x \in X | x\ satisfies\ (B.4)\ and\ the\ capacities\ of\ links\}$$

and the cost function is of the form $f(x) = f(x(1), x(3))$.

$$f(x(m)) = \begin{cases} -1 & if\ the\ source\ is\ a\ DU \\ 1 & if\ the\ node\ is\ an\ XPU \\ 0 & otherwise \end{cases}$$

And the general *multi-commodity flow problem* becomes

$$\begin{aligned} minimize \quad & f(x) \\ subject\ to \quad & x \in F \end{aligned}$$

where we assume that F is convex and f is convex over F.

Concluding, since we can reduce the multi-commodity flow problem with side constraints with integer values to our problem and the first one is NP-complete, our problem is also NP-complete.

B.2. NP-Hardness proof of the problem defined in Chapter 4

This section proves that the problem formulated in Section 4.2, i.e. the problem of optimally configuring the system for an ML task, expressed in Equation (4.1) and Equation (4.2), is NP hard.

Proof: The proof can be obtained via a reduction from the knapsack problem [131], a combinatorial optimization problem where a set of N numbered items is given, each of them associated with a weight ω_s and a value ν_s. The goal is to select a subset of items with maximum total value and total weight less or equal to a maximum given capacity, Ω.

Our reduction maps any given instance of the knapsack problem to a simpler, *special-case* instance of our own, as set forth next.

The sets of L-nodes and I-nodes are respectively $\mathcal{L} = \{l_1 \dots l_N\}$ and $\mathcal{I} = \{i_1 \dots i_N\}$, i.e., there are as many L-nodes as there are items in the knapsack problem, and as many I-nodes as there are L-nodes. Further, we connect all L-nodes in a logical full mesh, and impose that each I-node $i_s \in \mathcal{I}$ can only be connected to the corresponding L-node $l_s \in \mathcal{L}$. We also set the number of iterations to an arbitrary number $\hat{K} > 0$, and the number of samples generated by each I-node to an arbitrary number $r > 0$.

Given the above, matrix $\mathbf{P}$ is fixed and the decisions concern only matrix $\mathbf{Q}$, which is now a diagonal matrix with elements $q(i_s, l_s)$, mapping into the x_s variables in the knapsack problem. Specifically, we activate edge (i_s, l_s) in our problem if and only if $x_s = 1$, i.e., $q(i_s, l_s) \leftarrow x_s$. Furthermore, we map edge costs in our problem into item weights in the knapsack problem. In particular, let ν_s correspond to the opposite of the link cost c_{i_s,l_s}, then we have a perfect correspondance between the objective of the knapsack problem and that in Equation (4.1).

Next, we need to map the capacity constraint in the knapsack problem to constraint Equation (4.2). To this end, we first set $T^{\max} \leftarrow \infty$. Then, given that $\mathbf{P}$ is fixed, γ is also known and fixed, and each L-node can only receive data from one I-node only, the amount of data received by L-node i_s in each iteration is r or 0, depending on the value of x_s. A correspondence between the constraint in the knapsack problem and that in our problem is then established by fixing $\epsilon^{\max} \leftarrow \Omega$, and setting β and ξ in the expression of the learning error at a single L-nodes in such a way that:

$$\frac{\epsilon_{l_s}^K(\mathbf{P}, \mathbf{Q})}{\gamma|\mathcal{L}|} = \frac{1 - \beta \log\left(\hat{K} - \xi\right)}{\gamma|\mathcal{L}|} = \omega_s. \tag{B.6}$$

Last, we need the reduction to take (at most) polynomial time. In our case, it is straightforward to see that the mapping takes linear time, namely $O(|\mathcal{L}| + |\mathcal{I}|)$, hence, the condition is fulfilled.

In summary, any instance of an NP-hard problem can be transformed into a special-case instance of our own, which proves the thesis. ■

C Algorithms Pseudo-codes

C.1. Algorithm for fixed RAN elements (Heuristic 2)

Algorithm 7 Heuristic 2 of Chapter 2: Heuristic algorithm for fixed RAN elements

procedure HEURISTICFIXEDRAN
  $DUsNotUsed \leftarrow DUs$
  while $(UsedXPUs < MaxXPUs)\&\&(maxDUit > 1)\&\&(DUsUsed < NumberDUs)$ **do**
    $maxDUit \leftarrow 0$
    for all $r \in XPUPlacement$ **do**
      $maxDUXPU \leftarrow 0$
      for all $l \in DUsNotUsed$ **do**
        $Path1_{f^l} \leftarrow ShortestPath(DU_l, XPU_r)$
        while $(Capacity(link) + f^l > MaxCapacity(link),\ link \in Path1)$ and $(Not\ All\ Links\ Removed)$ **do**
          Remove links that cannot transport f^l
          $Path1_{f^l} \leftarrow ShortestPath(DU_{f^l}, XPU_r)$
        $Path2_{f^l} \leftarrow ShortestPath(XPU_r, Destination)$
        while $(Capacity(link) + f^{l+} > MaxCapacity(link),\ link \in Path2)$ and $(Not\ All\ Links\ Removed)$ **do**
          Remove links that cannot transport f^l
          $Path2_{f^l} \leftarrow ShortestPath(XPU_r, Destination)$
        Recompute delays for flows already routed
        if Recomputed delays satisfy their maximum delay **then**
          Keep the paths and the DUs that are placed for the current XPU
          $maxDUXPU \leftarrow maxDUXPU + 1$
      if $maxDUXPU > maxDUXPUsaved$ **then**
        $maxDUXPUsaved \leftarrow maxDUXPU$
        Save the information for all the DUs that uses this XPU
    if $maxDUXPUsaved > 1$ **then**
      $maxDUXPUit \leftarrow maxDUXPUsaved$
      Save the information for all the DUs that uses this XPU
      Update $DUsNotUsed$ removing the ones that uses the selected XPU
  $flag \leftarrow 1$
  while $flag == 1$ **do**
    $flag \leftarrow 0$
    for all $l \in eNBs$ **do**
      for all $k \in BackhaulFlowsOfeNB(l)$ **do**
        $Path_{b_k^l} \leftarrow ShortestPath(source_l, Destination)$
        while $(Capacity(link) + b_k^l > MaxCapacity(link),\ link \in Path)$ and $(Not\ All\ Links\ Removed)$ **do**
          Remove links that cannot transport b_k^l
          $Path_{b_k^l} \leftarrow ShortestPath(source_l, Destination)$
        Recompute delays for flows already routed
        if Recomputed delays satisfy their maximum delay **then**
          Keep the path of the new backhaul flow and update the loads in the links
        else
          $flag \leftarrow 1$
          Remove one DU from the XPU that accommodates more DUs
          if The XPU selected contains only 2 DUs **then**
            Remove the two DUs
          Accommodate to other XPU
          Update all the information saved for those DUs
          Remove the information of the backhaul flows placed
          Look for another XPU to accommodate the DU

References

[1] N. Molner, A. de la Oliva, I. Stavrakakis, and A. Azcorra, "Optimization of an integrated fronthaul/backhaul network under path and delay constraints," *Ad Hoc Networks*, vol. 83, pp. 41–54, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870518306206

[2] B. Nemeth, N. Molner, J. M. Pérez, C. J. Bernardos, A. D. la Oliva, and B. Sonkoly, "Delay and reliability-constrained vnf placement on mobile and volatile 5g infrastructure," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[3] F. Malandrino, C. F. Chiasserini, N. Molner, and A. De La Oliva, "Network support for high-performance distributed machine learning," *arXiv preprint arXiv:2102.03394*, 2021.

[4] N. Molner, S. González, T. Deiß, and A. de la Oliva, "The 5g-crosshaul packet forwarding element pipeline: Measurements and analysis," in *2017 Fifth International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN)*, 2017, pp. 1–6.

[5] A. Tzanakaki, M. Anastasopoulos, N. Gomes, P. Assimakopoulos, J. M. Fàbrega, M. S. Moreolo, L. Nadal, J. Gutiérrez, V. Sark, E. Grass, D. Camps-Mur, A. de la Oliva, N. Molner, X. C. Perez, J. Mangues, A.Yaver, P. Flegkas, N. Makris, T. Korakis, and D. Simeonidou, *Transport network architecture*. John Wiley & Sons, 2018.

[6] C. Casetti, C. F. Chiasserini, T. Deiß, P. A. Frangoudis, A. Ksentini, G. Landi, X. Li, N. Molner, and J. Mangues, "Network slices for vertical industries," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 254–259.

[7] C. Casetti, C. F. Chiasserini, J. M. Pérez, N. Molner, T. Deiß, J. E. B. González, G. L. C.T. Phan, F. Messaoudi, N. Serrano, J. Mangues, and C. Turyagyenda, "The vertical slicer: Verticals' entry point to 5g networks," in *The 27th European Conference on Networks and Communications (EuCNC 2018)*, 2018.

[8] C. Casetti, C. F. Chiasserini, N. Molner, J. M. Pérez, T. Deiß, C. Phan, F. Messaoudi, G. Landi, and J. B. Baranzano, "Arbitration among vertical services," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 153–157.

[9] K. Antevski, J. M. Pérez, N. Molner, C. F. Chiasserini, F. Malandrino, P. Frangoudis, A. Ksentini, X. Li, J. S. Lozano, R. Martínez, I. Pascual, J. Mangues-Bafalluy, J. Baranda, B. Martini, and M. Gharbaoui, "Resource orchestration of 5g transport networks for vertical industries," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 158–163.

[10] 5G-Crosshaul, "D1.2: Final 5G-Crosshaul system design and economic analysis," December 2017.

[11] Cisco Visual Networking Index, "Cisco annual internet report 2018-2023," *Cisco white paper*, 2020.

[12] ——, "2020 global networking trends report," *Cisco white paper*, 2020.

[13] ——, "Global mobile data traffic forecast update, 2016-2021," *Cisco white paper*, 2017.

[14] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[15] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[16] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[17] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*. Ieee, 2010, pp. 27–33.

[18] W. Shi, 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] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[20] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[21] Mobile, China, "C-RAN: the road towards green RAN," *White Paper, ver*, vol. 2, 2011.

[22] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for mobile networks - a technology overview," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, First Quarter 2015.

[23] B. Guo, W. Cao, A. Tao, and D. Samardzija, "Cpri compression transport for lte and lte-a signal in c-ran," in *7th International Conference on Communications and Networking in China*. IEEE, 2012, pp. 843–849.

[24] A. de la Oliva and J. A. Hernandez and D. Larrabeiti and A. Azcorra, "An overview of the CPRI specification and its application to C-RAN-based LTE scenarios," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 152–159, February 2016.

[25] C. P. R. I. eCPRI Interface Specification, "eCPRI specification v1.0," August 2017.

[26] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 105–111, Oct 2015.

[27] 3GPP RAN3, "TR 38.801 V14.0.0," Mar 2017, Available at: http://www.3gpp.org/ftp/Specs/archive/38_series/38.801/.

[28] X. Costa-Perez, A. Garcia-Saavedra, X. Li, T. Deiss, A. de la Oliva, A. di Giglio, P. Iovanna, and A. Moored, "5g-crosshaul: An sdn/nfv integrated fronthaul/backhaul transport network architecture," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 38–45, February 2017.

[29] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. D. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, Jun 2017.

[30] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2282–2308, Third Quarter 2016.

[31] K. Sundaresan, M. Y. Arslan, S. Singh, S. Rangarajan, and S. V. Krishnamurthy, "FluidNet: A flexible cloud-based radio access network for small cells," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 915–928, April 2016.

[32] A. Checko, A. P. Avramova, M. S. Berger, and H. L. Christiansen, "Evaluating C-RAN fronthaul functional splits in terms of network level energy and cost savings," *Journal of Communications and Networks*, vol. 18, no. 2, pp. 162–172, Apr 2016.

[33] D. S. Reeves and H. F. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing," *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, vol. 1, pp. 84–91, April 1997.

[34] A. Orda and R. Rom, "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length," August 1989.

[35] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-time Channels," June 1994.

[36] M. R. Kabat, M. K. Patel, and C. R. Tripathy, "An Efficient Algorithm for Delay Delay-variation Bounded Least Cost Multicast Routing," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 3, no. 3, 2009.

[37] J. M. Smith and T. van Woensel, "Topological network design of general, finite, multi-server queueing networks," *European Journal of Operational Research*, March 2009.

[38] M. Garetto and D. Towsley, "Modeling, Simulation and Measurements of Queuing Delay under Long-tail Internet Traffic," *ACM 2003*, June 2003.

[39] M. Olvera-Cravioto, J. Blanchet, and P. Glynn, "On the Transition from Heavy Traffic to Heavy Tails for the M/G/1 Queue: The regularly varying case," *The Annals of Applied Probability*, vol. 21, no. 2, pp. 645–668, 2011.

[40] J. M. Smith, "M/G/c/K blocking probability models and system performance," *Performance Evaluation*, vol. 52, pp. 237–267, 2003.

[41] J. MacGregor Smith, "Properties and performance modelling of finite buffer M/G/1/K networks," *Computers & Operations Research*, vol. 38, pp. 740–754, 2011.

[42] A. Gowda, J. A. Hernández, D. Larrabeiti, and L. Kazovsky, "Delay analysis of mixed fronthaul and backhaul traffic under strict priority queueing discipline in a 5g packet transport network," *Transactions on Emerging*

Telecommunications Technologies, vol. 28, no. 6, 6 2017. [Online]. Available: http:https://doi.org/10.1002/ett.3168

[43] S. Agarwal and F. Malandrino and C. F. Chiasserini and S. De, "Joint VNF Placement and CPU Allocation in 5G," *IEEE INFOCOM 2018*, April 2018.

[44] A. Chakrabartiy and C. Chekuriz and A. Guptax and A. Kumar, "Approximation Algorithms for the Unsplittable Flow Problem," September 2005.

[45] H. Cho and A. Wein, "Unsplittable Flows," in *Final Project*. MIT.

[46] A. Karandikar, "Approximation Algorithms for Stochastic Unsplittable Flow Problems," in *Master Thesis*. School of Computer Science Computer Science Department Carnegie Mellon University Pittsburgh, PA, December 2015.

[47] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[48] T. Wan and P. Ashwood-Smith, "A performance study of cpri over ethernet with ieee 802.1 qbu and 802.1 qbv enhancements," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[49] 3GPP RAN3, "Small cell virtualization functional splits and use cases version SCF159.07.02 Release 7," January 2016, Available at: http://www.3gpp.org/ftp/Specs/archive/38_series/38.801/ (Accessed 29 May 2018).

[50] D. P. Bertsekas and R. Gallager, in *Data Networks*, 1987, p. 186.

[51] N. Bram, K. Mario, V. Sofie, C. Didier, and P. Mario, "How can a mobile service provider reduce costs with software-defined networking?" *International Journal of Network Management*, vol. 26, no. 1, pp. 56–72. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.1919

[52] Ruiquan Jing and Jianjun Tang and Luis Miguel Contreras Murillo and Rui Tang and Qiuyou Wu and Jean-Michel Caia and Yuanbin Zhang, "Consideration on 5G transport network reference architecture and bandwidth requirements," January 2018.

[53] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in lte cellular networks: Key design issues and a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 678–700, 2013.

[54] 3GPP, "Study on Scenarios and Requirements for Next Generation Access Technologies," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.913, 08 2017, version 14.3.0.

[55] 5GACIA, "5G Non-Public Networks for Industrial Scenarios," 5G Alliance for Connected Industries and Automation, White Paper, 7 2019.

[56] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, May 2018, pp. 1–9.

[57] H. Cambazard, D. Mehta, B. O'Sullivan, and H. Simonis, "Bin Packing with Linear Usage Costs - An Application to Energy Management in Data Centres," *Principles and Practice of Constraint Programming - 19th International Conference*, p. ?, 2013, best Application Paper Award. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00858159

[58] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 731–741.

[59] D. K. Friesen and M. A. Langston, "Variable sized bin packing," *SIAM journal on computing*, vol. 15, no. 1, pp. 222–230, 1986.

[60] M. M. Baldi, D. Manerba, G. Perboli, and R. Tadei, "A Generalized Bin Packing Problem for parcel delivery in last-mile logistics," *European Journal of Operational Research*, vol. 274, no. 3, pp. 990–999, 2019. [Online]. Available: https://ideas.repec.org/a/eee/ejores/v274y2019i3p990-999.html

[61] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo, *Bin Packing Approximation Algorithms: Survey and Classification*. New York, NY: Springer New York, 2013, pp. 455–531. [Online]. Available: https://doi.org/10.1007/978-1-4419-7997-1_35

[62] S. Kekki, W. Featherstone, Y. Fang, P. Kuure *et al.*, "MEC in 5G networks," European Telecommunications Standards Institute (ETSI), White paper 28, 6 2018.

[63] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, 2017.

[64] 3GPP, "Service requirements for the 5G system; Stage 1," 3rd Generation Partnership Project (3GPP), Technical Specification (TS), 06 2018, version 15.5.0.

[65] Y.-T. Chen and W. Liao, "Mobility-aware service function chaining in 5g wireless networks with mobile edge computing," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[66] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[67] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware vnf chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 267–276.

[68] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[69] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2449–2457.

[70] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*. IEEE, 2018, pp. 693–701.

[71] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 2016, pp. 1–4.

[72] A. Santoyo-González 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.

[73] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, 2018.

[74] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power iot edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 7–12.

[75] 3GPP, "Study on Communication for Automation in Vertical Domains," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.804, 07 2020, version 16.3.0.

[76] A. Mourad, P.-H. Kuo, D. Rapone, L. Cominardi *et al.*, "5g-coral initial system design, use cases, and requirements," Tech. Rep. deliverable D1.1, 2018.

[77] B. Nogales, V. Sanchez-Aguero, I. Vidal, and F. Valera, "Adaptable and automated small uav deployments via virtualization," *Sensors*, vol. 18, no. 12, p. 4116, 2018.

[78] ITU-T, "Consideration on 5G transport network reference architecture and bandwidth requirements," International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), Study Group 15 Contribution 0462, 2 2018.

[79] L. Cominardi, L. M. Contreras, C. J. Bcrnardos, and I. Berberana, "Understanding QoS Applicability in 5G Transport Networks," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2018, pp. 1–5.

[80] V. Suryaprakash, J. Møller, and G. Fettweis, "On the modeling and analysis of heterogeneous radio access networks using a poisson cluster process," *IEEE Transactions on Wireless Communications*, vol. 14, no. 2, pp. 1035–1047, 2015.

[81] V. Sanchez-Aguero, F. Valera, I. Vidal, C. Tipantuña, and X. Hesselbach, "Energy-Aware Management in Multi-UAV Deployments: Modelling and Strategies," *Sensors*, vol. 20, no. 10, p. 2791, 2020.

[82] M. Suter, R. Eidenbenz, Y.-A. Pignolet, and A. Singla, "Fog application allocation for automation systems," in *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE, 2019, pp. 97–106.

[83] 3GPP, "Overview of 3GPP Release 8," 3rd Generation Partnership Project (3GPP), Technical Specification (TS), 09 2016.

[84] R. Stuhlfauth, "UMTS Long Term Evolution(LTE)," 11 2012.

[85] V. Nikolikj and T. Janevski, "A cost modeling of high-capacity lte-advanced and ieee 802.11 ac based heterogeneous networks, deployed in the 700 mhz, 2.6 ghz and 5 ghz bands," *Procedia Computer Science*, vol. 40, pp. 49–56, 2014.

[86] N. Patriciello, S. Lagen, L. Giupponi, and B. Bojovic, "5g new radio numerologies and their impact on the end-to-end latency," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6.

[87] B. Halvarsson, A. Simonsson, A. Elgcrona, R. Chana, P. Machado, and H. Asplund, "5g nr testbed 3.5 ghz coverage results," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, June 2018, pp. 1–5.

[88] T. Toukabri, G. R. Barbara Martini, C. J. Bernardos, X. Li *et al.*, "5G-TRANSFORMER final system design and Techno-Economic Analysis," Tech. Rep. deliverable D1.4, 2019.

[89] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, and A. Mourad, "5gen: A tool to generate 5g infrastructure graphs," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2019.

[90] R. Fourer, D. M. Gay, and B. W. Kernighan, "Ampl. a modeling language for mathematical programming," 1993.

[91] I. Gurobi Optimization, "Gurobi optimizer reference manual," *URL http://www. gurobi. com*, 2015.

[92] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *USENIX OSDI*, 2014.

[93] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," *arXiv preprint arXiv:1803.07250*, 2018.

[94] H. Y. Ong, K. Chavez, and A. Hong, "Distributed deep q-learning," *CoRR*, 2015.

[95] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, 2018.

[96] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, 2019.

[97] H. H. Zhuo, W. Feng, Y. Lin, Q. Xu, and Q. Yang, "Federated deep reinforcement learning," *arXiv preprint arXiv:1901.08277*, 2019.

[98] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[99] ETSI, "Zero touch network & Service Management (ZSM)," https://www.etsi.org/committee/zsm, online; accessed July 2020.

[100] ——, "Experiential Networked Intelligence (ENI)," https://www.etsi.org/committee-activity/eni, online; accessed July 2020.

[101] Operator Defined Next Generation RAN Architecture and Interfaces, "O-RAN Working Group 2: AI/ML workflow description and requirements," Tech. Rep. O-RAN.WG2.AIML-v01.01, online; accessed July 2020.

[102] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Towards self-learning edge intelligence in 6g," *IEEE Communications Magazine*, 2020.

[103] ETSI, "MEC Working Item 36, MEC in resource constrained terminals, fixed or mobile," https://portal.etsi.org/webapp/WorkProgram/, online; accessed July 2020.

[104] A. Kadav and E. Kruus, "Asap: asynchronous approximate data-parallel computation," *arXiv preprint arXiv:1612.08608*, 2016.

[105] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *IEEE IPDPSW*, 2018.

[106] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *IEEE INFOCOM*, 2019.

[107] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[108] A. A. Abdellatif, C. F. Chiasserini, and F. Malandrino, "Active learning-based classification in automated connected vehicles," in *IEEE INFOCOM PERSIST-IoT Workshop*, 2020.

[109] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless iot intrusion detection," *IEEE Wireless Communications*, 2018.

[110] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, "Communication-efficient distributed reinforcement learning," *arXiv preprint arXiv:1812.03239*, 2018.

[111] Y. Li, I.-J. Liu, Y. Yuan, D. Chen, A. Schwing, and J. Huang, "Accelerating distributed reinforcement learning with in-switch computing," in *ISCA*, 2019.

[112] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[113] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *International conference on machine learning*, 2013.

[114] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[115] OASIS Standard, "MQTT Version 5.0, Mar. 2019," https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, online; accessed July 2020.

[116] "zenoh: Zero Overhead Pub/sub, Store/Query and Compute," http://zenoh.io, online; accessed July 2020.

[117] 3GPP, "TS23.501, System architecture for the 5G System (5GS), Rel. 15," https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144, online; accessed July 2020.

[118] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, 2019.

[119] N. J. Nagelkerke *et al.*, "A note on a general definition of the coefficient of determination," *Biometrika*, 1991.

[120] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, 2012.

[121] ITU, "AI/ML in 5G Challenge 2020," https://www.itu.int/en/ITU-T/AI/challenge/2020/, online; accessed November 2020.

[122] C. Perlich, F. Provost, and J. S. Simonoff, "Tree induction vs. logistic regression: A learning-curve analysis," *Journal of Machine Learning Research*, 2003.

[123] D. Bolton, "The multinomial theorem," *The Mathematical Gazette*, pp. 336–342, 1968.

[124] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming The State of the Art*. Springer, 1983.

[125] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete applied mathematics*, 1984.

[126] R. K. Iyer and J. A. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Advances in Neural Information Processing Systems*, 2013.

[127] L. Valerio, M. Conti, and A. Passarella, "Energy efficient distributed analytics at the edge of the network for iot environments," *Elsevier Pervasive and Mobile Computing*, 2018.

[128] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, 2018.

[129] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, 2018.

[130] D. P. Bertsekas, in *Network Optimization: Continuous and Discrete models*, 1998, p. 349.

[131] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey," in *Combinatorial optimization—eureka, you shrink!* Springer, 2003.