TECHNISCHE UNIVERSITÄT
ILMENAU

Benefits and Limits of Machine Learning for the Implicit Coordination of SON Functions

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt in der Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von Herrn M.Sc. Diego Fernando Preciado Rojas

Datum der Einreichung: 04. 07. 2022

Datum der Verteidigung: 14. 11. 2022

Gutachter:
1. Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel, Technische Universität Ilmenau
2. Prof. Dr.-Ing. Aydin Sezgin, Ruhr-Universität Bochum
3. Prof. Dr.-Ing. Jens Mückenheim, Hochschule Merseburg

doi: 10.22032/dbt.54710
urn: urn:nbn:de:gbv:ilm1-2022000419

Abstract

Due to the introduction of new network functionalities in next-generation mobile networks, e.g., slicing or multi-antenna systems, as well as the coexistence of multiple radio access technologies, the optimization tasks become extremely complex, increasing the OPEX (OPerational EXpenditures). In order to provide services to the users with competitive Quality of Service (QoS) while keeping low operational costs, the Self-Organizing Network (SON) concept was introduced by the standardization bodies to add an automation layer to the network management. Thus, multiple SON functions (SFs) were proposed to optimize a specific network domain, like coverage or capacity.

The conventional design of SFs conceived each function as a closed-loop controller optimizing a local objective by tuning specific network parameters. However, the relationship among multiple SFs was neglected to some extent. Therefore, many conflicting scenarios appear when multiple SFs are instantiated in a mobile network. Having conflicting functions in the networks deteriorates the users' QoS and affects the signaling resources in the network. Thus, it is expected to have a coordination layer (which could also be an entity in the network), conciliating the conflicts between SFs. Nevertheless, due to interleaved linkage among those functions, it is complex to model their interactions and dependencies in a closed form. Thus, machine learning is proposed to drive a joint optimization of a global Key Performance Indicator (KPI), hiding the intricate relationships between functions. We call this approach: implicit coordination.

In the first part of this thesis, we propose a centralized, fully-implicit coordination approach based on machine learning (ML), and apply it to the coordination of two well-established SFs: Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB). We find that this approach can be applied as long as the coordination problem is decomposed into three functional planes: controllable, environmental, and utility planes. However, the fully-implicit coordination comes at a high cost: it requires a large amount of data to train the ML models. To improve the data efficiency of our approach (i.e., achieving good model performance with less training data), we propose a hybrid approach, which mixes ML with closed-form models. With the hybrid approach, we study the conflict between MLB and Coverage and Capacity Optimization (CCO) functions. Then, we apply it to the coordination among MLB, Inter-Cell Interference Coordination (ICIC), and Energy Savings (ES) functions. With the hybrid approach, we find in one shot, part of the parameter set in an optimal manner,

which makes it suitable for dynamic scenarios in which fast response is expected from a centralized coordinator.

Finally, we present a manner to formally include MRO in the hybrid approach and show how the framework can be extended to cover challenging network scenarios like Ultra-Reliable Low Latency Communications (URLLC).

Kurzfassung

Bedingt durch die Einführung neuer Netzfunktionen in den Mobilfunknetzen der nächsten Generation, z. B. Slicing oder Mehrantennensysteme, sowie durch die Koexistenz mehrerer Funkzugangstechnologien, werden die Optimierungsaufgaben äußerst komplex und erhöhen die OPEX (OPerational EXpenditures). Um den Nutzern Dienste mit wettbewerbsfähiger Dienstgüte (QoS) zu bieten und gleichzeitig die Betriebskosten niedrig zu halten, wurde von den Standardisierungsgremien das Konzept des selbstorganisierenden Netzes (SON) eingeführt, um das Netzmanagement um eine Automatisierungsebene zu erweitern. Es wurden dafür mehrere SON-Funktionen (SFs) vorgeschlagen, um einen bestimmten Netzbereich, wie Abdeckung oder Kapazität, zu optimieren. Bei dem konventionellen Entwurf der SFs wurde jede Funktion als Regler mit geschlossenem Regelkreis konzipiert, der ein lokales Ziel durch die Einstellung bestimmter Netzwerkparameter optimiert. Die Beziehung zwischen mehreren SFs wurde dabei jedoch bis zu einem gewissen Grad vernachlässigt. Daher treten viele widersprüchliche Szenarien auf, wenn mehrere SFs in einem mobilen Netzwerk instanziiert werden. Solche widersprüchliche Funktionen in den Netzen verschlechtern die QoS der Benutzer und beeinträchtigen die Signalisierungsressourcen im Netz. Es wird daher erwartet, dass eine existierende Koordinierungsschicht (die auch eine Entität im Netz sein könnte) die Konflikte zwischen SFs lösen kann. Da diese Funktionen jedoch eng miteinander verknüpft sind, ist es schwierig, ihre Interaktionen und Abhängigkeiten in einer abgeschlossenen Form zu modellieren. Daher wird maschinelles Lernen vorgeschlagen, um eine gemeinsame Optimierung eines globalen Leistungsindikators (Key Performance Indicator, KPI) so voranzubringen, dass die komplizierten Beziehungen zwischen den Funktionen verborgen bleiben. Wir nennen diesen Ansatz: implizite Koordination. Im ersten Teil dieser Arbeit schlagen wir eine zentralisierte, implizite und auf maschinellem Lernen basierende Koordination vor und wenden sie auf die Koordination zweier etablierter SFs an: Mobility Robustness Optimization (MRO) und Mobility Load Balancing (MLB). Anschließend gestalten wir die Lösung dateneffizienter (d. h. wir erreichen die gleiche Modellleistung mit weniger Trainingsdaten), indem wir eine geschlossene Modellierung einbetten, um einen Teil des optimalen Parametersatzes zu finden. Wir nennen dies einen "hybriden Ansatz". Mit dem hybriden Ansatz untersuchen wir den Konflikt zwischen MLB und Coverage and Capacity Optimization (CCO) Funktionen. Dann wenden wir ihn auf die Koordinierung zwischen MLB, Inter-Cell Interference Coordination (ICIC) und

Energy Savings (ES) Funktionen an. Schließlich stellen wir eine Möglichkeit vor, MRO formal in den hybriden Ansatz einzubeziehen, und zeigen, wie der Rahmen erweitert werden kann, um anspruchsvolle Netzwerkszenarien wie Ultra-Reliable Low Latency Communications (URLLC) abzudecken.

To Patricio and Teresa

"El barro al barro, el polvo al polvo, la tierra a la tierra, nada empieza que no tenga fin, todo lo que empieza nace de lo que se acabó."

J. Saramago

Acknowledgement

As I conclude my PhD work, I take this opportunity to express my gratitude to all the people and institutions that supported me through this quest. This PhD thesis was carried out within the Integrated Communication Systems (ICS) group as part of the Faculty of Computer Science and Automation. It was financed by the German Academic Exchange Service / Deutscher Akademischer Austauschdienst (DAAD) under the DAAD Research Grants for Doctoral Programmes in Germany.

First, I would like to sincerely thank my supervisor Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel for his guidance, support, and encouragement in pursuing challenging research directions. Secondly, I would also like to thank my colleagues in the ICS group for their critical feedbacks and constructive proposals during our research seminars and workshops. I extend my gratitude to our group's administrative team, especially Nicole Sauer and Jürgen Schmidt for their ever friendly assistance with the organizational tasks.

Secondly, I wish to also thank Martin Kasparick, Renato Cavalcante, and Prof. Dr.-Ing. habil. Slawomir Stanczak from the Signal and Information Processing group in the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute for their guidance to strengthen some of the research ideas within this thesis.

I wish to also thank Prof. Dr.-Ing. Jens Mückenheim and Prof. Dr.-Ing. Aydin Sezgin for accepting to review my thesis as well as being members of the examination board.

Finally, I would also like to thank Angela for her encouragement along this path. Most importantly, I thank my entire family back in Colombia, for their support throughout this time.

Contents

List of Figures

List of Tables

List of Relevant Acronyms

3GPP	third Generation Partnership Project
5GMF	5G Mobile communications promotion Forum
5G-PPP	5G infrastructure Public-Private Partnership
AAS	Active Antenna Systems
ANN	Artificial Neural Network
ANR	Automatic Neighbour Relation
BMU	Best Matching Unit
CAC	Call Admission Control
CAPEX	CAPital EXpenditure
CCO	Coverage and Capacity Optimization
CIO	Cell Individual Offset
COC	Cell Outage Compensation
COD	Cell Outage Detection
CQI	Channel Quality Indicator
CSS	Concurrent learning with Spatial Separation
C-SON	Centralized SON
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ECB	Edge-to-Centre Boundary
eMBB	enhanced Mobile BroadBand
eNB	evolved Node B
ES	Energy Savings
FCAPS	Fault, Configuration, Accounting, Performance, Security
FDD	Frequency-Division Duplexing
FFNN	Feed Forward Neural Network
FFR	Fractional Frequency Reuse
FPI	Fixed Point Iteration
FPR	False Positive Rate
FR	Frequency Reuse
GBR	Guaranteed Bit Rate

GMAR	Geometric Mean of Available Resources
gNB	next-generation Node B
GSM	Global System for Mobile Communications
HC	Hierarchical Clustering
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
HFR	Hard Frequency Reuse
HO	HandOver
HOAP	HandOver Aggregated Performance
ICI	Inter-Cell Interference
ICIC	Inter-Cell Interference Coordination
ITU-R	International Telecommunication Union - Radio sector
KPI	Key Performance Indicator
LBO	Load Balancing Optimization
LTE	Long Term Evolution
MAE	Mean Absolute Error
MARO	Multi-aspect / multi-domain resource optimization
MDP	Markov Decision Process
MDT	Minimization of Drive Testing
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MLB	Mobility Load Balancing
mMTC	massive Machine Type Communications
MRO	Mobility Robustness Optimization
MST	Minimum Spanning Tree
NF	Network Function
NGMN	Next Generation Mobile Networks
N-PLMN	Non-Public Land Mobile Networks
NSI	Network Slice Instance
NSSI	Network Slice Subnet Instance
NWDAF	Network Data Analytics Function
OFDMA	Orthogonal Frequency-Division Multiple Access
OPEX	OPerational EXpenditure
OSS	Operations Support Systems
PCI	Physical Cell Identity
PDCP	Packet Data Convergence Protocol

PFR	Partial Frequency Reuse
PLMN	Public Land Mobile Network
PNF	Physical Network Function
PP	Ping-Pong
PRB	Physical Resource Blocks
QoS	Quality of Service
QoE	Quality of Experience
QL	Q-Learning
RACH	Random Access CHannel
RAN	Radio Access Network
RAT	Radio Access Technology
RB	Resource Block
RE	Resource Element
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RLF	Radio Link Failure
RMSE	Root-Mean-Square Error
ROC	Receiver Operating Characteristic
RRC	Radio Resource Control
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
SA	Simulated Annealing
SDO	Standards Developing Organization
SF	Self-organized network Functions
SFR	Soft Frequency Reuse
SINR	Signal-to-Interference-plus-Noise Ratio
SON	Self-Organized Networks
SOM	Self-Organizing Maps
TNR	True Negative Rate
TSL	Temporal Separation during Learning
TTI	Transmission Time Interval
TTT	Time-To-Trigger
UA	User Association
UE	User Equipment
UMAP	Uniform Manifold Approximation and Projection
URLLC	Ultra-Reliable Low latency Communication

VNF Virtual Network Function
WSS Within cluster Sum of Squares

1 Introduction

With the advent of 5G, the ability to offer agile on-demand services to the users is mandatory. Therefore lifecycle operations such as initial service deployment, configuration changes, upgrades, scale-out, scale-in, optimization, self-healing, etc., should be fully automated processes. Fortunately, as the evolution of mobile technologies has demanded, more flexible architectures are targeted, aiming at reducing the time-to-market of the services as well as CAPital EXpenditure (CAPEX) and OPerational EXpenditure (OPEX). As shown in Fig. 1.1, the CAPEX is mainly driven by the first two network deployment stages[1] (i.e., planning and initial rollout), whereas the cost attributed to the operation and maintenance (which are expected to be long-term and expensive tasks) are accounted for by the OPEX.

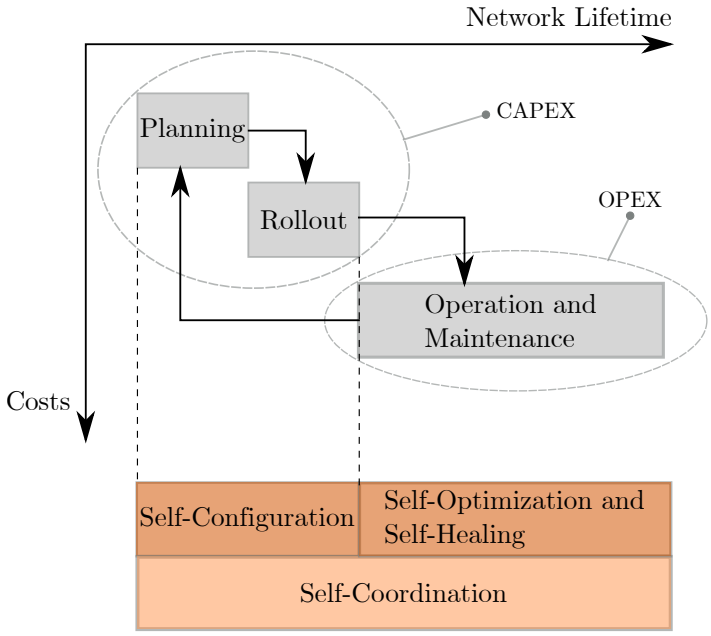


Figure 1.1: Network deployment stages.

To address the expected challenges during the continuous operation and management of multiple Radio Access Technology (RAT)s, the Self-Organized Networks (SON) con-

[1] Which are meant to be continuous but short-duration phases.

cept was initially conceived by the Next Generation Mobile Networks (NGMN) Alliance [1]. From the network operator perspective, SON was intended to ease optimizing the Quality of Service (QoS) of the users while being financially and operationally efficient (i.e., minimizing CAPEX and OPEX). Specifically, three categories of Self-organized network Functions (SF)s have been considered: self-configuration (coping with the optimization challenges of the early stages of the network deployment), self-optimization, and self-healing for the long-lasting optimization tasks, as shown in Fig. 1.1.

Regarding the long-lasting self-optimization branch, multiple SFs were proposed to optimize a specific network dimension (out-of-many). A subset of these functions is depicted in Fig. 1.2, in which three canonical and conflicting optimization dimensions are shown: Quality of Experience (QoE), network coverage, and network capacity.

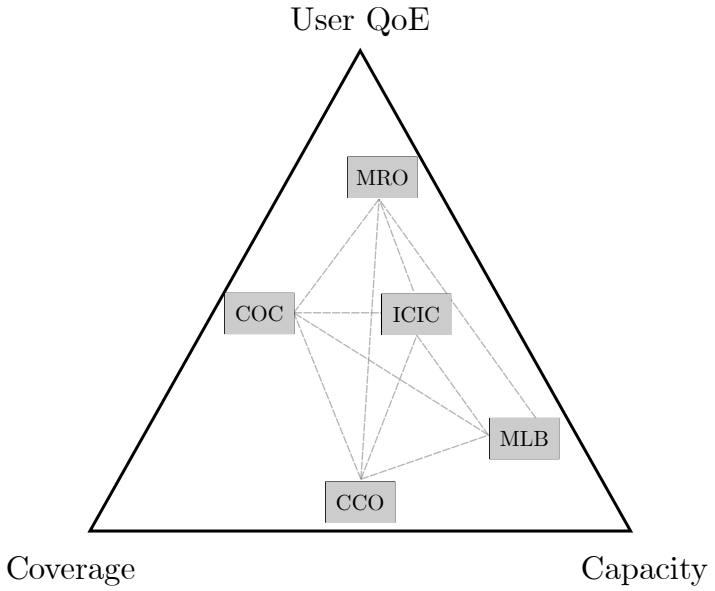


Figure 1.2: Tradeoffs among canonical network optimization dimensions.

Near each corner of the triangle in Fig. 1.2, there is a SF designed to improve that particular aspect of the network. Specifically, Mobility Robustness Optimization (MRO) is intended to reduce the drop calls/sessions due to the mobility of the users (improving the quality), Mobility Load Balancing (MLB) is designed to distribute the load across the whole network (improving the network capacity), Coverage and Capacity Optimization (CCO) is proposed (as its name suggests) to model and optimize the compromise between coverage and capacity, Inter-Cell Interference Coordination (ICIC) is meant to improve the throughput of the users, especially in the cell borders (i.e., quality and network capacity), and finally, Cell Outage Compensation (COC), as the name suggests, recovers the coverage once a cell outage is detected,

minimizing coverage holes and improving the quality of the users.

From an individual perspective, every SF receives information about the state of the network (what we call environmental variables) and adjusts a parameter set (from now on, controllable variables) to optimize a cell-specific Key Performance Indicator (KPI). Unfortunately, there are tight relations among the different variables consumed (or tuned) by the SFs with respect to other SFs. In other words, an isolated analysis and design of the SFs yield suboptimal states in the network and (very likely) instabilities, as will be explained in Section 2.3.

Any mechanism to cope with the conflicting scenarios among multiple SFs is gathered under the standard Self-coordination denomination (see Fig. 1.1), which becomes extremely important and challenging for next-generation networks, as the number of optimization dimensions, SFs, controllable and environmental variables is expected to be remarkably high.

We elaborate on an automatic approach to coordinate multiple SFs boosted by Machine Learning (ML) techniques throughout this document.

1.1 Thesis Objectives and Scope

This thesis deals with the coordinated operation of the canonical SFs depicted in Fig. 1.2 in a centralized way, what is called in the literature a Centralized SON (C-SON) approach [2]. The following research questions are addressed:

1. feasibility of framing a centralized coordination approach in the 5G network architecture,
2. necessary conditions to formulate the centralized SF coordination problem as a fully holistic optimization solution,
3. benefits and limits of having a holistic ML-based coordination framework,
4. viability to improve the scalability and data-efficiency[2] of a holistic coordination scheme.

[2]Defined as the amount of training data needed to attain the best possible parameter set during the network operation.

1.2 Proposed Solutions

The complexity of the coordination problem explodes when heterogeneity in the network (i.e., multiple RAT or several cell types) is introduced, as well as concepts like network slicing or Multiple Input Multiple Output (MIMO) are considered. Additionally, the introduction of new network functionalities and new SFs yields an increase in the dimensionality of the coordination problem, making closed-form modeling unfeasible. Therefore, we propose the joint optimization of SFs employing ML techniques to reduce human intervention.

In the first part of this document, a fully ML-based coordination framework is proposed in which the high dimensional joint optimization problem is reformulated in a low dimensional space where the optimization task reduces to a simple lookup procedure. However, we run into the problem of requiring a vast amount of information for training the involved ML models (due to the curse of dimensionality).

To reduce, to some extent, the amount of information required for the training, a hybrid approach considering both ML stages as well as closed-form modeling is proposed in the second part of this document.

1.3 Publication List

In line with the research directions discussed in Section 1.2, the contributions of the thesis can be categorized into three major groups:

- implicit (ML-based) coordination,
- extensions to the implicit coordination approach, and
- a hybrid coordination approach.

As part of the dissemination process of the main ideas elaborated in this thesis, the following papers were published (a description of the main milestones of each one is also supplied):

1.3.1 Fully Implicit Coordination by ML Methods

- Tanmoy Bag, Sharva Garg, Diego Preciado, Zubair Shaik, Jens Mueckenheim, and Andreas Mitschele-Thiel. Self-organizing network functions for handover optimization in LTE cellular networks. In *Mobile Communication-Technologies and Applications; 24. ITG-Symposium*, pages 1–7. VDE, 2019.

This paper deals with the identification and implementation of the variables involved in the coordination between MLB and MRO in an open-source, system-level simulator.

- Diego Preciado and Andreas Mitschele-Thiel. Machine learning-based SON function conflict resolution. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2019.

  This paper presents the feasibility of finding a ML model capable of predicting in a centralized manner a global KPI using controllable variables (i.e., network parameters) and environmental variables (i.e., beyond the network operator's control) related to the execution of several SON functions. As it turns out, this is the first stage of a fully-implicit coordination framework, which will be explained in detail in Chapter 3. The conflict between MLB and MRO is considered in this study.

- Diego Preciado and Andreas Mitschele-Thiel. A scalable SON coordination framework for 5G. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

  This study presents the feasibility of reformulating a high-dimensional optimization problem in a low-dimensional space (using ML) and solving the new optimization problem in a centralized manner using a simple lookup procedure. By chaining the models of this paper with the one above, we present the complete fully-implicit coordination framework, which will be explained in detail in Chapter 3. The conflict between MLB and MRO is considered in this paper.

- Diego Preciado, Faiaz Nazmetdinov, and Andreas Mitschele-Thiel. Zero-touch coordination framework for self-organizing functions in 5G. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2020.

  End-to-end results for the ML-based, fully-implicit coordination approach are presented in this paper using two ML groupings: Self-Organizing Maps (SOM) with hierarchical clustering and Uniform Manifold Approximation and Projection (UMAP) with Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). A comparison with benchmark solutions based on temporal-spatial execution separation (see Section 2.5) is also supplied. The conflict between MLB and MRO is considered in this paper.

1.3.2 Extensions to Implicit Coordination

- Gerald Budigiri, Diego Preciado, Andreas Mitschele-Thiel, and Stephen Mwanje. Optimal rules mining in SON for distributed intelligence in future cognitive cellular networks. In *2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, pages 1–6. IEEE, 2019.

  Since the centralized and fully-implicit approach removes the human intervention in the network optimization, there is a lack of explainability of the rules and policies generated to optimize the network. Therefore, an ML explainability stage is added to the approach to automatically derive high-level rules in the shape of: **IF** (environmental condition is met) **THEN** (set specific parameter values). The results of this paper are not included in this document. The reader is kindly referred to [7] to find results for the conflict between MLB and MRO.

- Tanmoy Bag, Sharva Garg, Diego Preciado, and Andreas Mitschele-Thiel. Machine learning-based recommender systems to achieve self-coordination between SON functions. *IEEE Transactions on Network and Service Management*, 17(4):2131–2144, 2020.

  Unlike the centralized and fully-implicit approach discussed above, this paper shows a decentralized method for solving, in an implicit way, the conflicts among two different SFs, namely CCO and ICIC (as explained in Section 2.3.2). The ML theory used corresponds to Recommendation Engines (in the content-based and collaborative filtering domains). The results of this paper are not included in this document. The reader is kindly referred to [8].

1.3.3 Hybrid Coordination Approach

- Diego Preciado, Martin Kasparick, Renato L. G. Cavalcante, and Slawomir Stanczak. SON function coordination in campus networks using machine learning. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2022.

  A hybrid approach is presented in this paper to improve the data efficiency of the fully-implicit approach. The main idea is to solve part of the optimization problem in a closed form using Fixed Point Iteration (FPI) and the other part using ML (hereby the hybrid denomination). This paper applies the hybrid approach to a specific campus network scenario, in which the concept of Service Test Point (STP) is introduced as a point in the network where information

about traffic profiles and radio conditions is known. A Feed Forward Neural Network (FFNN) is used to predict the best possible antenna downtilt, whereas FPI is used to find the optimal user association, which is afterward enforced into the network. The specific results for campus networks are not included in this thesis but for a macrocell scenario as detailed in Chapter 4 for the conflict between MLB and CCO.

- Diego Preciado and Andreas Mitschele-Thiel. A data driven coordination between load balancing and interference cancellation. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE, 2022.

  This paper applies the hybrid coordination approach to MLB and ICIC in a macrocell scenario and shows that it can be easily extended to cover another SF called Energy Savings (ES). The details about this study are supplied in Chapter 5.

1.4 Thesis Outline

This section sketches the rest of the content of the thesis and suggests a reading order based on different audience backgrounds.

In Chapter 2, some fundamental concepts regarding mobile networks basics and modeling (from a high-level perspective and mainly related to interference-limited systems) are visited, which will be needed to elaborate on the main ideas throughout this document. The framing of a C-SON approach from the architectural point of view is also proposed. Finally, a deep explanation about SON evolution as well as conflicts among multiple SFs is supplied.

In Chapter 3, a fully implicit coordination by ML methods is proposed. The motivation behind this scheme is that, in C-SON architectures, there is global observability of the states of every cell and every SF, which eases the centralized decision-making process. The applicability of this coordination scheme is studied through the conflict between MRO and MLB functions. We identify the limits that are encountered with this straightforward solution, mainly due to the well-known *curse of dimensionality* .

In Chapter 4, a hybrid approach to overcome the limits encountered with the fully implicit approach is presented. The proposed solution still runs in a closed-form manner, but this time two loops are considered: an outer loop which is entirely based on ML predicting part of the network parameters, and an inner loop, which estimates in

a closed-form manner the rest of the parameters (hence the "hybrid" designation). Offloading part of the parameter estimation to the inner loop, we obtain data efficiency (better predictions with a lower amount of data). The studied SFs in this chapter correspond to MLB and CCO.

In Chapter 5, an extension to the hybrid approach is elaborated, including interference cancellation techniques as well as energy savings. The studied SFs in this chapter are MLB, ICIC, and ES. Multiple SFs have been considered all over this document. The reason behind that is that we wanted to test the robustness of the proposed solution for multiple network optimization objectives. However, there is a common denominator in all the chapters: the inclusion of the MLB function. As it turns out (as it will be explained later on), the MLB-related parameters become the worst offenders in terms of dimensionality. Therefore, we claim MLB is an interesting function to study while designing a data-driven coordination mechanism.

Finally, the main conclusions of this research, as well as some directions for future work, are suggested in Chapter 6.

The details about the main ML models and the heuristics chosen as benchmark schemes can be found in the Appendix section.

The proposed order of reading of this thesis is presented in Fig. 1.3.

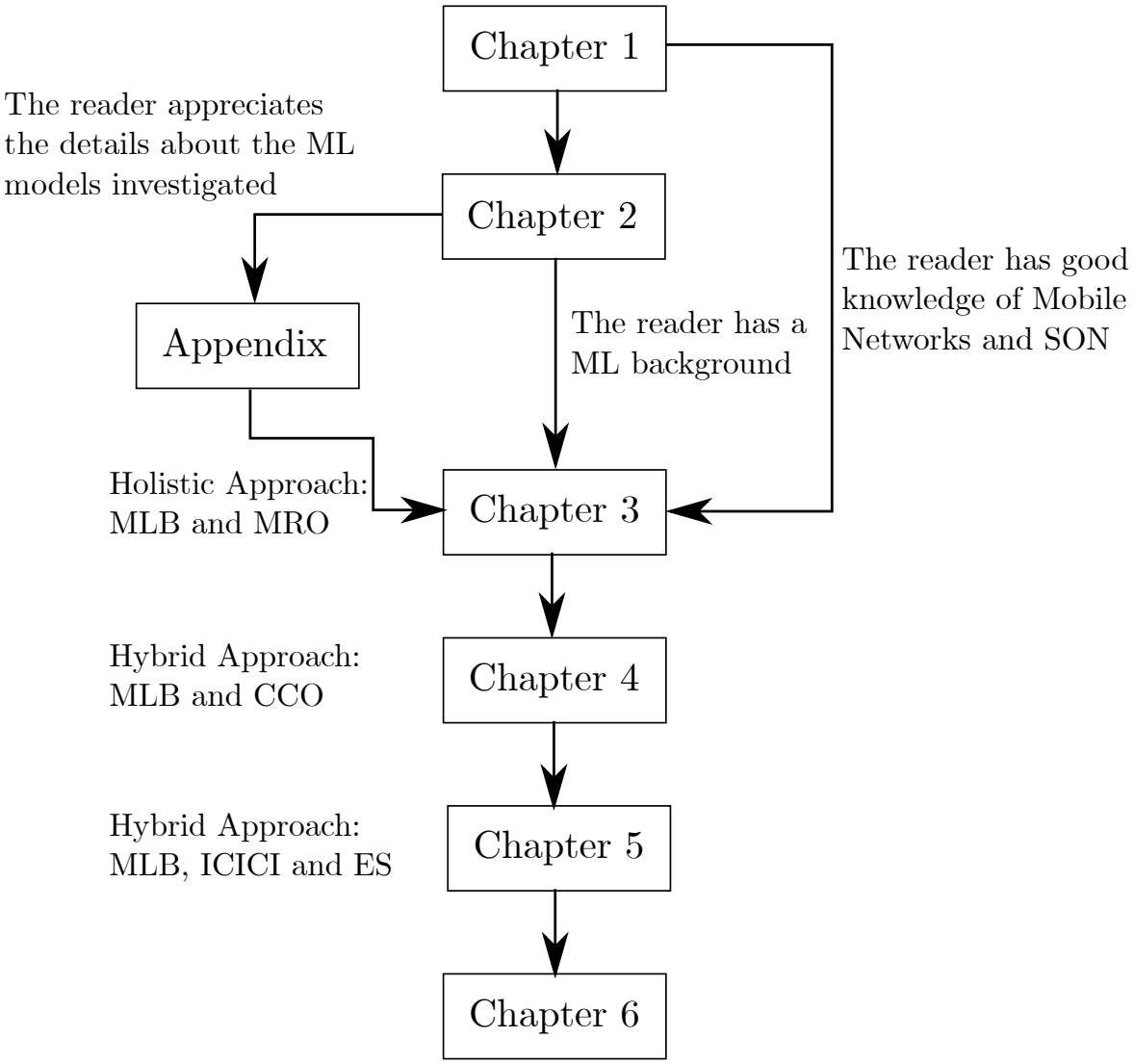


Figure 1.3: Proposed thesis reading order for different audiences.

2 Fundamentals

The design of new mobile network technologies is a dynamic process that started in the 80s and has experienced a continuous evolution through the so-called network generations:

- **1G networks**. This technology was introduced in the early 80s and was characterized by analog services, low network capacity, and very large and expensive User Equipment (UE)s.
- **2G networks**. This generation was adopted in the early 90s. Global System for Mobile Communications (GSM) was the main representative. It was a digital technology that enabled more efficient use of the resource as well as cheaper devices. It was initially designed for voice services through a centralized circuit-switched core domain, and later on, it supported Short Message Service (SMS).
- **2.5G networks**. General Packet Radio Service (GPRS) networks were characterized by the introduction of a packet-switched core domain to carry data traffic which became popular during the GSM roll-out.
- **3G networks**. Universal Mobile Telecommunication System (UMTS) was developed from GSM by completely changing the technology used on the air interface, based on Wideband Code Division Multiple Access (WCDMA), while keeping the core network (both circuit and packet-switched) mainly unchanged. The standard bandwidth allocated to the base stations is 5 MHz for uplink and downlink.
- **3.5G networks**. They took off around 2005. The main enabler was the introduction of High-Speed Packet Access (HSPA) both in uplink as well as in downlink, i.e., High-Speed Uplink Packet Access (HSUPA), and High-Speed Downlink Packet Access (HSDPA) respectively.
- **4G networks**. For this technology, the Evolved Packet Core (EPC) replaced the packet-switched core, whereas there is no equivalent to the circuit-switched

domain, which means that the voice traffic must go over IP (e.g., using an IP Multimedia Subsystem)[1]. Unlike 2G or 3G, LTE provides UEs with always-on connectivity by setting up an IP connection (known as a default bearer) for a device when it switches on and maintaining that connection until it switches off. In the radio part, LTE introduces Orthogonal Frequency-Division Multiple Access (OFDMA) for downlink and Single Carrier Frequency Division Multiple Access (SC-FDMA) for uplink. The allocated bandwidth is more flexible: 1.4, 3, 5, 10, 15, or 20 MHz [11]. From the architectural point of view, LTE is flatter than the previous technologies in the sense that no explicit radio controller is needed, i.e., 3G-Radio Network Controller (RNC) tasks are now executed by an evolved Node B (eNB).

- **4.5G networks**. They are also coined under the "LTE-advanced" denomination, mainly characterized by the introduction of the concept of carrier aggregation (which allows operators to combine two or more LTE carriers to increase the channel capacity) as well as the enhanced use of MIMO antennas[2] and the introduction of low-power base stations to improve the coverage (known as relay nodes).

- **5G networks**. 5G technology represents a breakthrough in the mobile world concerning the way how the network architecture and network management are conceived. The main enablers of 5G technology are:
  - the 5G Radio (including new spectrum options, new antenna structures, an extended physical layer, protocols designs, etc.),
  - the cloud network/edge computing/virtualization concepts (enabling new network architectures, in which decentralization, as well as the use of commodity servers to host specialized network functions, is foreseen),
  - and Artificial Intelligence (leveraging smart network management in what is traditionally called the Automation Plane - See Fig. 2.12).

  These factors are called "major inflection points" in [12].

[1] As a matter of fact, Long Term Evolution (LTE) was designed to be fully IP-based.
[2] MIMO was already used by LTE in Release 8. LTE-advanced enhanced this further.

2.1 System under Study

In this thesis we use the following standard notations: scalars are denoted by lowercase letters (e.g. x and y) whereas boldface letters are intended to denote vectors (e.g. $\boldsymbol{x}$ and $\boldsymbol{y}$). The i-th element of a vector $\boldsymbol{x}$ is denoted by x_i. A vector inequality $\boldsymbol{x} \geq \boldsymbol{y}$ should be understood as a component-wise inequality. Sets are defined with calligraphic fonts (e.g. $\mathcal{X}$ and $\mathcal{Y}$). By $\| \cdot \|_2$, we denote the standard $l2$ norm a.k.a. the Euclidian norm. Sets of nonnegative and positive reals are denoted by $\mathbb{R}_+$ and $\mathbb{R}_{++}$. The set of positive integers is denoted by $\mathbb{Z} \coloneqq \{1, 2, ...\}$. For the sake of unit conformity, we adopt the nonstandard convention of adding the superscript $[\cdot]^{[\mathrm{dBm}]}$ to the quantities that should be considered in dBm rather than linear units (unless the units are explicitly mentioned otherwise).

The relevant notation considered throughout this chapter is provided in Table 2.1.

Table 2.1: List of variables

Description	**Symbol**
Set of cells	$\mathcal{B} = \{1, ..., B\}$
Set of UEs	$\mathcal{S} = \{1, ..., S\}$
i-th cell's transmit power i-th cell	$p_i \in \mathbb{R}_+$
Transmit power vector	$\boldsymbol{p} \in \mathbb{R}_{++}^B$
i-th cell utilization	$\rho_i \in [0, 1]$
Cell utilization vector	$\boldsymbol{\rho} \in [0, 1]^B$
Link budget from i-th cell to m-th UE	$g_{i,m}(\cdot)$
SINR from i-th cell to m-th UE	$\mathrm{SINR}_{i,m}(\cdot)$
Base station's Bandwidth	BW
Noise Power measured over BW	$\sigma^2 \in \mathbb{R}_+$
distance between BS of the i-th cell to m-th UE	$d_{i,m} \in \mathbb{R}_+$
Zero-mean Gaussian distributed random variable	$\chi_\Omega \in \mathbb{R}$
Shannon capacity between i-th cell to m-th UE	$c_{i,m}(\cdot) \in \mathbb{R}_+$
Reduction power factor in the i-th cell center	$\eta_i^c \in [0, 1]$
Reduction power factor at the i-th cell edge	$\eta_i^e \in [0, 1]$
m-th UE's requested data rate	$\gamma_m \in \mathbb{R}_+$
i-th cell's user association	$\mathcal{P}_i$
Set of user associations	$\mathcal{P} \coloneqq \{\mathcal{P}_1, ..., \mathcal{P}_i, ..., \mathcal{P}_B\}$
Average response time in the i-th cell's	$T_i(x) \in \mathbb{R}_{++}$
Waiting time in the i-th cell's queue	$W_i(x) \in \mathbb{R}_+$
Throughput perceived by m-th UE in the i-th cell	$R_{i,m} \in \mathbb{R}_+$
Hysteresis	H
Time to trigger	TTT
Cell Individual Offset between cells i and j	$\mathrm{CIO_{i,j}}$

2.1.1 Interference Limited Systems

Regardless of the mobile network technology, the dynamics in the radio domain are ruled by well-established phenomena and tradeoffs. On the first-hand side, because the mobile networks are designed to serve multiple users, a multiplexing scheme is needed to reduce the access contention to the scarce radio resources, namely: space, frequency (for 2G), time (for 2G), orthogonal codes (for 3G) or orthogonal subcarriers[3] (for 4G and 5G). The multiplexing of several UEs takes place in the network's physical layer and aims at optimizing signal strength and quality while reducing the interference.

Interference is the major limiting factor in mobile networks (in terms of performance and capacity). There are various interference scenarios in the network, and they depend on the data flow direction, i.e., uplink (from UE to the base station) or downlink (from the base station to the UEs). This document focuses on the downlink interference scenarios in a Frequency-Division Duplexing (FDD) system, like the one presented in Fig. 2.1.

In the downlink direction, the sources of interference can be other base stations operating in the same frequency band (at the exact moment) or any noncellular system which inadvertently leaks energy into the frequency band of a specific cell.

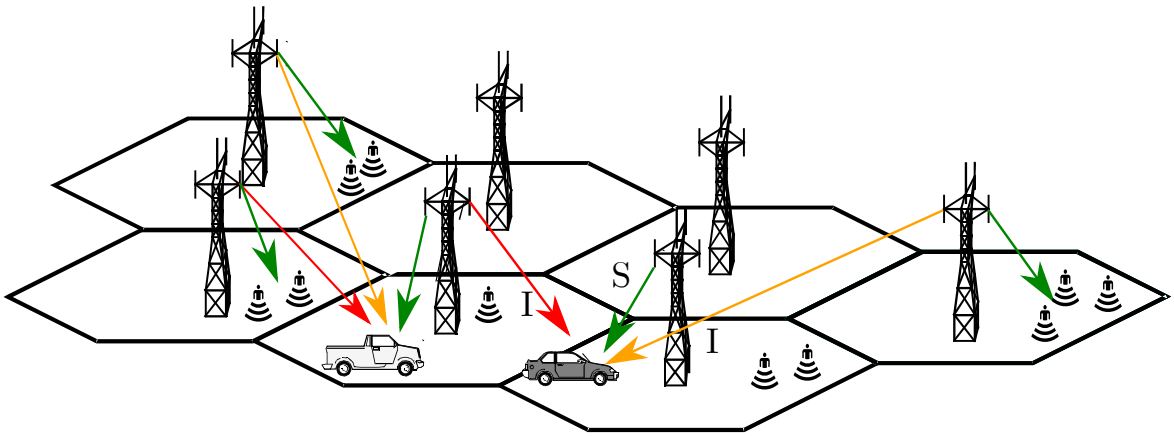


Figure 2.1: Downlink interference scenarios.

In Fig. 2.1, the downlink desired signal is S, whereas the interference from neighbor base stations is I. To allow the receiver at the UE to decode a message successfully, it is expected to have a high Signal-to-Interference-plus-Noise Ratio (SINR) value, defined as:

$$\text{SINR} = \frac{S}{\sum_k I_k + \sigma^2} \tag{2.1}$$

[3] Which corresponds to frequency and time as well.

where σ^2 accounts for the average thermal noise power measured over the frequency bandwidth allocated to the serving base station [13], which is BW = 10 MHz for all the base stations throughout this thesis. The summation in the denominator in Eq. (2.1) groups all the "active" interfering "co-channel" cells, denoted by k. The term "active" refers to the cells that are transmitting in a specific frequency at some time (since they can generate interference to the neighbors, whereas the inactive cells are assumed not to create any interference whatsoever[4]). The activity factor of a cell, i.e., the percentage of used resources, is measured through the cell utilization (ρ_k), as we will see below. In Fig. 2.1, there are two base stations with no users (not a common situation), therefore, they do not impose any interference level on the neighborhood. The term "co-channel cells" groups all the cells transmitting in the same time-frequency resources (which does not necessarily mean the tier-one neighbors if some frequency reuse scheme is used in the network, as explained in Section 2.1.2).

Additionally, it is well-established that the average received signal strength at any point decays as a power law of the distance of separation between a transmitter and receiver. Therefore, in Fig. 2.1, the interference level received by active co-channel cells located far away is lower (orange arrows) than the interference level from close interferers (red arrows).

Let us consider a Public Land Mobile Network (PLMN) with B cells, represented by the set $\mathcal{B} = \{1, ..., B\}$, and S UEs, represented by the set $\mathcal{S} = \{1, ..., S\}$. Let p_i be the transmitting power of serving cell i, and $g_{i,m}(\cdot)$ the link budget from i-th cell to m-th user[5]. According to Eq. (2.1), the perceived SINR for the m-th user being served by i-th cell is given by:

$$\mathrm{SINR}_{i,m}(\boldsymbol{\rho}, \cdot) = \frac{p_i g_{i,m}(\cdot)}{\sum_{k \neq i}^{B} p_k g_{k,m}(\cdot)\rho_k + \sigma^2} \tag{2.2}$$

we claim that $g_{i,m}(\cdot)$ aggregates the impact of three main radio propagation mechanisms: reflection, diffraction, and scattering. There are several large-scale propagation models based on the physics of these three mechanisms in the literature. In this document (especially for the simulation studies), we consider the so-called Log-distance Path Loss Model with Log-normal Shadowing for a carrier frequency of 2 GHz in a macrocell scenario, expressed as in Eq. (2.3) [14].

$$[g_{i,m}(\cdot)]^{[\mathrm{dBm}]} = 128.1 + 37.6\log(d_{i,m}) + \chi_\Omega \tag{2.3}$$

[4] We neglect the interference from channels and signals that are always active even without users.

[5] $(\cdot)$ expresses the dependency of the channel losses on multiple parameters, e.g., the antenna tilt, azimuth, or the distance between the UE and the base station.

where $d_{i,m}$ corresponds to the distance between the transmitter and receiver and χ_Ω represents a zero-mean Gaussian distributed random variable (in dB) with standard deviation Ω. The last term in Eq. (2.3) measures the so-called shadow fading caused by large obstacles, e.g., buildings or hills, that obscure the line-of-sight signal between the base station and the UE. The second term in Eq. (2.3) expresses the increment in the link budget as the distance between a transmitter and a receiver increases. It is worth mentioning that the channel losses are applied to the desired signal as well as the interference. Therefore, in an ideal scenario, to minimize inter-cell interference, two co-channel cells should be located far away from each other.

A side effect of a high interference level is a degradation in the quality: e.g., missed or blocked calls if there is interference on control channels or cross-talk in traffic channels (especially for voice calls). Apart from degradation in the network quality, there is also an impact on the capacity of the system. As a matter of fact, there is a fundamental upper bound on the achievable capacity, given by the Shannon theorem [13] and expressed as in Eq. (2.4).

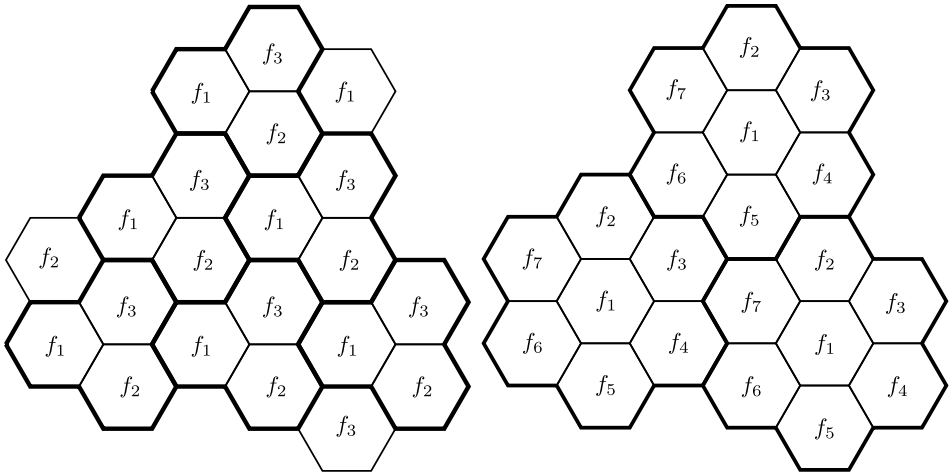
$$c_{i,m}(\boldsymbol{\rho}, \cdot) = \mathrm{BW} \log_2 \left(1 + \mathrm{SINR}_{i,m}(\boldsymbol{\rho}, \cdot)\right) [bps] \tag{2.4}$$

where BW is the bandwidth assigned to a base station. The smaller BW, the smaller the UE achievable data rate. The higher the interference levels, the lower the $\mathrm{SINR}_{i,m}$ and therefore, the lower the system capacity. That is why current mobile networks are considered interference-limited systems.

To mitigate the interference created by the neighbor cells, it is possible to divide BW into several subbands and assign them with a specific reuse scheme. Of course, the effective bandwidth allocated to each cell is reduced, which negatively impacts the achievable rate according to Eq. (2.4). The advantage of such an approach is that the distance between two co-channel cells is much higher, which reduces the interference. To illustrate this tradeoff, we introduce the "reuse cluster" concept as the set of cells that collectively use the complete BW.

In Fig. 2.2 two reuse schemes are depicted. In Fig. 2.2(a) a frequency reuse of 3 is used (please notice that the size of the reuse clusters is 3 cells), whereas in Fig. 2.2(b) a reuse scheme of 7 is implemented. It is possible to see that, for the same network layout, the distance between two co-channel cells is higher for the reuse 7 scheme (although the bandwidth assigned to each base station is lower).

However, to improve the spectral efficiency (defined as the information rate that can be transmitted over a given bandwidth in kbps/Hz), it is desired to have frequency reuse equal to 1 (i.e., the whole BW should be allocated to each cell). To improve the



(a) Reuse 3 (b) Reuse 7

Figure 2.2: Frequency reuse schemes

spectral efficiency while keeping low levels of interference, multiple frequency-based mitigation techniques have been proposed, as explained in Section 2.1.2.

2.1.2 Interference Mitigation Schemes

Improving the Quality of Experience (QoE) of the UEs is challenging, especially at the cell edge, where the radio propagation losses from the serving cell are expected to be higher as well as the interference level from the neighborhood. ICIC is a SF proposed to mitigate inter-cell intra-frequency interference (as will be detailed in Section 2.2). It can be carried out in several radio resource domains: time, frequency, space, and power (or a combination of them). For any of those domains, it is possible to find multiple studies with attractive candidates to mitigate Inter-Cell Interference (ICI) in various scenarios (either homogeneous or heterogeneous). The reader is kindly referred to [15], a good survey regarding modern ICIC techniques in the multiple radio resource domains.

This thesis focuses on frequency-based ICIC techniques, which divide each cell into center and edge regions and then allocate different subcarriers to UE in different locations. Two main branches are discussed in the literature [16]:

1. Hard Frequency Reuse (HFR): a.k.a reuse-n techniques, in which the whole frequency band (BW) is divided into n equal but orthogonal subbands and the adjacent cell will be allocated different subbands, as it was explained in Section 2.1.1. As we saw, these schemes reduce the ICI at the cost of very low bandwidth

utilization (i.e., low spectral efficiency). Only $1/n$ resources are utilized in each sector.

2. Fractional Frequency Reuse (FFR): this scheme offers frequency reuse factors between 1 and 3. It divides the whole available resources into two subsets or groups to serve the cell-edge UEs as well as cell-center UEs. This category is mainly divided into [16]:

   a) Partial Frequency Reuse (PFR): a.k.a. strict FFR, allows a "common" subband to be used in all sectors with equal power (i.e., that subband exhibits a reuse factor of 1), while the power allocation of the remaining subbands is coordinated among the neighboring cells to create one subband with a low ICI level in each sector, which is commonly assigned to edge UEs (i.e., those subbands have a reuse factor larger than 1).

   b) Soft Frequency Reuse (SFR): BW is divided into three adjacent orthogonal subbands in each cell. Two subbands are allocated to the center UEs, and the third subband is allocated to the edge UEs, as shown in Fig. 2.3.

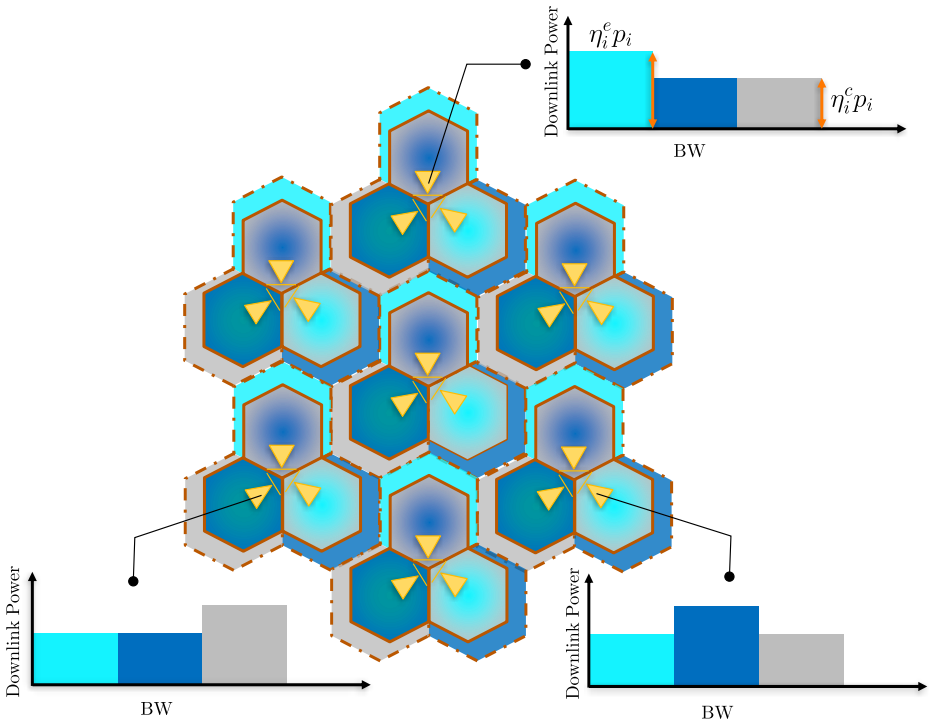


Figure 2.3: Soft frequency reuse scheme

As SFR provides a higher spectrum reuse factor [17] (and therefore higher spectral efficiency), we focus on it throughout the rest of the document.

To formally introduce the SFR scheme depicted in Fig. 2.3, let $\mathcal{B} = \{1, ..., B\}$ be the set of B cells in a macrocell network. We assume that the maximum transmit power for i-th next-generation Node B (gNB) is p_i in Watts, with $\boldsymbol{p}$ being the (maximum) downlink power vector. As expected in a SFR pattern, no two neighbor cells use the same frequency subband at the edge. In this way, ICI is mainly caused to an UE at the edge of the cell by a signal transmitted toward the neighbor cell's center users. Consequently, if the neighbor reduces its transmit power towards its center users, ICI is also further reduced. Thus, in SFR, every cell applies reduction power factors η_i^c and η_i^e at the center and the edge respectively. Therefore, the transmit power for resources that are assigned to edge users is $\eta_i^e p_i$ whereas for the resources assigned to center users is $\eta_i^c p_i$. Since we require high throughput and coverage for users at the edge, we set $\eta_i^e = 1$ for all cells as in [18].

Unfortunately, the use of SFR leads to natural tradeoffs among performance metrics such as coverage for cell-edge users, network throughput, and spectral efficiency. That compromise becomes more complex if the load imbalance among neighboring cells is considered because it causes inefficient resource utilization and low throughput. Accordingly, a tight relationship between ICIC and MLB functions is evident, as well as the necessity of considering a coordination scheme between both SFs. Additionally, as different user association schemes will result in diverse interference levels in the network, ICIC schemes should also be jointly considered with user association. We elaborate on this idea in detail in Chapter 5.

2.1.3 Radio Resource Management

The main objective of Radio Resource Management (RRM) mechanisms is to ensure efficient use of resources taking advantage of multiple adaptation techniques while supplying the UE with the requested QoS. The main RRM mechanisms for an OFDMA-based network[6] work in conjunction with the SFs and are distributed across the Layers 1 to 3 in the base station, as shown in Fig. 2.4, which is adapted from [19]. Please recall that we consider only the downlink channel, which is OFDMA-based.

The RRM functions in Layer 3 in Fig. 2.4 are considered semi-dynamic mechanisms in the sense that they are only executed during the setup of a new data session, whereas the function in Layers 1 and 2 are highly dynamic as their execution takes place every Transmission Time Interval (TTI) which is fixed to 1 millisecond for LTE or flexible for 5G (0.125 milliseconds being the lowest, used for latency-sensitive traffic).

[6] LTE uses OFDMA in the downlink whereas 5G uses it for both uplink and downlink.

Regarding the interaction between RRM and SON, we conceived the SFs coordination and execution as a slow outer loop that consumes information from faster RRM cycles (every 1 millisecond) as shown in Fig. 2.5.

Layer	User Plane	Control Plane	RRM Mechanisms	Execution
Layer 3	PDCP	RRC	QoS management; Admission control; Persistent scheduling	During setup of data flows
Layer 2	RLC, MAC	RLC, MAC	H-ARQ manager; Dynamic scheduling; Link adaptation	Every TTI
Layer 1	PHY	PHY	PDCCH adaptation; CQI manager; Power control	Every TTI

Figure 2.4: RRM functions.

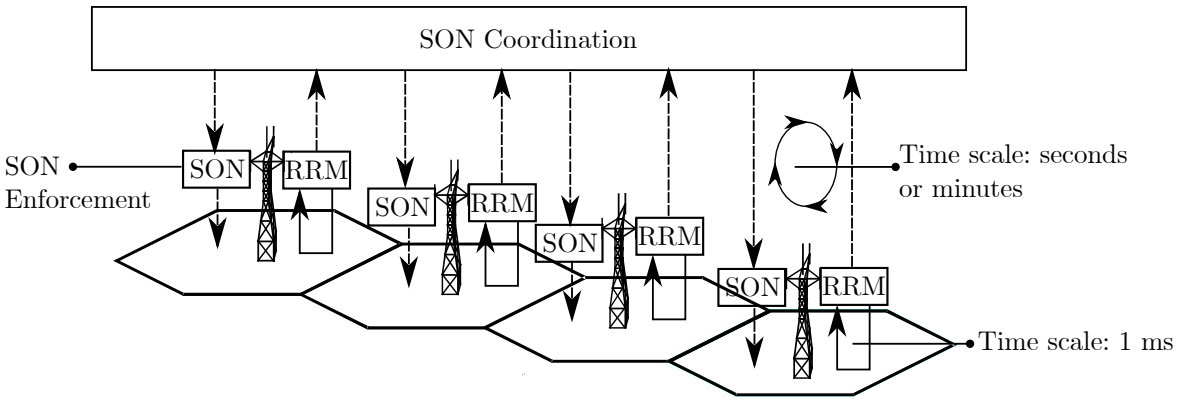


Figure 2.5: SON functions and RRM mechanisms interaction.

A detailed explanation of the RRM mechanisms in Fig. 2.4 is out of the scope of this document, and we focus especially on the mechanisms which have an impact (according to our criteria) on the SF execution, namely Channel Quality Indicator (CQI) manager, link adaptation and dynamic scheduling. A detailed explanation of the SFs will be provided in Section 2.2.

- CQI manager. This mechanism processes the Channel Quality Indicator reports (in downlink[7]) from active UEs. The CQI can be considered as a feedback from the user containing information about the channel quality. Based on these reports, the base station makes decisions about the link adaptation as well as the scheduling.

[7] As well as the Sounding Reference Signals (SRS) in the uplink.

- Link adaptation. This mechanism is executed in the time domain and is in charge of selecting the best modulation and channel coding schemes according to the channel conditions. Suppose the channel is noisy (meaning a low SINR). In that case, a robust modulation should be selected, e.g., QPSK (which allows transmitting 2 bits/symbol), whereas if the radio conditions are good, a higher-order modulation like 64-QAM (6 bits/symbol) should be selected, enabling a higher data rate. Likewise, the coding schemes (which represent the level of redundancy injected in the data flow) can change according to the channel conditions. This mechanism is also known as the Adaptive Modulation and Coding (AMC) scheme.

- Dynamic scheduling. The Layer 2 packet scheduler performs scheduling decisions every TTI, allocating Physical Resource Blocks (PRB)s to the users based on information like the CQI, QoS Class Identifiers (QCI), information about the retransmission status (H-ARQ), buffer status of the UEs, UE capabilities, available resources (PRBs), etc. The outputs of the scheduler correspond to the scheduled users, the PRB assignment to every user, the modulation and coding scheme selected for every transmission, and the transmitted power per PRB.

For the sake of completeness, we elaborate a bit more on the time-frequency resources scheduling because it is essential for MLB and ICIC modeling in the rest of the document. An OFDMA symbol (which can transport multiple bits depending on the selected modulation) mapped to a subcarrier represents a Resource Element (RE). These OFDMA symbols as REs are grouped in subframes of 1-millisecond duration (the same TTI duration), composed of two time slots of 0.5 milliseconds each. The time slots are further divided in the frequency domain into Resource Block (RB)s. Each RB comprises 12 sub-carriers of 15 kHz each making it 180 kHz wide (see Fig. 2.6). The minimum allocation unit to the users in the downlink is a PRB corresponding to two RBs, i.e., 14 symbols along 12 subcarriers.

The number of PRBs depends on the allocated channel bandwidth of a base station (i.e., BW). For instance, for BW = 10 MHz, the number of PRBs is 50. We define the resource utilization of i-th cell (ρ_i) as the ratio between the number of used PRBs and the number of available PRBs. Clearly, ρ_i represents the resource usage factor that was introduced in Eq. (2.2). In general, neglecting all the details of the scheduling process, ρ_i depends on the UE requested data rate as well as the capacity of the channel. It is possible to estimate the cell utilization through Eq. (2.5)

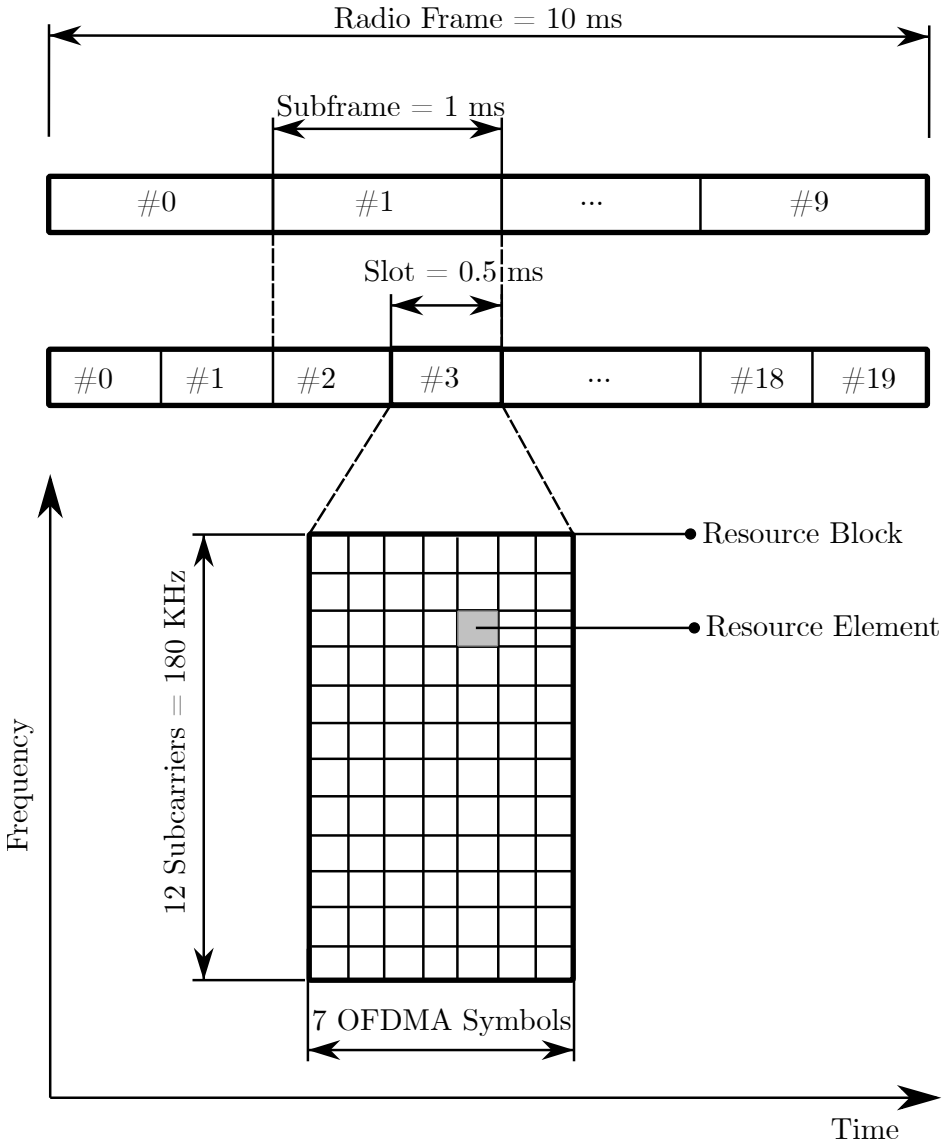


Figure 2.6: OFDMA grid

$$(\forall i \in \mathcal{B})\ \rho_i = \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \cdot)}, \tag{2.5}$$

where $\mathcal{P}_i$ corresponds to the set of users associated with the cell i, the numerator γ_l represents the UE requested data rate, and the denominator corresponds to the Shannon bound in Eq. (2.4). This estimation will come in handy, especially in Chapters 4 and 5, in which the impact (on the network performance) of several user associations ($\mathcal{P} \coloneqq \{\mathcal{P}_1, ..., \mathcal{P}_i, ..., \mathcal{P}_B\}$) is assessed before applying any change into the network.

2.1.4 A Minimalist Model of the Scheduler

There are many versions of scheduling algorithms in real network deployments since their design is not that standardized by Standards Developing Organization (SDO)s but rather left to the vendor's criteria. The scheduling policies for OFDMA-based systems can be classified into [20, 21]:

- Opportunistic algorithms: assume a full-buffer UE traffic model (i.e., UE traffic queues are continuously backlogged). They work by scheduling a UE when its instantaneous channel quality is higher than its average channel condition over time. Two main representatives are the Proportional Fair and Proportional Fair Exponential schedulers.

- Fair algorithms: this category of schedulers aims at improving the fairness among users. Fairness is an important requirement that should be considered to guarantee minimum performance even to users experiencing bad channel conditions (e.g., cell-edge users). The two main representatives are the Round Robin and Max-Min Fair schedulers. In this thesis, we consider a Round Robin scheduler.

- Throughput Based Algorithms: these schedulers aim to maximize the spectral efficiency by assigning each PRB to the UE that can achieve the maximum throughput in the current TTI. The main representative is the Maximum Rate scheduler.

- Delay Based Algorithms: these scheduling techniques are mainly designed for delay-sensitive applications and consider the time that a UE data packet has spent in the scheduler queue to be served first. The main representative is the Maximum Largest Weighted Delay First scheduler.

Considering the L2 scheduler is a critical component in the RRM as well as SON, and for the sake of gathering the main performance metrics related to the scheduling process, a minimalist model is proposed. This will be especially useful in the selection of a suitable cost function in Chapters 4 and 5.

We follow the modeling approach of a Round Robin scheduler presented in [22], in which the users are dispatched on a first-come-first-serve manner whenever there are enough time-frequency resources. If there are no resources (e.g., ρ_i is extremely high), we assume there is a queue that holds customers awaiting service.

Newly arriving UE data packets join the single queue, work their way up to the head of this queue in a first-come-first-serve fashion, and then finally receive a fraction of the time-frequency resources, i.e., PRBs (known as "quantum" in [22]). When that "quantum" expires and if they need more service (i.e., the data transfer has not been completed yet), they then return to the tail of that same queue and repeat the cycle. This scheme is known as the Processor-Sharing model, and it is depicted in Fig. 2.7.

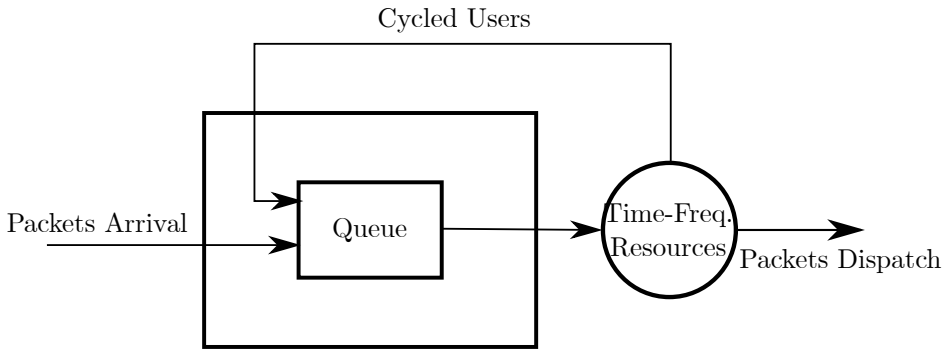


Figure 2.7: Processor-sharing M/G/1 scheduler

When there is no contention for the time-frequency resources (e.g., there is only one UE in the network), and the user needs x seconds to dispatch all its packets, then the average response time, $T(x)$, is equal to x. However, when multiple UEs share the same resources, some time will be spent in the queue, $W(x)$. Of course, $T(x)$ and $W(x)$ are direct measurements of the latency associated with the scheduler which should be minimized, especially thinking on Ultra-Reliable Low latency Communication (URLLC) services for 5G.

There are closed-form expressions for both $T(x)$ and $W(x)$ (normalized with the nominal service duration x with no contention), assuming that the whole packet arrival-dispatch process of the i-th cell can be modeled as a M/G/1 system:

$$\frac{T_i(x)}{x} = \frac{1}{1-\rho_i} \tag{2.6}$$

$$\frac{W_i(x)}{x} = \frac{\rho_i}{1-\rho_i} \tag{2.7}$$

Some properties of the Round Robin scheduler pop up: on the first-hand side, according to Eq. (2.6), the response time, $T(x)$, has a linear relationship with nominal service time x, i.e., a packet twice as long as some other will spend on average twice as long in the network. On the other side, and according to Eq. (2.7), if we consider $\frac{W_i(x)}{x}$ as a penalty for transmitting x seconds of service over a shared resource, then it is independent of the UE's service time distribution (which depends on the Shannon bound, and therefore on the SINR), and is in that sense "fair", as we mentioned it before.

Apart from the latency relationships in Eq. (2.6) and Eq. (2.7), it is possible to find in the literature a figure of merit for the data rate perceived in a M/G/1 - Processor Sharing queue. According to [23, 24], the throughput perceived by the m-th UE connected to i-th cell is given by Eq. (2.8).

$$R_{i,m} = (1-\rho_i)\, c_{i,m}(\boldsymbol{\rho}, \cdot) \tag{2.8}$$

where the second term corresponds to the Shannon bound in Eq. (2.4).

The term $(1-\rho_i)$ in all the previous equations, corresponds to the available resources in the i-th cell. In Chapters 4 and 5, we define a cost function that is compliant with the performance metrics in Eq. (2.6), Eq. (2.7), and Eq. (2.8).

2.1.5 Mobility Management

When an UE is either in a voice call or transmitting/receiving data traffic, it is said to be in active state (specifically, the user is in *RRC_CONNECTED* state [25]). If the user happens to be in movement, it is crucial to keep the ongoing session up regardless of the velocity and the movement direction. Otherwise, if the connection can not be retained by the network, there will be dropped calls/sessions, which affects the QoS of the users. As the user is moving, it is likely that the cell that is offering service to the user will change. Therefore, a network procedure to smoothly transfer an active session from one cell (source cell) to another one (destination cell) is employed.

The network uses the network-triggered and user-assisted HandOver (HO) procedure to control the user association for UEs that are actively transmitting or receiving data. The objectives here are to maximize the UE's data rate and the overall network capacity. The mobility management for active users is standardized in [26], [27]. In this research, we focus on the mobility management for UEs in the active state because the

HO rules the dynamics of two well-established SFs, MRO and MLB, which generates a conflict as it will be mentioned in Section 2.3.1.

According to the standard, the HO is a three-step procedure:

- **UE measurements**. The UE measures the signal levels from the serving and neighboring cells and sends the measurement report to the serving cell. To set up the measurements, the serving cell sends a *Radio Resource Control (RRC) Connection Reconfiguration* message to the UEs. Within that message, there is an information element called *measurement configuration*, which contains information about when the UE should report back to the serving cell. The measurement procedure is well-established in the standard, see [28], [29].

- **HO decision**. Based on the measurement report, the cell decides about HO triggering. The most general reporting mechanism in mobile networks is event-triggered periodic reporting, in which a UE starts to send periodic measurement reports if a signal level crosses over a threshold and stops if the signal crosses back [11]. Throughout this document, we consider the A3 HO event, which triggers the vast majority of HOs in LTE corresponding to intra-frequency HOs [11].

- **HO execution**. Corresponds to the signaling flow over the X2 interface (for LTE) or Xn (for 5G) or through the core network if the cells lack an interface among them. We do not spend too much time explaining this step since we consider the bottleneck of the network is the radio part rather than the backhaul or the core network. If more details are needed, [27] is a suitable reference.

In the A3 HO event, a HO is carried out whenever the neighbor cell Reference Signal Received Power (RSRP) becomes stronger than the RSRP of the serving cell by a specific offset, and this condition is met for a time of at least Time-To-Trigger (TTT) (between 0 and 5120 milliseconds). We estimate the RSRP as the product of the transmitting power of serving cell i, that is p_i, and the channel gain from cell i to user m, that is $g_{i,m}(\cdot)$[8], namely: $[p_i g_{i,m}(\cdot)]^{[\mathrm{dBm}]}$, in which the placeholder $[\cdot]^{[\mathrm{dBm}]}$ implies that the quantity within the brackets is expressed in dBm. Let $\mathcal{B} = \{1, ..., B\}$ be the set of B cells in a macrocell network, a simplified version[9] of the entry condition for an A3 HO from cell i to cell j is given by Eq. (2.9):

$$[p_i g_{i,m}(\cdot)]^{[\mathrm{dBm}]} + H + \mathrm{CIO_{i,j}} < [p_j g_{j,m}(\cdot)]^{[\mathrm{dBm}]}, \tag{2.9}$$

[8] $(\cdot)$ expresses the dependency of the gain on multiple parameters, e.g., the dependency on the antenna tilt which is studied in Chapter 4.

[9] The optional frequency-specific offsets have been neglected.

where the Cell Individual Offset (CIO) is a power offset given in [dB]. For the sake of avoiding coverage holes, we consider $\mathbf{CIO} \coloneqq (\mathrm{CIO_{i,j}})^{B\times B}$ with $\mathrm{CIO_{i,j}} = -\mathrm{CIO_{j,i}}$ and $\mathrm{CIO_{i,i}} = 0\ \forall\, i, j \in \mathcal{B}$. H is the hysteresis value which shows the difference needed between the measurements of the serving and the neighbor cells to trigger the A3 event also in [dB].

Likewise, the UE leaves the event A3 reporting when the condition in Eq. (2.10) is met for at least TTT.

$$[p_i g_{i,m}(\cdot)]^{[\mathrm{dBm}]} - H + \mathrm{CIO_{i,j}} > [p_j g_{j,m}(\cdot)]^{[\mathrm{dBm}]}, \tag{2.10}$$

While the entry condition is held, the UE sends measurement reports to the cell every *reportInterval* (from 120 milliseconds up to 60 minutes [29]). The maximum number of reports the UE sends can go from 1 to 64 (or unlimited) [29]. In Fig. 2.8, it is possible to visualize the aforementioned concepts.

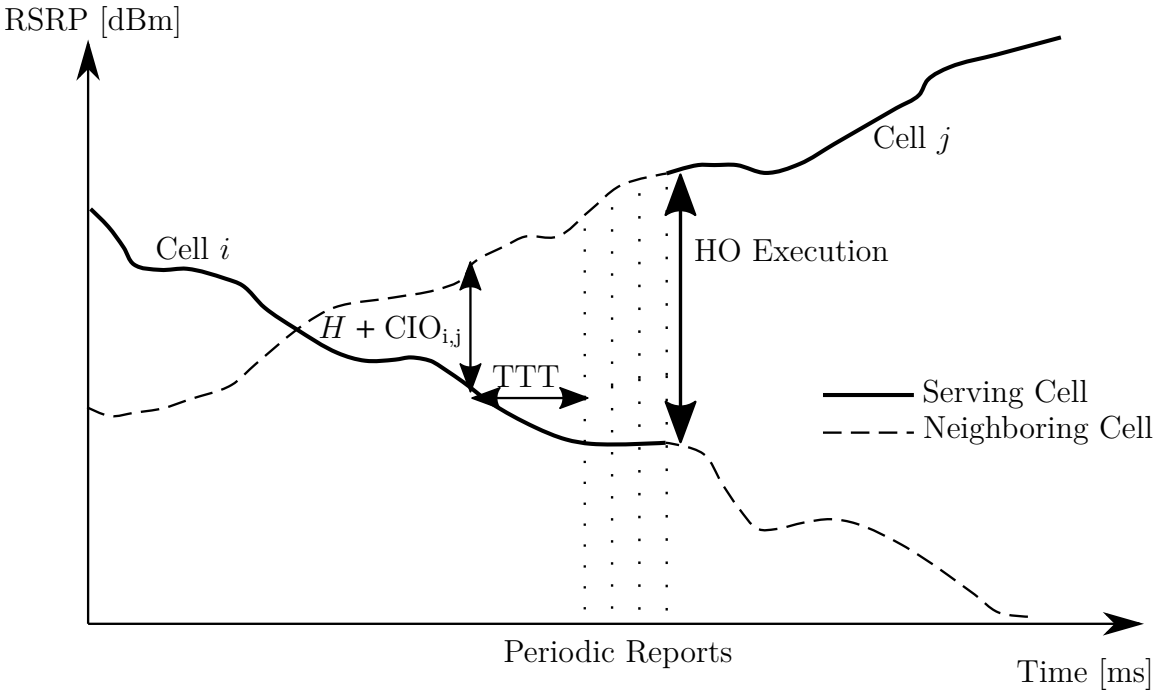


Figure 2.8: Measurement reporting and handover decision using A3 entry condition

2.2 SON: Origins and Evolution

We have described so far the main network-related concepts associated with the execution of SFs. In this section, we dive into the details about the historical background

and specifics of the SON concept, especially from the standardization perspective. As the main references in the rest of this chapter correspond to SDO's deliverables, we consider it relevant to picture them (at least the principal ones) along the standardization track. For mobile networks, the standards are defined by the third Generation Partnership Project (3GPP), which gathers contributions from members of multiple regional standards bodies. Before a standard is defined, the requirements and use cases need to be determined. This is carried out through forums either in the International Telecommunication Union - Radio sector (ITU-R) or NGMN Alliance (which correspond to industry forums). Additionally, research forums are considered, especially the projects part of the 5G infrastructure Public-Private Partnership (5G-PPP) and 5G Mobile communications promotion Forum (5GMF), which collect research input from academia and industry.

SFs were proposed and standardized to provide self-adaptation capabilities to mobile networks on different fronts: configuration, optimization, and healing, thus reducing the error-prone human intervention. Unfortunately, the conventional design of these SFs was based on single objective optimization approaches where SFs were considered as standalone agents aiming at one very specific local objective (e.g., reducing the interference or increasing the coverage). Thus, complex interdependencies between SFs were, to some extent, unattended, so when more than one function is acting on the network, conflicts are inevitable. Therefore, it was needed to introduce a self-coordination stage to settle the conflicting scenarios.

A taxonomy of the SFs (as introduced by the SDOs) is proposed in Fig. 2.9. Five categories group all the SFs: self-configuration/optimization/healing/auxiliary functionalities and self-coordination (the latter is the focus of this document).

SON was initially conceived by the NGMN Alliance to address expected challenges during the continuous operation and management of multiple RATs [1]. Soon after [1] was issued, the standardization process for the use cases proposed by NGMN Alliance (not all of them) started and continues since then. The 3GPP Release 8 of LTE defined the basic concepts and requirements regarding initial deployment, rollout, and integration of networks, especially considering the **Automatic Neighbour Relation (ANR)** and **automated configuration of Physical Cell Identity (PCI)** functions [30], [31].

During the Release 9 of LTE, and product of the work of the 3GPP RAN3 working group, canonical SFs for intra-RAT self-optimization were introduced [32], namely:

- **CCO**. This function deals with the joint optimization of the coverage and capacity of the network. The CCO concept was defined for LTE in [33, 34, 35].

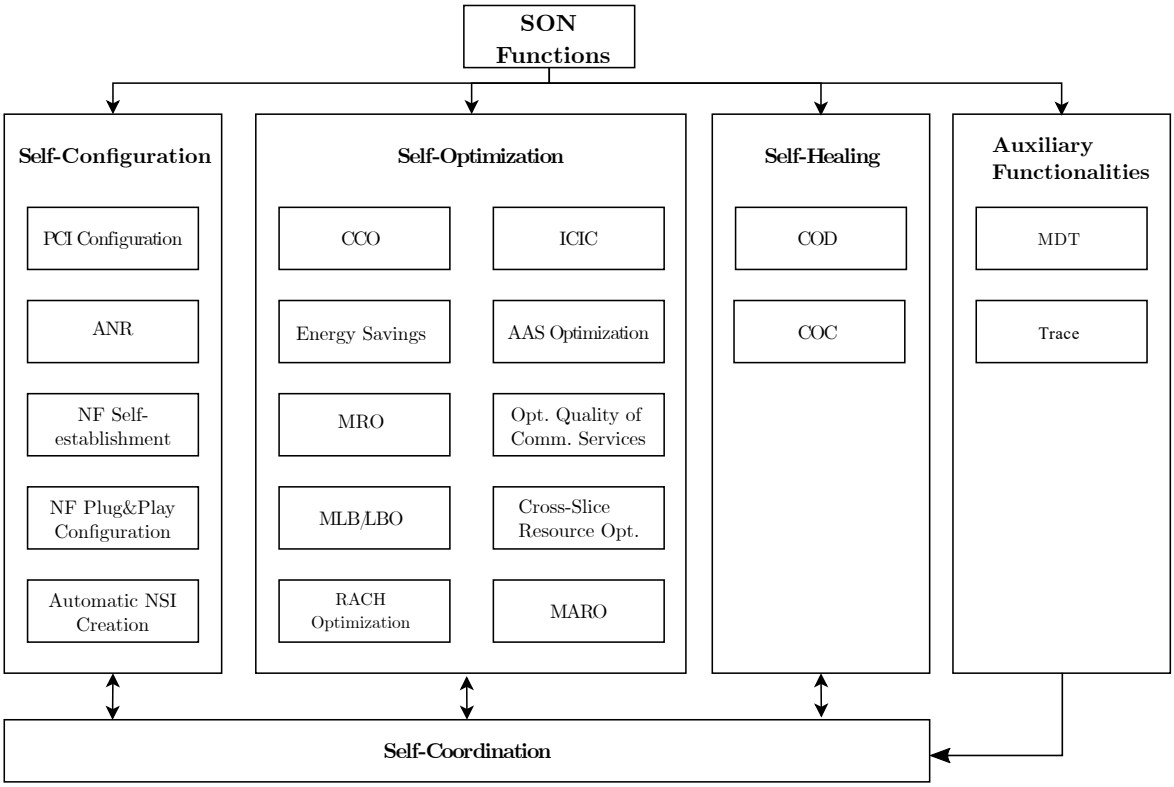


Figure 2.9: SON functions taxonomy and functional split

According to the SDOs, this function applies to 5G; however, the functions specific to 5G radio technology (e.g., beam management) need to be taken into account [2].

- **ES**. With this SF, the energy savings are achieved through switching off inactive network cells or by reducing the transmit power, as long as the UEs QoS is not affected. This function is commonly executed during off-peak-traffic (low traffic demand) scenarios - we will observe this in Chapter 5. The coverage, as well as the capacity of the energy-saving cell, must be taken over by neighboring cells, either in the same RAT (intra-RAT ES) or a different one (inter-RAT ES). Intra and inter-RAT ES for LTE are proposed in [36], whereas, for 5G, an application study is available in [37].

- **Automated configuration of PCI**. This function also belongs to the self-configuration category, and it is in charge of automatically assigning a unique identifier to a New Radio (NR) cell by the gNB Distributed Unit (DU) and the management system. It was standardized for LTE in [27] (see clause 22.3.5), and an applicability study in 5G is presented in [37].

- **MRO**. This self-optimizing function aims at detecting and solving the problems associated with inter and intra-RAT mobility, e.g., Radio Link Failure (RLF)s due to too-late HO, too-early HO, HO to wrong cell, unnecessary HO, Ping-Pong (PP) HO. The definition of MRO for LTE is given in [34, 27], and a study for applicability to 5G can be found in [37].

- **MLB or Load Balancing Optimization (LBO) function**. This function corresponds to a self-optimizing use case. MLB (nowadays LBO), enables overloaded cells to transfer part of their loads to neighboring cells, yielding a fairly distributed load across the network, decreasing the wastage of resources, increasing the probability of acceptance of new sessions, and boosting the throughput. This process is carried out by modifying intra-RAT and inter-RAT mobility parameters (like MRO), as will be explained later on. The load balancing concept was introduced for LTE in [34], [27], whereas a study for 5G is available in [37].

- **Random Access CHannel (RACH) optimization**. This is another self-optimization function. The RACH needs to be adequately configured to provide enough random access opportunities to the UEs regardless of the cell sizes. RACH optimization is in charge of finding the best trade-off between access opportunities and the resources which have to be sacrificed for them. The conceptual design for LTE is available in [27], whereas a study for its application in 5G can be found in [37].

- **ANR**. It traditionally belongs to the self-configuration category. It is in charge of discovering neighbors and creating/updating the corresponding relationships, as well as automatically setting the X2 (LTE) and Xn (5G) interfaces among neighbors. It was introduced for LTE in [38]. For 5G, the functionality is presented in [39, 40, 41]

- **ICIC**. This use case belongs to the self-optimizing category. This SF corresponds to a multi-cell RRM function whose objective is to mitigate ICI and improve the SINR, especially at the cell edge. ICIC needs to take into account information (e.g., the resource usage status and traffic load situation) from multiple cells to make the most favorable decisions. ICIC mechanisms for LTE were proposed in [27], including both frequency and time domain techniques. ICIC also applies to 5G according to [2].

As from Release 10, enhancements to some of the previously defined SF were proposed, as well as adaptations to support multi-RAT scenarios and the introduction

of self-healing networks[10]. The concepts of self-healing for LTE are described in [42]. The main representatives for this category of SFs are:

- **Cell Outage Detection (COD)**. In charge of detecting sleeping cells[11], out-of-service cells, and so on [42].
- **COC**. Once an outage state is detected in one cell, this function is in charge of distributing the load of the affected cell across the neighborhood [42].

The **Minimization of Drive Testing (MDT)** (coming from the 3GPP RAN2 working group) functionality was also introduced in the 3GPP Release 10 of LTE as a way to collect UE measurements either when the UE is in active or idle mode. As its name implies, the idea is to minimize the frequency of conventional and expensive measurements made using specialized measurement equipment installed on vehicles. For LTE, MDT was introduced in [43], and the application to 5G is discussed in [37]. A complementary functionality has been introduced in 3GPP Release 15: **Trace and reporting of Tracing Data (Trace)**, indicating the ability to trace all active calls in a cell or multiple cells as well as keep records of control signaling information for specified interfaces [44].

In Release 14, a SF for **Active Antenna Systems (AAS)** in LTE was introduced [33, 34] to create a framework for splitting and merging cells as well as changing the cell shaping. In 5G, it is expected that this task will be absorbed by the CCO SF [2]. From Release 16 on, and mainly regarding 5G, the following SFs have been under definition:

- **Network Slice Instance (NSI) resource allocation optimization**. This function adjusts the allocation of radio and core resources (e.g., scaling in or out the resources) for the NSIs [2]. This SF is mainly supported by predictive reports supplied by the Network Data Analytics Function (NWDAF) Network Function (NF).
- **Self-establishment of 3GPP NF**. This SF is based on a server-agent interaction. A recently deployed node requests configuration data and software from a self-configuration service producer, and it receives the configuration data

[10] A critical factor in fault management if we consider that in the Radio Access Network (RAN), every cell is in charge of serving a dedicated coverage area with little (if any) redundancy. Therefore, reducing Mean Time To Repair (MTTR) is crucial

[11] A cell which carries no traffic despite the presence of UE requesting service but not generating any alarm.

that may include radio configuration data, IP connectivity data, and software installation. Once the information is received, the node proceeds with the set of instructions received [2].

- **Multi-vendor Plug & Play of NFs**. Once a NF is deployed, either Physical Network Function (PNF) or Virtual Network Function (VNF), this SF is in charge of connecting the NF to the management system providing support for self-configuration processes as automatically as possible [2].

- **Automatic NSI creation**. This SF enables operators to create the NSI that contains 5G Core NF, gNB Central Unit (CU), and gNB Distributed Unit automatically, based on the requirements given by the authorized customers [2].

- **Optimization of the quality of communication services**. This SF enables service quality optimization. For example, the goal could be to minimize average latency on the communication services provided to the URLLC category of services. If this SF detects a performance degradation, it modifies the configuration parameters in the corresponding RAN and core nodes or NSIs/Network Slice Subnet Instance (NSSI)s [2].

- **Cross-slice network resource optimization**. This SF is in charge of optimizing the resource allocation across multiple NSIs of the total available physical and virtual resources [2].

- **Multi-aspect / multi-domain resource optimization (MARO)**. The objective of this SF is to enable joint resource optimization across multiple domains (e.g., access network, core network) among multiple network aspects, including the radio resources, VNFs, and network slices [2]. This SF uses Fault, Configuration, Accounting, Performance, Security (FCAPS) information which is analyzed, and resource provisioning policy improvements are generated afterward, for instance: changing the characteristics of transport links based on observed transport network performance bottleneck or changing the network slice or subnet resource provisioning for inter-slice resource optimization [2].

As it will be seen in Section 2.3, multiple SFs tweak the same (or related) network parameters, usually under incompatible policies, generating instabilities and conflicting scenarios. Thus a coordination mechanism is imperative. On that note, the Self-coordination concept was introduced for LTE in the 3GPP Release 11 [34], which is also applicable to 5G, although the adaptation to 5G is still under construction as

the early coordination framework does not account for multiple network aspects, e.g., virtualization or slicing [2].

As shown in Fig. 2.9, the self-coordination framework spans over the self-configuration, self-optimization, and self-healing categories and could also receive input data from the Auxiliary Functionalities branch (as a matter of fact, that is one necessary condition for the coordination approach proposed in Chapters 4 and 5).

2.3 Conflicts among SFs

Throughout this research, we mainly consider "canonical" functions of the self-optimization branch in Fig. 2.9, namely: MLB, MRO, ICIC, CCO, and ES. From every possible combination of SFs drawn out of this subset, it is possible to discover a conflicting scenario that turns a coordination scheme into a major necessity. With the help of Fig. 2.10, some conflicting scenarios are illustrated in the following subsections. Because we pursue a centralized solution, whenever we mention a specific SF, we mean the central entity which has visibility of other instances of the same or different SFs in the whole network.

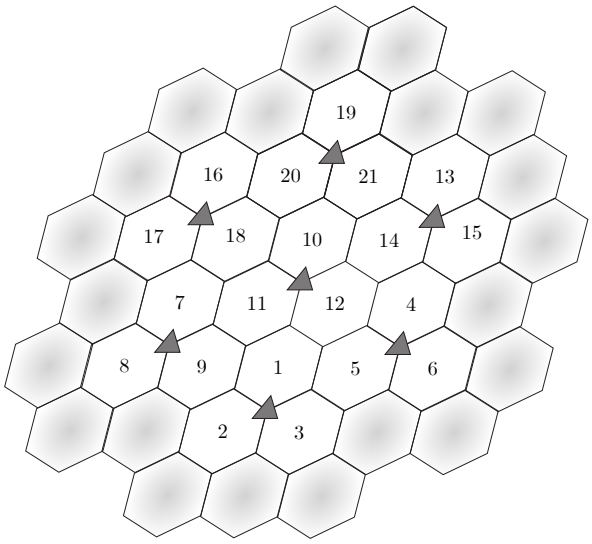


Figure 2.10: Coordination among multiple SFs.

2.3.1 Conflicts between MRO and MLB

Both MRO and MLB functions optimize network performance by adjusting handover parameters and influencing the same KPIs. Typical implementations of MRO consider

TTT and H in A3-HO events (see Section 2.1.5) as input parameters, whereas MLB uses the CIO values: $CIO_{i,j}$ [34].

Since two different SFs modify parameters related to the same network procedure (in the opposite direction or towards the same direction but on different scales), conflicting commands could take place, as shown in the following hypothetical scenarios:

1. The MRO function may adjust handover parameters (e.g., increase H of source Cell 1 in Fig. 2.10) to minimize the number of too early handovers from Cell 1 to Cell 2, while the MLB function may adjust handover parameters (e.g., decrease the CIO of source Cell 1 to target Cell 2) to advance the handover from Cell 1 to Cell 2 in case the load of Cell 2 is much less than Cell 1. A conflict is evident here.

2. If the MLB function detects one cell is overloaded, it could try to shrink its cell size, transferring edge users to the neighborhood. Thus, the overloaded cell could reduce the CIO for some of its neighbor cells (advancing some HO procedures towards the tier-one neighbors). Nevertheless, if the radio conditions in a destination cell are not good enough for the users that were handed over, some RLF (due to too-early handovers) could occur, so the MRO function (after analyzing the performance metrics) adjusts the neighbor relationship between the overloaded cell and the neighbor cell such that the UEs do not handover that easily to the neighborhood. However, that jeopardizes the original load balancing procedure driven by MLB in the overloaded cell. Therefore, orchestration between both SFs is paramount.

3. Alternatively, if one user is moving at high velocity in a heterogeneous network (where macrocells co-exist with micro and femtocells), the user should be served by a macrocell to avoid multiple subsequent HOs (e.g., towards and inbetween microcells) and in such a way to reduce signaling traffic, so MRO function in the macrocell could increase H or the TTT (and in that way delaying the HOs) which in turn increases the offered load in the macrocell which could advance the triggering of some MLB actions to compensate a possible overload. Once again, the need for coordination between both SON use cases is evident.

This kind of conflicts[12] jeopardizes the global stability of the system and consumes a significant amount of signaling resources. We study this conflict within our proposed coordination framework in Chapter 3.

[12] Known in the literature as output parameter conflict [45].

2.3.2 Conflicts between ICIC and CCO

This well-known conflict occurs when the same network parameters are modified in different directions due to the greedy objectives of individual SFs. Let us consider the following scenario: the antenna downtilt in Cell 1 could be reduced by a CCO instance to increase the coverage over a specific area, but an increase in the ICI created by Cell 1 is expected. Once the generated ICI is increased, the ICIC function may try to increase the downtilt of the antenna again. With the change rollback, the network ends up in the initial scenario, and the situation could be triggered again. Therefore, a coordination mechanism coping with a compromise between both objectives is required. This type of conflict is known in the literature as a direct characteristic conflict [45].

2.3.3 Conflicts between MRO and CCO

This conflict is categorized as a measurement conflict [45], which is present when a change in the network configuration influences the network metrics, but it takes some time until those changes show an effect in the measurements. For example, an instance of CCO could change the tilt configuration of a cell according to its optimization policy. This change has a long visibility delay time (in the network statistics), so in the meanwhile, an instance of MRO could decide to modify handover parameters based on out-of-date information. Therefore, once the change of tilt is visible in the network, the solution found by MRO could become obsolete. We propose a method to deal with this conflict (as future work) in Chapter 6.

2.3.4 Conflicts between MLB and CCO

To illustrate this conflict please consider (as an example) the cell layout depicted in Fig. 2.10. The conflict occurs when an instance of CCO in Cell 1 modifies the transmitting power or electrical tilt, impacting the cell size. A MLB instance simultaneously running on a neighboring Cell 2 modifies the CIO between Cell 2 and Cell 1 to achieve a better load distribution between the cells. However, the size of Cell 1 modified by the CCO function has an impact on the handover procedure as the overlap area with the neighboring Cell 2 changes. This causes MLB parameter changes in Cell 2 to be potentially unsatisfactory. We handle the coordination of this specific problem in Chapter 4.

2.4 SF Coordination Complexity

It can be deduced from the above scenarios that the relationship mesh among SFs, network parameters, and KPIs is quite interleaved. This is mainly due to the fact that there is a vast number of parameters related to SFs, and the number of relations is increasing as more SFs and network functionalities are introduced by the SDOs.

For the sake of reference, a list of crucial SON-related parameters has been considered by the standard for LTE in [46] and 5G in [34], as well as the SON-related network performance information in [37]. To get an idea of the complexity of the conflict dynamics among SFs, Fig. 2.11 shows the degree of interaction amidst some SFs concerning the network parameters for one (and only one) specific RAT and NSSI. The straight lines represent direct parameter-SF interaction, whereas the dashed lines represent indirect interaction, e.g., that a change in a particular parameter affects the SF decision-making process.

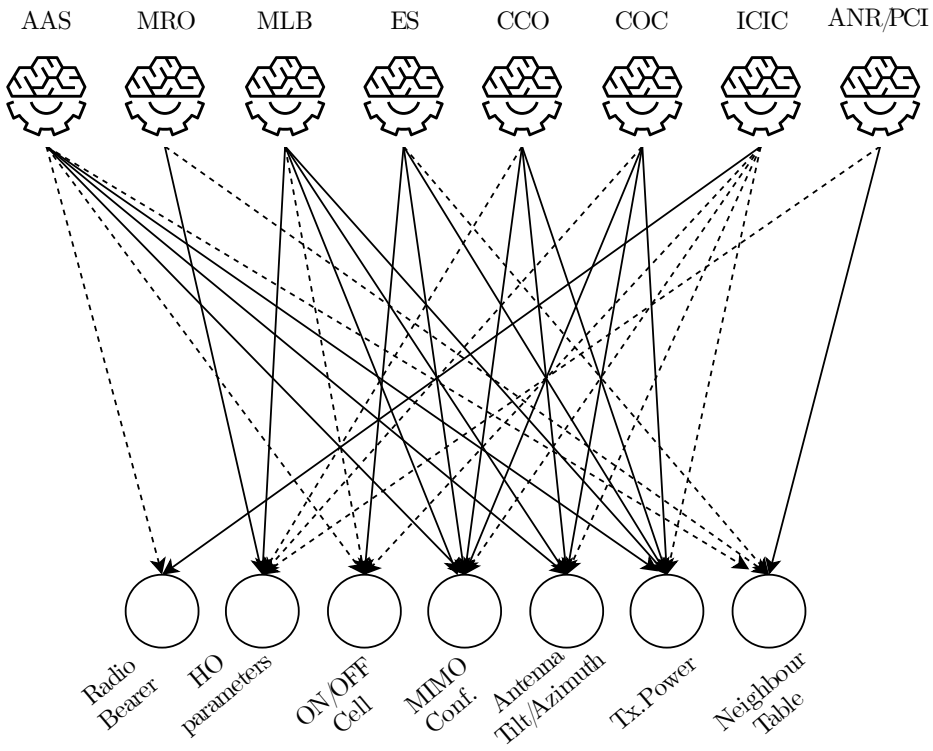


Figure 2.11: Interaction mesh among SFs and their corresponding network parameters

It is worth it mentioning that the location of the functions within the network has not been explicitly stated yet in Fig. 2.11. As it will be shown in Section 3.2.1, there are conflicts between the same SF (or different SFs) executed in neighboring cells concurrently. We call this kind of conflict: inter-cell dependencies. One example of

these dependencies is the conflict between CCO and ICIC described in Section 2.3.2. As the reader can guess, there are also intra-cell dependencies, e.g., the relationship between ANR and MLB/MRO.

The complexity of the coordination problem explodes if heterogeneity in the network is introduced, as well as concepts like slicing or MIMO are considered. For example, consider the mobility management ruled by the handover procedure: it is expected to have explicit scenarios for inter-RAT mobility, inter-beam mobility, or inter-slice mobility, increasing the complexity of the SFs modifying the handover parameters, e.g., MRO and MLB (see Fig. 2.11). Consequently, the introduction of new network functionalities and new SFs yields an increase in the dimensionality of the coordination problem.

Furthermore, it is worth mentioning that an operation management domain could be highly customizable, in the sense that not all the operators will have an instance for all the SFs (this will depend on the business model of each operator) and probably not for all of the RATs or geographic areas. In addition, any attempt to coordinate SFs should be vendor-agnostic and scale to multiple network sizes: from legacy PLMN to small deployments available, for instance, in Non-Public Land Mobile Networks (N-PLMN), a.k.a. campus network scenarios.

At this point, the convolution of the coordination problem is evident, so simultaneously considering multiple SFs increases the difficulty of modeling the network dynamics in a closed-form way. Therefore, it is expected that, for the next generation of mobile networks, a smart approach to coordinate all SFs shall be defined. Here comes ML into the landscape as an enabler for 5G to provide a smart automation layer to the network, easing the network management as long as enough information is available to train a trustworthy ML model.

Accordingly, it is expected to have a smart layer hosting the ML models and performing the "self-everything" tasks in the network. Thus, apart from the typical functional split in the network (i.e., user plane, control plane, management plane), a so-called Automation Plane is conceived, as in Fig. 2.12, which is adapted from [47].

In the section below, a thorough analysis of the state of the art of SF's coordination methods is provided along with a non-standard taxonomy of the considered references. Important SON-related research projects are also mentioned.

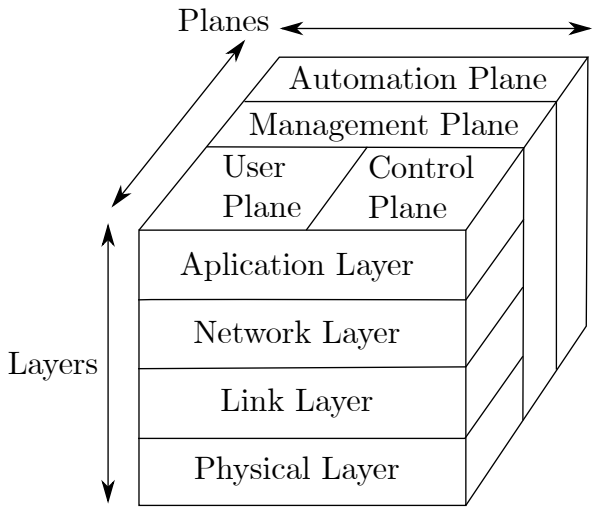


Figure 2.12: Communication layers and planes.

2.5 SF Coordination: the State of the Art

According to a review of the available literature, we consider that there are five research directions regarding the coordination of SFs. This taxonomy is far from standard and is used to ease the framing of our contributions in this thesis.

- (Temporal-spatial) separation. In this category, the execution of conflicting SFs is carried out using a temporal or spatial separation. In general, with the execution separation, the complexity of the interaction among SFs is reduced. However, we claim that this approach is too slow in order to meet the requirements of highly dynamic network environments.

- Explicit coordination. In this branch, there is explicit modeling of the interactions among conflicting SFs. The downside of these methods is two-fold: deep expertise in network modeling is needed, and highly simplifying assumptions about the model conditions are required (otherwise, the modeling is intractable).

- Tailing (or a posteriori) coordination. In this category, it is considered that many agents are executing greedy optimization algorithms, finding local policies, and thereafter a coordinator conciliates the possible conflicts among the local policies. A drawback of these approaches is that the optimality achieved by the individual agents is lost with the conciliation stage; therefore, there is a lack of optimality understanding during the mediation process.

- Heading (or a priori) coordination. The methods within this category are meant to avoid the conflicts among SFs from their design. The idea is that with the

right design of conflict-free SFs, network instabilities disappear. Similar to the explicit coordination category, deep expertise is needed to design functions without conflicts. Furthermore, strong assumptions about the model conditions are required. Additionally, as was shown in Section 2.2, the design of SFs is not that flexible because it has constraints from the standardization perspective.

- Implicit coordination. In this category, the coordination problem is rendered into a joint optimization problem, in which the optimization of a global metric (or utility) is sought. The utility definition hides all the complexity of the network dynamics and should be aligned with the business strategy of the network operator.

In the following subsections, we present a detailed discussion about some representatives of the above categories.

2.5.1 Temporal-Spatial Separation

There have been several research projects focusing on the study of SON. One of them is the FP7 SOCRATES (Self-Optimization and self-ConfiguRATion in wirelESs networks) project, in charge of developing and demonstrating algorithms for self-coordination, optimization, healing, and coordination of 3GPP-compliant networks. Within the SOCRATES project, and regarding the self-coordination branch, the authors in [48] propose two schemes to mitigate conflicts among SFs. First, they propose a SF execution separation in the time domain depending on the number of measurements needed to make a statistically-meaningful decision to modify the SF-related parameters; the authors call this the "separation approach", and they claim it substantially lowers the overall SF interaction complexity. The authors also propose a so-called coordination method based on Hidden Markov Chains to determine precisely when a reoptimization should occur; the authors claim that the advantage of the "coordination scheme" is that the re-optimization is triggered when it is most appropriate and not at predefined intervals. Therefore, with the coordination scheme, it is possible to react faster than with the separation scheme because the latter restricts reoptimizations to the end of measurement intervals of predefined lengths. Unfortunately, the authors in [48] do not present any quantitative results for any of the SFs introduced in Section 2.2.

Regarding separation schemes, the authors in [49, 50] also propose a spatial-temporal decoupling of the SFs. Every SF is modeled as an agent running Q-Learning, and their

execution is separated in either space, time, or both. In [49, 50], simulation studies considering MLB and MRO showed that a distributed coordination approach where SFs learn to minimize their effects on one another is a promising solution for network management. The authors in [51] also provide a time-separation-based execution of SFs. The authors study the conflicts between MRO and ES and present simulation results regarding the reduction of the power consumption and the HO failures while solving an integer linear programming problem.

2.5.2 Explicit Coordination

It is also possible to find C-SON approaches based on an explicit formulation of the coordination problem. For instance, the authors in [52, 53] coordinate the execution of MLB, CCO, and COC using an explicit model of the dynamics of the SFs through the formulation of a joint optimization problem. A solution to the optimization problem is found using a structured search algorithm called the Taxi Cab Method and a network partitioning method. In [54], the authors extend the work in [52, 53] by adding a rule to progressively switch off some base stations while ensuring a minimum coverage. Thus, [54] presents a centralized framework based on Taxi Cab Method and network partitioning to optimize MLB, CCO, and ES jointly. In [55], the authors coordinate the execution of MLB, ICIC, and ES in a heterogeneous network with macro and pico cells, using network partitioning techniques and the Nelder-Mead algorithm to solve a bandwidth allocation problem in a FFR-based network.

Regarding explicit coordination methods, but with decentralized implementation, the authors in [56] propose a mechanism to stabilize a system of ordinary differential equations using control theory, specifically Rosen's concave games. The authors study the coordination among MLB, Call Admission Control (CAC), and CCO. For this approach, there is a strong assumption about the linearity of the system, and every SF is modeled through one (and only one) KPI and one network parameter. That work was carried out within the FP7 UNIVERSELF project [57], a major SON-related research initiative along with SOCRATES. Another reference considering explicit modeling is available [58]. The authors propose an open-loop estimation of "good-enough" parameter sets and a fine-tuning task of the estimation using a slower closed loop. The first estimation of the parameter set is carried out considering two types of parameters per SF: one regarding the cell size (or cluster) and another regarding the resource usage efficiency within the cell (or cluster). Explicit modeling of the relationship between the two types of parameters per SF and the involved KPIs is needed. Results consid-

ering an energy-related KPI through the modifications related to MLB and ICIC are available in the [58].

2.5.3 Tailing Coordination

Another branch of the self-coordination theory in the literature corresponds to tailing or a posteriori coordination. The authors in [59] propose a functional architecture based on two steps: first, every SF is decomposed as a Markov Decision Problem, which is solved using reinforcement learning theory, more specifically the actor-critic algorithm; second, a coordination game is formulated to conciliate the individual policy of every SF (obtained in the first step). An application of this framework to the coordination between CCO and ICIC is presented in [59]. Another reference considering tailing coordination is available in [60], in which the coordination between MLB and MRO is tackled.

In [60], the authors propose to truncate the output range for SFs parameters leading the HO margin towards extreme values to some predefined thresholds. When the SFs produce parameter changes greater than those thresholds, the respective SF output is truncated to the corresponding threshold value. The authors claim that the extreme values of CIO and Hysteresis are obtained in extremely high mobility or load conditions.

In [61], the authors propose a tailing coordination mechanism using a centralized coordinator, which solves conflicting policies between MRO and MLB functions. Every function is modeled as a reinforcement learning agent (specifically actor-critic algorithm). The work in [61] was carried out within another important SON-related project, the FP7 SEMAFOUR project, whose objective was the development of multi-RAT/multi-layer SFs that provide a closed control loop for the configuration, optimization, and failure recovery of the network across different RATs [62].

2.5.4 Heading Coordination

This coordination is achieved mainly through a conflict-free SF design. [63] is a good representative of this category. The authors in [63] propose three design principles to achieve conflict-free design, namely: separation of control (which guarantees independence in the parameters used by a SF), separation of concern (which guarantees independence in the KPIs), and separation of time scales. In order to elaborate on these principles, the authors follow classical control theory approaches. [63] is another deliverable of the FP7 SEMAFOUR project.

Another heading coordination approach is presented in [64], in which the authors start with the identification and classification of potential conflicts that are possible among the main SFs. Based on the classification of all the possible conflicts, the authors propose decision trees for executing MLB and MRO in a conflict-free environment. The decision trees are generated using what the authors call "Trigger-Condition-Action policies". Of course, creating such decision trees demands deep domain expertise to model all the possible conflicts among SFs.

2.5.5 Implicit Coordination

One way to deal with the complexity of self-coordination is to consider an implicit coordination mechanism that allows the operator to define global KPIs and let the system automatically determine the best coordination decision based on the KPIs without human involvement. We consider that this approach represents the highest level of automation as it makes human intervention dispensable.

Some ideas in this direction are available in [65], in which an objective-driven SON coordination method is proposed. The authors propose a centralized coordinator using stochastic modeling of the SFs, and multiattribute utility theory to make decisions in probabilistic settings. Simulation results are presented for the coordination among MRO, MLB, and CCO. The authors claim that this objective-driven coordination allows the operator to define KPI objectives and let the system automatically determine the best coordination decision based on them without human involvement.

Another implicit approach is available in [66], in which the authors propose a "super KPI" expressing overall network status and guaranteeing conflict-free coordination of SFs. The "super KPI" is defined considering the coverage, capacity, and quality dimensions (see Fig. 1.2). The authors present results for a CCO and ANR scenario and consider a regression stage to study network traffic trends and predict future network resource requirements.

Within the SOCRATES project, the authors in [67] study the impact of the joint optimization between MRO and CAC with an implicitly defined metric. Even though CAC is considered more of a RRM management function, as shown in Fig. 2.4, CAC is related to MLB (in the sense that MLB aims at maximizing the available resources in the network and, therefore, maximizing the admission probability). Results in [67] show that joint optimization of CAC and MRO yields better numbers in terms of network performance than individually optimizing both functions. Although the authors in [65, 66, 67] propose the definition of a global KPI or utility function hiding

the complex dynamics of the network and driving the joint optimization procedures, we consider that they are not "fully-implicit" approaches because they require human intervention to some extent.

To the moment of writing this document, only [68] presents what we consider a fully-implicit coordination approach using machine learning and a multi-objective optimization heuristic. Even though some of the references above already use ML to some extent, e.g., reinforcement learning models as feedback controllers or regression techniques for forecasting, only [68] presents a tool that uses ML as a cornerstone within a multi-objective optimization problem. According to the authors, the motivation behind embedding a ML stage into a multi-objective optimization heuristic is that usually, several configurations need to be assessed during an optimization task, and trying them in a real network is not feasible. Nevertheless, if the network dynamics are captured through a ML model (e.g., using regression techniques), it is possible to run the heuristic directly on top of the model in an off-line manner reducing the impact on the network. Therefore, regression analysis shifts the processing complexity to the model creation stage and allows fast performance evaluations when optimization is running. The heuristic used in [68] is genetic algorithms, specifically NSGA-II. The authors study the conflict between MLB and MRO.

ML is a suitable tool to model complex relationships in a mobile network and reduce the processing complexity while optimizing the network (but not while generating the ML models). In that regard, and in order to frame a ML-based self-coordination solution into the 5G network architecture, the 5G-PPP-SELFNET project[13] [69] was recently planned to develop an efficient SON management framework for 5G.

The SELFNET architectural framework considers an automation plane (in compliance with Fig. 2.12) called "SON Autonomic Layer", which is responsible for solving the conflicts among several SFs. This layer is responsible for collecting, aggregating, and analyzing the FCAPS-related information. Thus, any ML model could be onboarded there.

Additionally, in the "SON Autonomic Layer", the operator has visibility across the whole network and resources, and the decision-making is rather centralized. Different studies about the feasibility of this framework are available in [70, 71, 72, 73, 74, 75], although unfortunately not directly related to the coordination of SFs in Section 2.2.

[13] SELFNET: A Framework for Self-Organized Network Management in Virtualized and Software Defined Networks.

2.5.6 Summary

Table 2.2 summarizes the previous references and frames them within the categories above.

Table 2.2: State-of-art Summary

Reference	Category	Technique	Considered SFs	Architecture
[48]	Separation	Hidden Markov Chains	None	decentralized
[51]	Separation	Integer Linear Programming	MRO/ES	decentralized
[49, 50]	Separation	Q-Learning	MRO/MLB	decentralized
[63]	Heading	Control theory	MLB/CCO	centralized
[64]	Heading	Decision Trees	MRO/MLB	centralized
[59]	Tailing	Actor-Critic and Game Theory	ICIC/CCO	decentralized
[60]	Tailing	Extreme values truncation	MRO/MLB	decentralized
[61]	Tailing	Actor-Critic	MRO/MLB	decentralized
[52, 53]	Explicit	Taxi Cab Method	MLB/CCO	centralized
[54]	Explicit	Enhanced Taxi Cab Method	MLB/CCO/ES	centralized
[56]	Explicit	Control Theory	MLB/CCO/CAC	decentralized
[55]	Explicit	Nelder-Mead Algorithm	MLB/ICIC/ES	centralized
[58]	Explicit	Clustering estimation	MLB/ICIC	decentralized
[67]	Partially-Implicit	Directed Search	MRO/CAC	decentralized
[65]	Partially-Implicit	Multiattribute utility theory	MLB/CCO/MRO	centralized
[66]	Partially-Implicit	Decision trees and regression	CCO/ANR	centralized
[68]	Fully-Implicit	SVM and NSGA-II	MRO/MLB	decentralized

We have seen that, as new network functionalities are being introduced in 5G, an increment in the number of network parameters is expected, as well as an increase in the interweaving degree among SFs. Therefore, solving a SF coordination problem in a closed form becomes intractable. SDOs have high hopes for an automation plane boosted by ML models yielding a closed-loop control in which human intervention could be dispensable. In the rest of this document, we pretend to show the benefits, limits, and workarounds regarding a centralized fully-autonomic coordination function.

3 Implicit Coordination Applied to Joint MLB and MRO Optimization

At a high level, the main takeaway from Chapter 2 regarding the coordination among SFs is as follows: the 5G (and beyond) network management should include a fully autonomic layer based on ML that shall boost a closed-loop control to coordinate conflicting network functions, taking advantage of the global visibility of FCAPS data attainable in the Operations Support Systems (OSS).

In that regard, in this chapter we propose a framework that is fully compliant with that idea, and use it to coordinate two SFs, namely MRO (for HO optimization) and MLB (for load balancing). Without coordination, performance degradation is expected due to the cross-dependencies between both SFs. To cope with the underlying dynamics, we propose a zero-touch coordination framework based on ML to automatically learn the interdependencies between the selected SFs and solve the joint network optimization problem.

3.1 Introduction

As mentioned in Section 2.2, SON comprises a set of functions in charge of ruling the behavior of a mobile network based on specific optimization targets and resource constraints. Roughly speaking, every SF takes as input some metrics of the environment (from now on called *environmental* variables) and, according to some policy, tunes network parameters (*controllable* variables) to solve a SF-specific optimization task.

We can think of every SF as a black-box controller aiming to solve specific single-objective optimization problems within the network. As described in Section 2.3.1, two of these black boxes are:

- MLB which is in charge of the balancing of the load across cells, therefore improving the network capacity, and

- MRO, whose goal is to reduce the number of RLFs induced by the mobility of the UEs, and thus improving the QoS.

These SFs have strong interdependencies (as both optimize handover parameters), which can induce conflicts undermining the stable network operation.

The need for dynamic allocation of SFs (e.g., based on the expected user profiles) is evident, especially if we consider two canonical 5G use cases: massive Machine Type Communications (mMTC) and enhanced Mobile BroadBand (eMBB). Because the mobility capabilities for mMTC are expected to be low, MRO function onboarding could be neglected. However, for eMBB, the instantiation of both SF is desired. Therefore, it is expected that SFs to be virtualized and instantiated on-demand (according to slice-specific QoS requirements). Hence the idea of Self-Organized networks Functions as a Service (SFaaS) is entirely plausible. Achieving this degree of flexibility, as well as meeting the condition of a solution being vendor-agnostic, and being adaptable to new network functionalities like slicing, are the main drivers to lean toward ML-driven coordination schemes, like the one proposed in this chapter.

As it was already pointed out in Section 2.3, isolated modeling of SFs is suboptimal, and it results in poor performance and problems with the system stability. Consequently, self-coordination is proposed as a mechanism to guarantee interoperability among multiple SFs and a conflict-free environment. According to Table 2.2, one of the most studied conflicting scenarios is the one involving MLB and MRO, which has references for almost every category of the taxonomy in the table, except for the explicit coordination branch (the reason behind being the lack of an explicit formulation of MRO dynamics[1]).

In [60, 61], the authors use tailing coordination mechanisms for the conflict between MRO and MLB, whereas in [64], a heading coordination technique for the same conflict is presented. As we mentioned in Section 2.5, with tailing coordination the optimality degree achieved by the individual agents is jeopardized during the conciliation stage. Likewise, for heading coordination, deep knowledge and strong assumptions about the modeling process are needed to design conflict-free SFs. Frequently, the expertise for SF modeling is not available right away and the generated models do not capture complex dynamics in real networks.

Regarding the implicit coordination category (see Table 2.2), the authors in [68] propose a distributed coordination scheme for conflict resolution between MRO and MLB based on a multi-objective optimization heuristic supported by a supervised

[1] In Chapter 6 we propose a way to explicitly model MRO within a joint optimization problem.

learning model. The ML model is in charge of predicting the performance for each isolated SF (i.e., without including information of the not-handled SF), whereas the heuristic searches for a set of non-dominated solutions[2]. We claim for the sake of optimality, it is better to predict the KPI at the network level, so dependencies between SFs located in different cells are considered right away and directly. Additionally, we claim that it is also possible to shift the complexity of the optimization step to a ML model to get a simpler optimization task (e.g., as simple as a table lookup).

The authors in [50] propose a separation-based coordination (see Table 2.2) between MRO and MLB. The main idea is to delay the execution of a SF until the effects of another conflicting executing SF function become visible in the system. The separation can be carried out in space or time, allowing the scheduling of the execution of SFs at different spatial-temporal points to minimize negative cross effects among the SFs but with the (intrinsic) limits of this concerning stability and finding the best global solution. In Section 3.3.1.1, we compare the performance of our proposed model with some of these approaches.

3.1.1 Contributions

Since modeling the cross effects between MLB and MRO usually demands expert knowledge which limits the self-organizing properties of the network and makes the network management expensive, we propose and implement a centralized method to jointly optimize SFs (achieving implicit coordination as a by-product of this optimization), taking advantage of the FCAPS information available in the OSS. Even though this method was tailored in this chapter for MLB and MRO interaction, it has also been applied to other conflicts, namely: MLB and CCO coordination in Chapter 4 and MLB and ICIC coordination in Chapter 5. The proposed method is exclusively boosted by ML models, and different groupings of the ML models were designed and implemented.

In this chapter, we show that it is possible to carry out the joint optimization task (yielding implicit coordination among SFs) as long as the SF dynamics can be decomposed into three separable domains: *environmental*, *controllable*, and *utility* parts. We also proposed a ML-based stage to interpolate unknown combinations of the last three domains and show how this helps to speed up the learning process (see Section 3.2.3.1). We were able to translate the convoluted, high-dimensional, and complex coordination problem into a human-readable low-dimensional representation in which

[2]Solutions that can not improve the performance of one SF without degrading the other one.

the optimization task is formulated as a lookup step (see Sections 3.2.3.2 and 3.2.3.3).

A comparison with the state-of-art benchmark was carried out. It was observed an improvement in the performance in a global KPI in steady-state. However, some limits were hit regarding the available data for the optimization as well as scalability issues which will be addressed in Chapters 4 and 5.

3.2 ML-based Joint Optimization

This section elaborates on our idea to jointly optimize SFs using ML. We claim that through this data-driven optimization, we achieve implicit coordination of the studied SFs.

The relevant notation considered throughout this chapter is provided in Table 3.1.

3.2.1 Network Management Model

Let SF_n^i be the n-th SF of the i-th cell. Let $\mathbf{K_n^i}$ be the controllable vector specific to SF_n^i, let $\boldsymbol{K}$ be the network-wide configuration vector, let $\mathbf{E_n^i}$ be the environmental vector impacting the n-th SF for i-th cell, and let $\boldsymbol{E}$ be the network-wide environmental variable vector. Every SF could be modeled as a learning agent locally interacting within one cell, so the whole network can be seen as a multi-agent system, where each learner's decision has an impact on other learners in the same cell (what we call: intra-cell dependencies) and probably on the neighborhood (i.e., inter-cell dependencies) as shown in Fig. 3.1.

Let $\mathrm{E_{dim}}$ be the total number of environmental variables in the network, which means $\boldsymbol{E} \in \mathbb{R}^{\mathrm{E_{dim}}}$. Additionally, let $\mathrm{K_{dim}}$ be the total number of controllable variables in the network[3], which means $\boldsymbol{K} \in \mathbb{R}^{\mathrm{K_{dim}}}$. Finally, let $\mathcal{U} \in \mathbb{R}$ be a global utility aggregating the coordination effect into the whole network performance defined as in Eq. (3.1):

$$\mathcal{U} \coloneqq h(\mathbf{K_1^1}, \mathbf{K_2^1}, ..., \mathbf{E_1^1}, \mathbf{E_2^1}, ..., \mathbf{K_1^2}, \mathbf{K_2^2}, ..., \mathbf{E_1^2}, \mathbf{E_2^2}, ...) \tag{3.1}$$

$h(\cdot)$ in Eq. (3.1) is called the *network management model* and represents the linkage function between $\mathcal{U}$ (dependent variable) and controllable and environmental variables the cell is exposed to (independent variables). $h(\cdot)$ is assumed unknown at the beginning of the optimization process.

[3]Both $\mathrm{E_{dim}}$ and $\mathrm{K_{dim}}$ depend on the number of cells, the number of SFs per cell, and the number of parameters per SF.

Table 3.1: List of variables

Description	**Symbol**
n-th SF of the i-th cell	SF_n^i
Configuration vector specific to SF_n^i	$\mathbf{K_n^i}$
Network-wide configuration vector	$\boldsymbol{K} \in \mathbb{R}^{\mathrm{K_{dim}}}$
Environmental vector impacting SF_n^i	$\mathbf{E_n^i}$
Network-wide environmental variable vector	$\boldsymbol{E} \in \mathbb{R}^{\mathrm{E_{dim}}}$
Total number of environmental variables in the network	$\mathrm{E_{dim}} \in \mathbb{Z}_+$
Total number of controllable variables in the network	$\mathrm{K_{dim}} \in \mathbb{Z}_+$
Global utility function	$\mathcal{U} \in \mathbb{R}$
Network management model	$h(\cdot)$
Change in environmental variables	$\Delta\boldsymbol{E} \in \mathbb{R}^{\mathrm{E_{dim}}}$
The best possible configuration vector	$\boldsymbol{K^*} \in \mathbb{R}^{\mathrm{K_{dim}}}$
Predicted utility value	$\overline{\mathcal{U}} \in \mathbb{R}$
Unknown environmental variable	$\overline{\boldsymbol{E}} \in \mathbb{R}^{\mathrm{E_{dim}}}$
Unknown controllable variable	$\overline{\boldsymbol{K}} \in \mathbb{R}^{\mathrm{K_{dim}}}$
Set of clusters (regardless the unit being clustered)	$\{\boldsymbol{C_k}\}$
The best possible configuration vector for cluster $\boldsymbol{C_k}$	$\boldsymbol{K^*_{C_k}} \in \mathbb{R}^{\mathrm{K_{dim}}}$
Dataset $\mathbf{X} \in \mathbb{R}^{m \times n}$ mapped into a lower dimensional space	$\mathbf{X'} \in \mathbb{R}^{m \times n'}$
Number of users under the service of the i-th cell	$N^i \in \mathbb{Z}_+$
Mean user velocity in the i-th cell	$\overline{V^i} \in \mathbb{R}_+$
HandOver Aggregated Performance (HOAP)	HOAP
Ping-Pong HO rate for i-th cell	PP_i
RLF rate in i-th cell (too-early HO)	RLFE_i
RLF rate in i-th cell (too-late HO)	RLFL_i
Unsatisfied users rate for i-th cell	$N^i_{\mathrm{uu}} \in [0, 1]$
User distribution for i-th cell	$\mathrm{uD}_i \in [0, 1]$
Set of cells	$\mathcal{B} = \{1, ..., B\}$
Set of neighbor cells for i-th cell	$\mathcal{N}(i)$
Reward function for Q-Learning	$r_t \in \mathbb{R}$
Change in the i-th cell utilization	$\Delta\rho_i \in [0, 1]$
i-th cell's transmit power i-th cell	$p_i \in \mathbb{R}_+$
i-th cell utilization	$\rho_i \in [0, 1]$
m-th UE's requested data rate	$\gamma_m \in \mathbb{R}_{++}$
A3-HO Hysteresis	H
A3-HO Time to trigger	TTT
Cell Individual Offset between cells i and j	$\mathrm{CIO_{i,j}}$

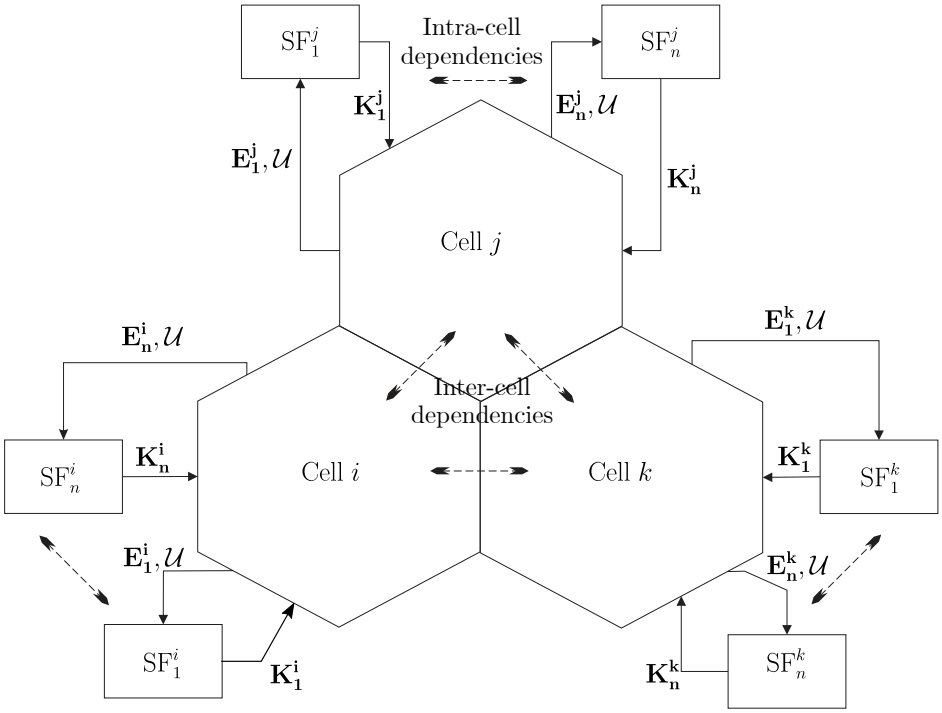


Figure 3.1: Inter- and intra-cell interactions among SFs

Since we have defined a global utility, we let every agent in Figure 3.1 collect data from the environment and send it to the OSS. In a C-SON architecture, we allow intra/inter-cell observability, and we claim that this will let us gain simplicity in the final optimization process. Because global decisions are made, concurrent modifications of parameters take place, and these are explicitly signaled to all the agents[4]. In this manner, the locally-gathered knowledge is exploited to understand (in a centralized way) the dynamics between all agents. Nevertheless, a drawback is expected: the optimization space becomes much larger and more sparsely populated, at least at the beginning of the process, which could yield unsatisfactory solutions (the so-called *cold-start problem*). To cope with this limitation, we propose a ML-based phase for speeding the "learning" up (to some extent), as shown in Section 3.2.3.

3.2.2 Visualization of the Optimization Problem

Regarding the functional decomposition of the optimization problem given in the previous section (namely, the separation between controllable, environmental, and utility planes), the optimization task could be considered as depicted in Fig. 3.2.

Let us assume that the current environmental conditions in the network lie within

[4]Every agent is in charge of the local configuration enforcement.

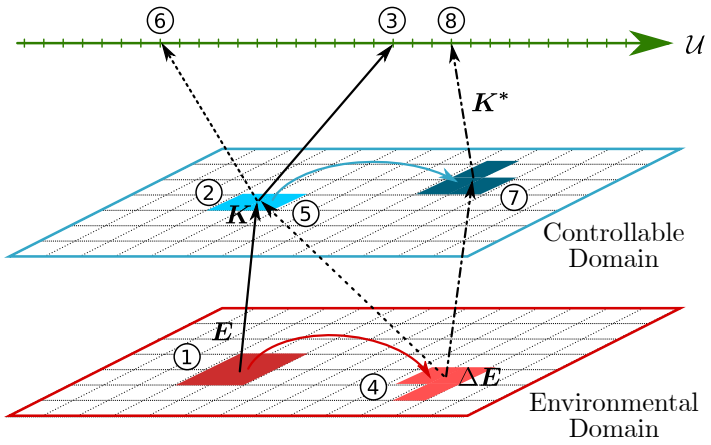


Figure 3.2: Graphical representation of the optimization problem.

the region ① in Fig. 3.2 and the current configuration set is within ② yielding a global $\mathcal{U}$ ③. As the environmental conditions are (to some extent) beyond the operator's control, at some point, a change in the non-controllable conditions could take place ($\Delta\boldsymbol{E}$), shifting the working point to region ④. If the operator keeps the previous configuration ⑤, the general performance might degrade to ⑥. Therefore, the network operator should find the new best ($\boldsymbol{K}^*$) configuration set (represented by ⑦) to improve the KPI the most (⑧) given the current environmental conditions in ④.

It is essential to point out that we propose to find regions or clusters in every plane (instead of specific values), as it is visible in Fig. 3.2, because we assert it is possible to reuse configuration sets for different environmental conditions. Additionally, we consider that different combinations of environmental and controllable variables could result in the same utility values. In Section 3.2.3, we propose an architecture based on the chaining of multiple ML stages to implement the aforementioned optimization approach.

3.2.3 Solution Architecture

The proposed SF's joint optimization architecture is depicted in Fig. 3.3. Each involved stage is described in the following subsections.

3.2.3.1 Step a) Model Completion

We claim that it is feasible to derive the network management model, $h(\cdot)$ using the appropriate regression techniques (and assuming sufficient historical information is available). Once $h(\cdot)$ is estimated with some confidence, it is possible to interpolate unknown combinations to some extent, to reduce the sparsity of the optimization space.

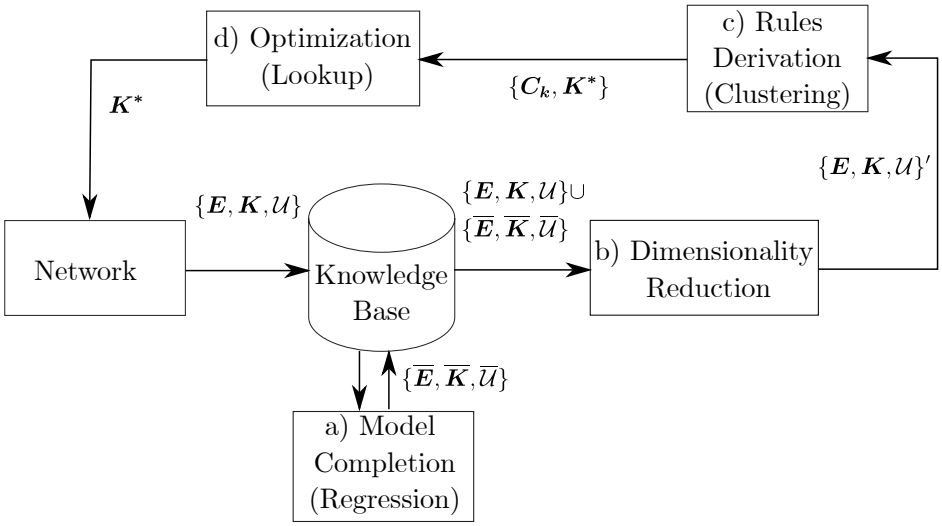


Figure 3.3: Global view of the proposed scheme.

Therefore, we can create an extended knowledge database with the links between dependent and independent variables. That is step a) in our proposed solution, as depicted in Fig. 3.3, in which we try to interpolate $\overline{\mathcal{U}}$ for never-seen-before combinations of $\{\overline{\boldsymbol{E}}, \overline{\boldsymbol{K}}\}$.

An important factor to consider in step a) is that the variance of a regression model[5] is inversely proportional to the difference between the number of independent variables, i.e., $\mathrm{E_{dim}} + \mathrm{K_{dim}}$, and the number of observations/samples represented by m [76]. That means that when $\mathrm{E_{dim}} + \mathrm{K_{dim}}$ is close to m, the variance of the model is pretty high. According to Eq. (3.1), the number of independent variables is expected to be high, so reducing the variance in our regression models is a major challenge to face. We propose to use regularization techniques in some of the models to tackle variance issues. That choice partially guides us through the selection of benchmark models in Section 3.3.1.3.

3.2.3.2 Step b) Dimensionality Reduction

Once we have a realization of the extended network management model ($h(\cdot)$) in a database, a naive optimization approach could be seen as a "lookup" process within it. However, the dimensionality of the structure is pretty high, and the operational cost to do such a scan process is prohibitively high. So we propose to apply dimensional-

[5] In regression techniques, the model's error can be expressed as the sum of three errors: the bias, the variance, and the irreducible error [76]. The bias is the difference between the real value and the expected estimation, so it measures the accuracy of the estimates. On the other hand, variance measures the spread or uncertainty in these estimates. Finally, the irreducible error is due to the noisiness of the data itself.

ity reduction without losing the linkage function between dependent and independent variables as depicted in step b) in Fig. 3.3 (using what we could call a *semi-supervised* dimensionality reduction). With this step, we translate the high-dimensional optimization problem to a low-dimensional representation that is human-readable. This gives us insights into the direction we should move in the low-dimensional space in order to find the best configuration. The remaining question is a suitable step size, which is calculated as explained in Section 3.2.3.3.

3.2.3.3 Step c) (Rules derivation by) Clustering of Similar Environmental States

As it turns out, the "movement" toward the best configuration in the low-dimensional space is constrained by the environmental conditions, in other words, there are several best configurations for different environmental conditions. Therefore, in step c) we cluster the low-dimensional space in an unsupervised manner based exclusively on the environmental conditions ($\boldsymbol{E}$), i.e. we discover a set of clusters $\{\boldsymbol{C_k}\}$.

In this research steps b) and c) in Fig. 3.3 are carried out using grouping methods a.k.a. *ensemble methods*, in which a ML model is built on top of another one to achieve better performance. Two grouping methods are proposed: the first one is the assembly made up of SOM and hierarchical agglomerative clustering as detailed in Appendix A, whereas the second one corresponds to the combination of UMAP and HDBSCAN as described in Appendix B. We elaborate a bit more on these two groupings in Sections 3.2.4 and 3.2.5 (especially the application in mobile networks), because we consider steps b) and c) are the cornerstones in the proposed framework. The details about the models are kept in the Appendix though.

3.2.3.4 Step d) Optimization (Lookup)

In step d) in Fig. 3.3, the framework finds out the current state of the network ($\boldsymbol{E}, \boldsymbol{K}$), which is associated with a specific cluster out of the set of clusters ($\{\boldsymbol{C_k}\}$) from step c), then it selects the best configuration ($\boldsymbol{K^*}$) within that current cluster. The selected configuration is chosen based on the Hamming distance between the current configuration vector ($\boldsymbol{K}$) and the candidates of configuration sets. That means we choose the configuration vector with the lowest number of changes with respect to the current configuration.

As mentioned before, for steps b) and c) in Fig. 3.3, we use two groupings, which are in charge of redefining the optimization problem in a lower-dimensional space. One of them is based on neural network theory whereas the other one is graph-based.

Even though the studied groupings have different characteristics, they have some commonalities: on the one hand-side, they work efficiently for high-dimensional data sets; second, they support "metric learning" (that means adding new points to an existing embedding), and finally both support supervised and semi-supervised dimensionality reduction i.e., labeled data[6] can be used as extra information for dimensionality reduction, as it is explained in detail in Appendix A.

3.2.4 First Grouping: Low-Dimensional SOM Representation and Hierarchical Agglomerative Clustering

As shown in Appendix A, a SOM could have more than one layer of feature representation. We propose to use three layers (one for every domain: controllable, environmental, and utility planes) which are trained progressively, as shown in the Algorithm 7. This arrangement is called Super-Organizing Map [77], and it is shown in Fig. 3.4 along with the way the data corresponding to Eq. (3.1) is presented to the model for training and testing purposes: please notice that the SOM is trained to optimize a score metric (e.g. Root-Mean-Square Error (RMSE), see Section 3.3.2) considering the true utility value for the test set ($\mathcal{U}_{\text{Test}}$) compared with the estimated value based on the controllable and environmental parts ($\overline{\mathcal{U}_{\text{Test}}}$). That corresponds to the dotted lines in Fig. 3.4. In that sense, the SOM acts as a traditional FFNN for prediction.

Additionally, as a product of the arrangement of the input data using the three domains jointly, it is possible to locate the best utility in a specific corner[7] of the utility layer of the SOM, as illustrated on the right-hand side of Fig. 3.4. Therefore, we claim we gain a sense of directionality in the low-dimensional space to optimize the utility.

As it will be shown in Section 3.3.2, once the model is trained, it is possible to get insights about the direction one should move within the grid to optimize $\mathcal{U}$: it is enough to look at the utility layer in the SOM to obtain the best direction. That means we have gained human readability about a complex optimization problem.

After having the dynamics mapped into a lower-dimensional representation (i.e., a trained SOM), we learn high-level/high-value behavioral rules that can be applied in complete sets of network conditions. This way, we ensure the scalability and portability of our "learning" as well as simplicity in the optimization process. To create portable

[6] Labeled data represents samples with information about the KPI.

[7] Depending on the random seed used for training the model, the location of the best corner will change.

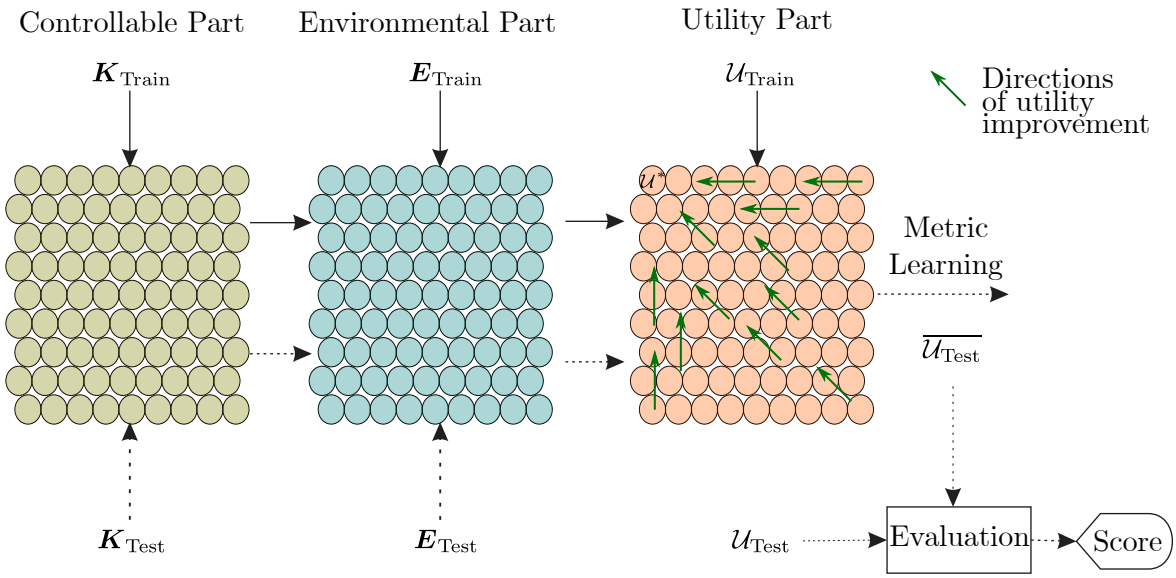


Figure 3.4: Training of a Super-Organizing Map.

models, we allow the reuse of the same controllable parameters for multiple environmental conditions. So, by creating clusters in the environmental space in Fig. 3.4 and projecting them to the other two grids, we find excursion zones where an optimization task is almost as simple as a lookup process. This process is observed in Fig. 3.5.

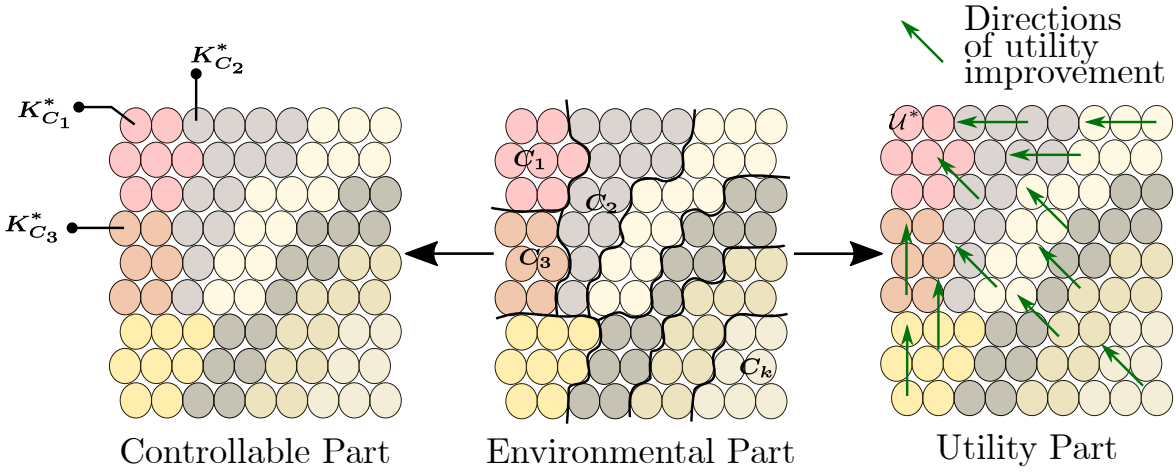


Figure 3.5: Cluster propagation and optimization process in the low-dimensional space.

In the right-hand part of Fig. 3.5, the sense of directionality to optimize the utility is shown. In the central grid of Fig. 3.5, a set of clusters ($\{\boldsymbol{C_k}\}$) is obtained (based on environmental conditions), and those clusters are propagated to the other two grids. The clusters are obtained according to the procedure in Appendix A and are calculated only considering the environmental conditions. The philosophy behind that decision is that we can only move as far as the environment (which is not controlled by the

operator) allows it. Once the clusters have been projected to the controllable layer of the SOM (left-hand side of Fig. 3.5), it is possible to find the set of configurations within the clusters which optimize the utility ($\boldsymbol{K}^{*}_{\boldsymbol{C_k}}$). The best configuration is feedback to the network afterward. It is worth mentioning that this approach is fully compliant with the architecture proposed in Fig. 3.2.

As described in Appendix A, the main hyperparameters for SOM correspond to $\mathrm{x_{dim}}$ (neurons on the horizontal axis) and $\mathrm{y_{dim}}$ (neurons on the vertical axis). To find the best values for these parameters, a grid search is carried out, as shown in Section 3.3.2.

3.2.5 Second Grouping: UMAP and HDBSCAN

For this grouping, the proposed workflow is made up of seven steps as shown in Fig. 3.6 and it is also fully aligned to Fig. 3.2 and the philosophy behind the general framework in Fig. 3.3. In ① we observe information regarding $\boldsymbol{E}$, $\boldsymbol{K}$, and $\mathcal{U}$ from the network. In ②, using UMAP, the observation is embedded into a lower-dimensional space $\mathbf{X}' \coloneqq \left\{x_i'\right\} \in \mathbb{R}^{m \times n'}$ (that is induced by a graph $\mathcal{H}$ as seen in Appendix B). In step ③, the HDBSCAN model determines the cluster for each observation. HDBSCAN can automatically determine the clustering $\{\boldsymbol{C_k}\}$ using the hyperparameter $\mathrm{m_{pts}}$ (as explained in Appendix B). After having a cluster associated with each observation in the low-dimensional space, in ④, ⑤ and ⑥ it is possible to retrieve the best configuration vector within the cluster. To do so, in ⑥ we lookup for the closest environmental condition in terms of euclidean distance, and in ⑦ then the best configuration vector ($\boldsymbol{K}^{*}$) for the current environmental conditions is enforced into the network.

It is important to point out that the ML model hyperparameters (which are correspondingly described in Appendix B) significantly affect the performance of this approach. Therefore a grid search procedure is carried out to find the best values for the parameters of each model, as shown in Section 3.3.2.

So far, we have kept the proposed solution in Fig. 3.3 as general as possible because we claim it could be applicable to any network management problem where the involved variables can be mapped into the three categories we have seen: controllable variables (or tunable parameters), environmental quantities (not under the operator's control), and performance metrics (in line with the operator's policies).

To show the feasibility of our framework, we have applied it to two SFs: MLB and MRO (see Section 2.3.1), which interact with each other through the handover procedure parameters (see Section 2.1.5). The specifics about the joint optimization of these SFs are given in the following section.

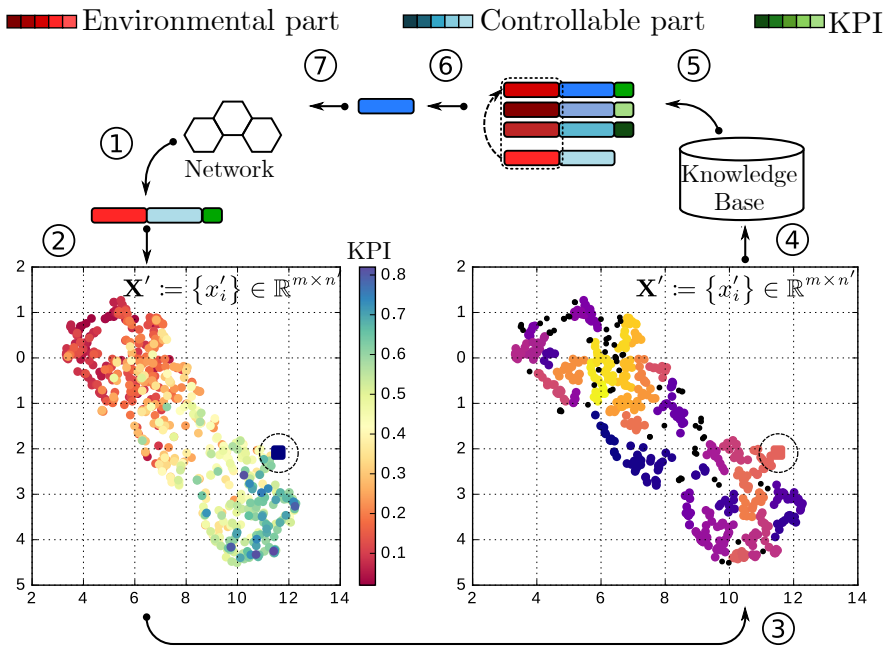


Figure 3.6: Data flow for the UMAP and HDBSCAN grouping.

3.3 Use Case: MLB and MRO

Let us consider a mobile network with B installed cells, represented by the set $\mathcal{B} = \{1, ..., B\}$. The average cell utilization at cell $i \in \mathcal{B}$ is ρ_i, $0 \leq \rho_i \leq 1$, which represents the average fraction of used resources (both in time and frequency), i.e., the average fraction of physical resource blocks (PRBs) utilized. Let N^i be the number of users under the service of the i-th cell. Additionally, let $\overline{V^i}$ be the mean user velocity in the i-th cell.

We consider that, for MRO, the main environmental variable is the mean user velocity ($\overline{V^i}$), the controllable variables are TTT and H, and the main metric is the HandOver Aggregated Performance (HOAP), defined as in (3.2):

$$\text{HOAP} := \frac{\left(w_1 \sum_{i \in \mathcal{B}} \text{PP}_i + w_2 \sum_{i \in \mathcal{B}} \text{RLFE}_i + w_3 \sum_{i \in \mathcal{B}} \text{RLFL}_i\right)}{\sum_{i \in \mathcal{B}} N^i} \tag{3.2}$$

where PP_i is the PP rate for i-th cell. A PP HO is registered when a successful HO from a cell j to another cell i occurs in a time less than the "PP time" after another successful HO had already occurred from i to j. The PP rate measures the rate of HO oscillations per user per second. RLFE_i and RLFL_i components in (3.2) are related to the RLF rate in i-th cell. A RLF occurs if the UE's SINR stays below a threshold for a duration equivalent to the critical time (T310) [29]. The RLFs are differentiated

between those due to HOs being triggered too early (second term in Eq. (3.2)) and those due to HOs being triggered too late (latter term in Eq. (3.2)). We consider that $w_1 + w_2 + w_3 = 1$.

For MLB, the environmental variables considered are: the border's user distribution uD_i, defined as the relationship between the number of users in the i-th cell's border over the total number of users in i-th cell (N^i) and ρ_i. The controllable variables are the CIO values and the main KPI for MLB is the rate of unsatisfied users N^i_{uu} which is the relation of the number of unsatisfaction events in cell/second and the number of users in the cell (N^i) as in Eq. (3.3). One unsatisfaction event occurs when the user's total achieved data rate in the continuous 1-second period is less than the Guaranteed Bit Rate (GBR). Intuitively N^i_{uu} can be seen as the probability of having one unsatisfied user in one cell.

$$N_{\mathrm{uu}} \coloneqq \frac{\sum_{i \in \mathcal{B}} N^i_{\mathrm{uu}}}{\sum_{i \in \mathcal{B}} N^i} \tag{3.3}$$

To explicitly coordinate MLB and MRO, we defined a combined metric as the weighted average of HOAP and N_{uu}:

$$\mathcal{U} \coloneqq w_4 \mathrm{HOAP} + w_5 N_{\mathrm{uu}} \tag{3.4}$$

The KPI in Eq. (3.4) is meant to be minimized. Table 3.2 summarizes the main variables involved in the optimization space.

Table 3.2: Main variables for MLB and MRO coordination

SF	**Environmental Var.**	**Controllable Var.**	**KPI**
MLB	uD_i, ρ_i	$\mathrm{CIO}_{i,j}$	N_{uu}
MRO	$\overline{V^i}$	TTT_i, H_i	HOAP

All the quantities in the second and third columns in Table 3.2 are the regressors in our model trying to explain the combined metric in Eq. (3.4).

From Section 2.1.5 and Table 3.2, it is possible to see that CIO values are defined adjacency-wise rather than cell-wise. Because a cell could have many neighbor cells, it is reasonable to think that CIO values become the worst offender in terms of dimensionality. This situation is a challenge for a fully ML-based optimization approach; thus, in Chapters 4 and 5 we propose an idea to face this issue. Please notice that the KPIs in Table 3.2 are defined network-wide; therefore, it is expected a collaborative

behavior of the cell agents. Finally, it is worth mentioning that the metric in Eq. (3.4) corresponds to the same global KPI defined in Eq. (3.1).

3.3.1 Comparative Evaluation of Algorithms for joint MLB-MRO Optimization

3.3.1.1 Baseline Algorithms

To compare the performance of our framework with state-of-art approaches, we use as a benchmark some of the proposed schemes in [50] where MLB and MRO are modeled as decentralized agents running a model-free Reinforcement Learning algorithm, namely Q-Learning (QL) as explained in Appendix C.

In [50], each SF acts as a control agent that observes the network to evaluate its activation triggers (both environmental and controllable variables), takes an action (exploration process), and gets feedback on the effect of that action at the end of a monitoring period. An action corresponds to the adjustment of any of the controllable variables in Table 3.2. Based on the effect of this action, the Q-learner receives an immediate reward, and the environment undergoes a transition into a new state, see Appendix C. The objective of the learner is to choose actions that maximize the discounted cumulative rewards over time. A greedy implementation of MLB is called QLB, whereas QMRO is a QL version of MRO [50].

- In [50], for QMRO, the reward function in Algorithm 11 is defined as the negative of Eq. (3.2), namely $r_t := -\frac{(w_1 \sum_{i\in\mathcal{B}} \mathrm{PP}_i + w_2 \sum_{i\in\mathcal{B}} \mathrm{RLFE}_i + w_3 \sum_{i\in\mathcal{B}} \mathrm{RLFL}_i)}{\sum_{i\in\mathcal{B}} N^i}$, whereas $\mathcal{S}$ (see Appendix C) is the set of discrete velocities, and $\mathcal{A}$ (see Appendix C) corresponds to the cartesian product of H and TTT. This is fully compliant with Table 3.2.

- In [50], for QLB, the reward is defined considering the impact on the utilization of the overloaded cell, i.e., $\Delta\rho_i$, and the extra load transferred to the neighbor cells i.e., $\Delta\rho_n \forall n \in \mathcal{N}(i)$, with $\mathcal{N}(i)$ representing the set of neighbor cells for i-th cell. $\mathcal{S}$ is defined as the cartesian product of discrete load values for both ρ_i, $\rho_n \forall n \in \mathcal{N}(i)$ and the user distribution uD_i. Regarding $\mathcal{A}$, the change in CIO values is considered. This is again compliant with the quantities in Table 3.2.

- QLH refers to a scheme where both MLB and MRO are working simultaneously without any coordination. It is expected to be the worst-case scenario in terms of network performance due to the underlying conflicts.

To decouple the execution of the agents previously defined in time and space, two separation approaches proposed in [50] are also considered as a benchmark, namely:

- Temporal Separation during Learning (TSL). This scheme is intended to reduce intra-cell conflicts in Fig. 3.1. During the agent learning period, there is a time separation between the SFs. Thereafter all the SFs are simultaneously executed.
- Concurrent learning with Spatial Separation (CSS). To reduce inter-cell conflicts in Fig. 3.1, two conflicting SFs should not take actions in two neighbor cells concurrently. Thus, with CSS, actions are only taken concurrently in two cells, if there is at least 1 other cell in between the two. The result is a reuse-3 concurrency structure among the cells.

3.3.1.2 Simulation Environment and Parameters

Simulation studies were carried out using a C++ event-based LTE downlink system-level simulator based on software libraries provided by Nokia Bell Labs and the University of Stuttgart's Institute of Communication Networks and Computer Engineering [78]. The network scenario (which is the same throughout this document) consists of 7 eNBs, each with 3 cells, where each cell has a system bandwidth BW= 10 [MHz]. The simulator implements a wraparound of the network for reliable interference and SINR calculations, see Fig. 3.7.

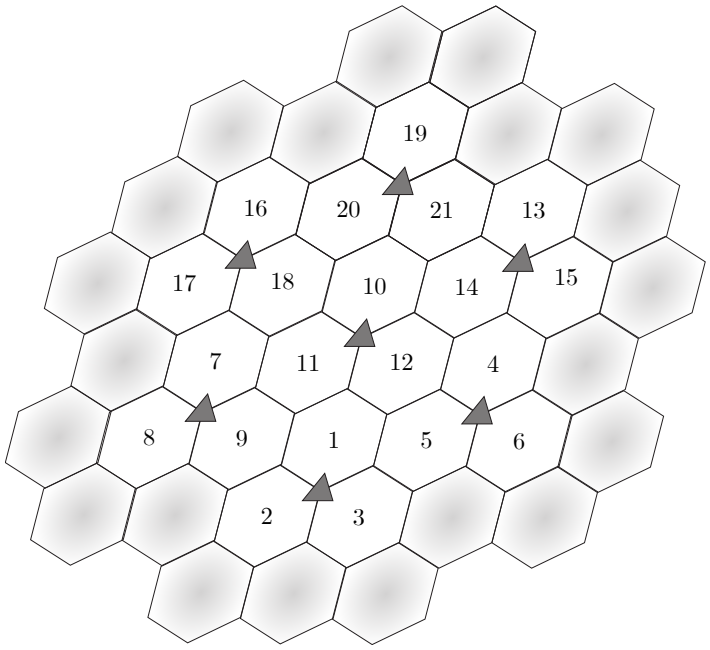


Figure 3.7: Network layout.

Mobile users are randomly placed in the coverage area and move around following a random walk mobility model. We have introduced an asymmetric load distribution via 30 static users in Cell 10. The simulation runs multiple batches, each lasting a specified batch period and within which multiple snapshots are taken. A single snapshot represents the aggregated effects of all the events that would have happened in the network since the last snapshot was taken. At the beginning of each batch, all users are re-introduced into the network following the random deployment model. Further details of the simulation parameters are given in Table 3.3.

Table 3.3: Simulation parameters

Type	**Parameter**	**Value**
System specifics	Inter-site distance	500 [m]
	Pathloss	See Eq. (2.3)
	Shadowing	See Eq. (2.3), Ω = 6[dB]
	eNB Tx power (p_i)	46dBm
	eNB Tx antennas	gain 15 [dBi], height = 32 [m]
Simulation specifics	Batch duration	180 [s]
	Snapshots	Every 10 [ms]
User specifics	Number of UEs	240 mobile, 30 static (cell 10)
	$\overline{V^i}$	10-120 [Km/h]
	Mobility model	Random Walk
	UE positioning	uniform distribution
	UE antennas	gain 2 [dBi], height = 1.5 [m]
	UE data rate (γ_m)	256-1024 [kbps]
Algorithm specifics	PP Time	5 [s]
	T310	0.2 [s]
	TTT	0-5120 [ms], 16 steps defined in [29]
	H	0-15 [dB], steps: 0.5 [dB]
	Weights in (3.2)	w_1 = 0.3 w_2 = 0.2 w_3 = 0.5
	Weights in (3.4)	w_4 = 0.5 w_5 = 0.5

3.3.1.3 Optimization of Hyperparameters

To learn the network management model $h(\cdot)$ in Eq. (3.1) and overcome the "cold-start" issue (to some extent), a supervised learning approach is proposed as in step a) in Fig. 3.3. The idea is to estimate the expected KPI based on the 150 regressors: 21 values per cell for the variables H_i, TTT_i, $\overline{V^i}$, uD_i, ρ_i and 45 values for CIO (one value per adjacency)[8]. Several well-established supervised ML models are compared

[8]That is the number in a rather regular hexagonal grid as the one shown in Fig. 3.7. However, in a real deployment, that number could be higher.

including regression trees (rTree), random forest regressors (rForest), support vector regressor (sVR), deep learning models (dLearning), and models with regularization[9], namely: ridge regression (ridge), lasso regression (lasso), elastic nets (eNet) and partial least squares regression (pLS).

For every model, a grid search was carried out to determine the best value of the associated hyperparameters. A detailed explanation of every model and its hyperparameters is out of the scope of this document. However, if further details are needed, [76] could be a starting point. The performance of the models is shown in Table 3.4.

Table 3.4: Supervised model's performance

Model	**RMSE**	R^2
rTree	0.078	0.625
rForest	0.056	0.812
sVR	0.047	0.863
dLearning	0.010	0.970
ridge	0.050	0.844
lasso	0.051	0.838
eNet	0.050	0.845
pLS	0.049	0.850

The best model is a dLearning structure which exhibits the smallest RMSE and highest correlation factor (R^2), followed by the pLS model. The final neural network architecture (found through a grid search procedure) consists of 4 hidden layers with 150/130/50/21 units respectively, using Rectified Linear Unit (ReLU) activation function and an output layer with one unit and linear activation function. In Fig. 3.8, we plot the predicted ($\overline{\mathcal{U}}$) and actual values of the utility function ($\mathcal{U}$) defined in Eq. (3.4) given by the dLearning model in the test set[10]. We can see that the points are located around the perfect correlation curve (black dashed line), where the real value is exactly as the predicted one, with low dispersion.

It is worth mentioning at this point that during the running time, only the knowledge database in Fig. 3.3 is extended by 10% with respect to the size of the training set using the deep learning model. Of course, continuous retraining of the model is needed as

[9] As it was mentioned in Section 3.2.3.

[10] We have used the traditional cross-validation approach where the training set is used to create a base model, the validation set is used for fine-tuning, and the test set to face the model to never-seen-before data and assess its performance.

more real information is gathered or when the network layout changes. That is done through an external loop that is in charge of validating the integrity of the data, the predictions, and the obsolescence of the trained model. Further considerations regarding this retraining loop are given in Chapter 4.

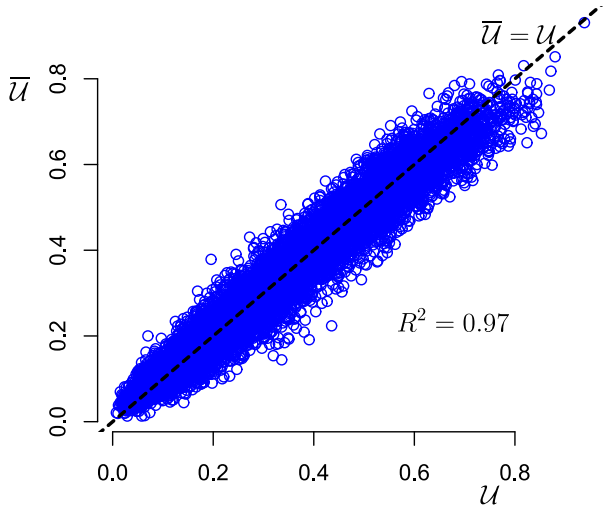


Figure 3.8: Predicted against real values for deep learning regression method.

As we mentioned before, steps b) and c) in Fig. 3.3 are performed using grouping methods. The first one is the combination of UMAP and HDBSCAN (U+H). A grid search process was conducted to find the best values for the main hyperparameters of the grouping. The results are available in Table 3.5. Although it is not possible to plot the low dimensional space (as the best number of dimensions in the reduced space is 5), the optimization process follows the workflow in Fig. 3.6.

Table 3.5: UMAP and HDBSCAN grouping's hyperparameters

Type	**Parameter**	**Value**
UMAP	k (see Appendix B)	10
	n' (see Appendix B)	5
HDBSCAN	minimum cluster size (see Appendix B)	5
	m_{pts} (see Appendix B)	1

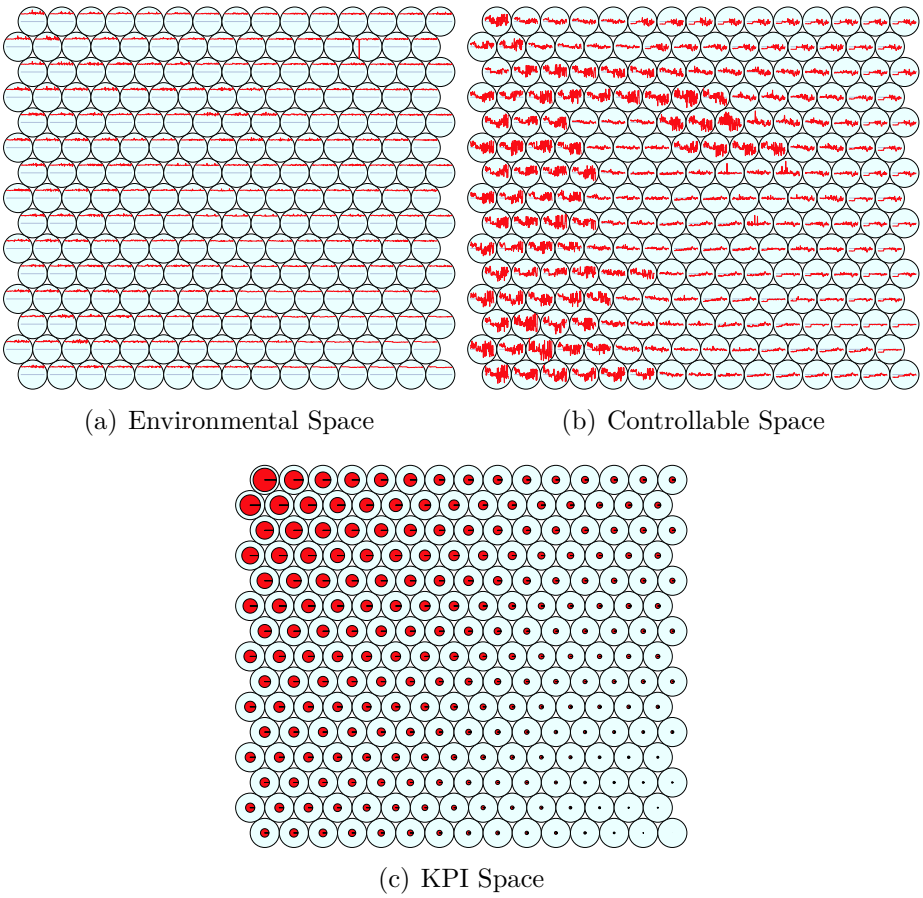
The second grouping is the combination of SOM and hierarchical clustering (SOM+HC). A grid search procedure took place to find the best hyperparameters which are shown in Table 3.6.

Because SOM reduces the dimensionality to 2D, we can obtain the standard side-by-side representation of the three spaces (see Fig. 3.4) to get an idea of what the trained

Table 3.6: SOM and hierarchical clustering hyperparameters

Type	Parameter	Value
SOM	x_{dim} (see Appendix A)	15
	y_{dim} (see Appendix A)	15

grids look like, as shown in Fig. 3.9.



(a) Environmental Space (b) Controllable Space

(c) KPI Space

Figure 3.9: Super-Organizing Map for MLB and MRO dynamics

Every neuron in the map contains a unique "spectrum" revealing the expected variable profile for all the samples collected within that unit. These profiles become the neuron's signature (see Appendix A) and every node can collect new data samples into it as long as the new incoming point has a similar variable's profile.

Looking at the KPI grid in Fig. 3.9(c) we realize that the worst value for the global metric in Eq. (3.4) lies in the top left-hand corner (the bigger the red circle is, the higher the KPI and the lower the network performance). The best value is diagonally

opposite.

By adding a hierarchical clustering stage on top of SOM (as indicated in Appendix A) based only on the environmental variables, namely ρ_i, uD_i, and $\overline{V^i}$, it is possible to find excursion zones where an optimization task is viable to change the configuration parameters (i.e., H_i, TTT_i and $\mathrm{CIO}_{i,j}$) to get close to the bottom and rightmost corner (where the best KPI value lies). It was found using the Elbow method that the number of clusters is 8 ($|\boldsymbol{C_k}| = 8$).

In Fig. 3.10, 8 clusters are depicted on top of the KPI grid. Within every excursion zone, all the nodes have quite similar values in the environmental variables but different values in the controllable quantities, so starting from an initial operating point, it is possible a parameter's fine-tuning process trying to move the operating point toward the bottom rightmost corner inside every excursion zone as it was explained in Fig. 3.5, where one initial operating point is shown and by changing the parameter accordingly ($\Delta\boldsymbol{K}$), it is possible to reach the best possible value of the indicator. Inter-zone changes are only possible if the environmental conditions change (which is not under the network operator's control).

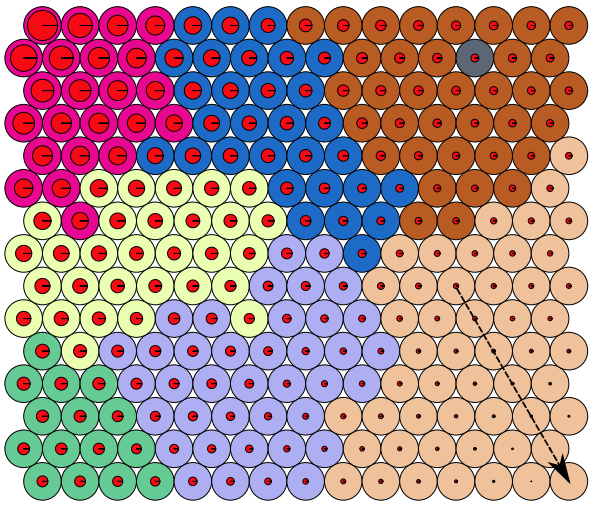


Figure 3.10: Intra-cluster optimization.

3.3.2 Comparative Analysis

Performance comparison among our proposed framework (with the above grouping models) and the aforementioned benchmark schemes was conducted at a low-speed scenario where $\overline{V^i}$ is randomly distributed around 30 [km/h]. In Fig. 3.11, it is shown the evolution over time for the HOAP (after an exponential smoothing process) for

the considered models. It is evident that after a long exploration phase, QMRO learns the best policy, whereas QLH or QLB can not reach the same optimality degree. On the other side, SOM+HC outperforms the benchmark from the very beginning of the simulation. The worst performance is achieved with the QLH and QLB algorithms.

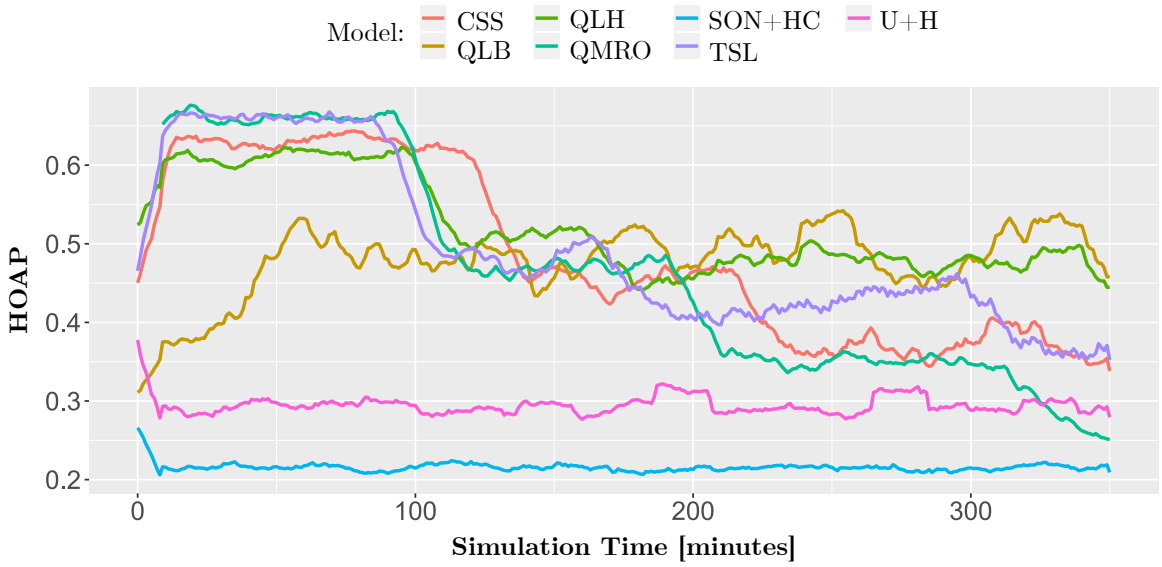


Figure 3.11: HOAP evolution after removal of high-frequency components.

The rate of unsatisfied users (N_{uu}) is shown in Fig. 3.12. For this low-speed scenario, QLB quickly reaches optimality (it highly depends on the users' speeds). The mean performance of SOM+HC is relatively stable across the simulation, although, it is not as good as the one obtained with QLB. The worst performance is achieved with the QLH and QMRO algorithms. The best performance is obtained with QLB and U+H.

The evolution of the global KPI defined in (3.1) is depicted in Fig. 3.13. It is evident that the two proposed grouping schemes outperform the baseline, being the SOM+HC-based approach the best one. Due to this fact, we stick to the SOM+HC grouping for future performance comparisons in the rest of this document.

Due to the different nature of the Reinforcement Learning (RL)-based models and semi-supervised approaches, like SOM+HC or U+H, a comparison in steady-state should be made for the sake of fairness. This comparison is depicted in Fig. 3.14. The figure shows that the worst performance is achieved when no coordination is present (QLH) and the best ones using U+H and SOM+HC. U+H provides good results in terms of unsatisfied users, whereas SOM+HC produces the best results in terms of HOAP. Therefore, we could further state that U+H is suitable for scenarios where mobility is expected to be low (e.g., mMTC), and SOM+HC could be used for

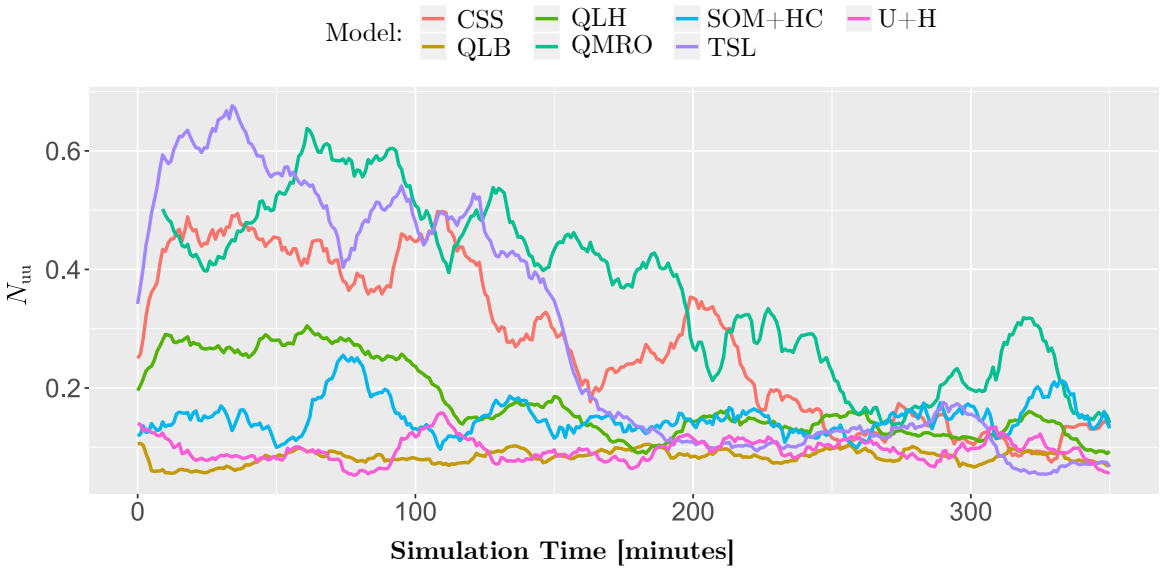


Figure 3.12: Smoothed rate of unsatisfied users profile.

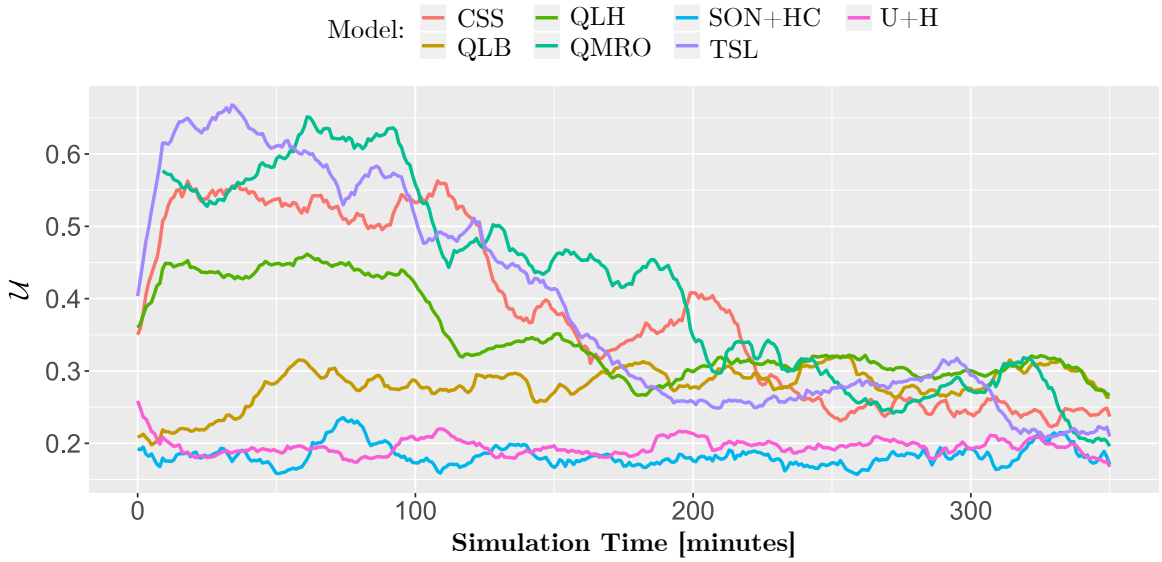


Figure 3.13: Global KPI comparison.

scenarios where mobility can not be neglected, providing the best compromise between both MLB and MRO.

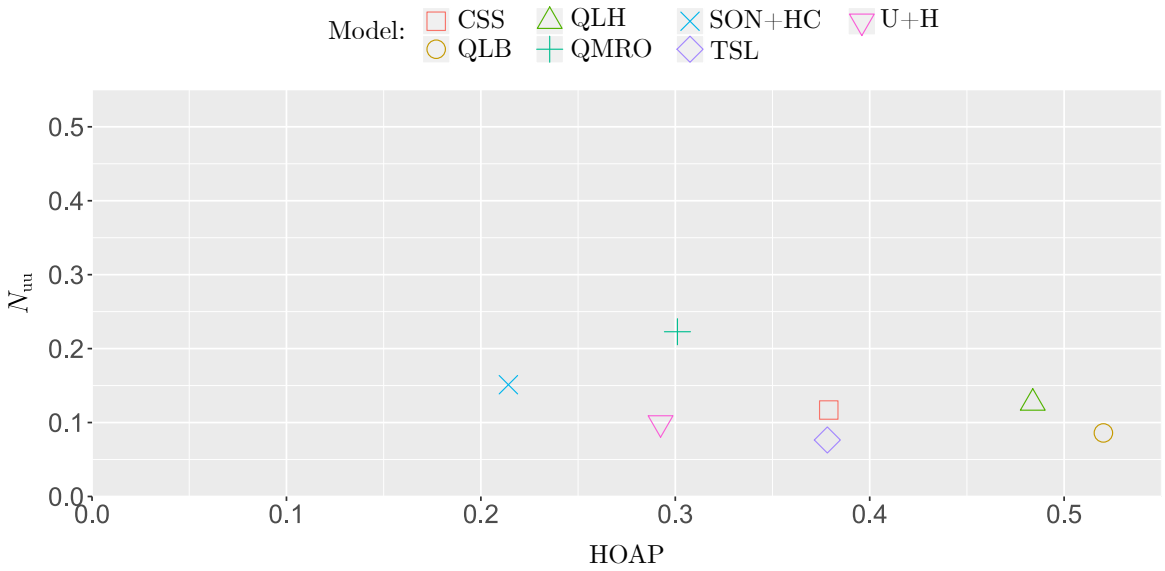


Figure 3.14: Average steady-state performance.

3.4 Summary

In this chapter, we have shown the applicability of ML techniques to get a higher degree of automation in mobile network management, namely the coordination between SFs. Although our approach was applied to the conflict between MLB and MRO, we claim it can be used for several SFs as long as a functional decomposition of the SFs into environmental, controllable, and metric variables is possible.

Two main challenges were faced: the time needed to learn the network management model and the dimensionality explosion. To cope with the first problem, a supervised learning approach based on regression techniques was followed to speed up the "learning" process and interpolate unseen combinations among dependent and independent variables. It turned out that deep learning models were the most suitable choices for this task. These estimators shall be retrained every time a new SF is instantiated, or a cell is deployed. Therefore a continuous outer loop (or even more depending on the nature of the ML models) for the ML model's integrity control should be present (this is fully compliant with [79]).

To face the high dimensionality of the optimization problem, two grouping methods were proposed to translate the optimization task to a low-dimensional space, one

based on the bagging of UMAP and HDBSCAN while the other one relies on the combination of SOM and hierarchical clustering. With both groupings, we show that it is possible to reduce the dimensionality of the optimization problem without losing the linkage between the combined dependent and independent variables, which means the underlying model of the dynamics between SFs was kept.

After defining excursion zones within the low dimensional space, based only on the environmental conditions, it was possible to reformulate the optimization problem as a simple lookup process where the current cell's working point is moved towards the best "corner" inside every excursion zone based on configuration adjustments.

Even though our C-SON approach has outperformed the selected benchmarks in this chapter, some limitations were observed with our architecture proposed in Fig. 3.3. Some motivations to adapt the ideas presented in this chapter are as follows:

- *Curse of dimensionality.* This is a well-known phenomenon in statistical decision theory, which has multiple manifestations; three of them are presented below to illustrate how obtaining regression models[11] capable of generalizing to new samples becomes exponentially more difficult when working with high-dimensional data [80, 81], as there is no notion of "closeness" as the dimensionality increases (which is somehow counterintuitive):

    1. If we randomly pick points[12] within the 2D-unit square, the chance of this point being located less than $\epsilon = 0.001$ from any border (i.e., being an "extreme" point along any dimension) is $(2\epsilon)^2 = 0.004$. For a higher number of dimensions (d), the probability becomes $(2\epsilon)^d$. But in a 10000-dimensional unit *hypercube*, the same probability is greater than 0.99999999. That means, that most points in a high-dimensional hypercube are close to the borders [76] and the interior is "empty".
    2. If now we pick two points from the same 2D-unit square, the mean distance between them will be around 0.52, in a 3D-unit cube, it will be 0.66, and in a 10000-dimensional unit *hypercube*, it will be about 408.25: there is plenty of space in higher dimensions [76]. So the distance between two randomly drawn data points increases drastically with their dimensionality.
    3. As the number of features or regressors grows (remember that in this chapter we used 150), the amount of data a model needs to generalize accurately

[11] Particularly those models that assume that similar regressors yield similar KPIs.

[12] Following a uniform distribution because (for instance) there is no prior knowledge of whether the optimization space has a low-intrinsic dimensionality, i.e., lies in a low-dimensional subspace or low-dimensional manifold.

grows exponentially. For d dimensions and v zones of interest of the global KPI, we seem to need $O(v^d)$ samples [80].

As a result, high-dimensional problems come with the risk of having very sparse optimization space: most training instances are likely to be far from each other. This also means that a new instance will likely be far away from any training instance, making predictions much less reliable than in lower dimensions as they will be based on much larger interpolations [76]. In short, the more dimensions the training set has, the greater the risk of overfitting it.

In theory, one solution to the curse of dimensionality could be to increase the size of the training set to reach a sufficient density of training instances. Unfortunately, in practice, it is quite expensive to get labeled data in a real scenario. Another option is to implement a prior stage of dimensionality reduction; however, for the sake of preserving the functional division among the controllable, environmental, and utility planes, which is the main idea behind our proposed framework, we do not consider this option. Another option is to engineer the features beforehand, e.g., do we need to modify both TTT and H for MRO since the final objective is to advance or delay a HO, and we can do that with only one?

As it turned out, the worst offenders in terms of dimensionality are the CIO values because, according to our definition in Section 2.1.5, they are adjacency-wise defined parameters rather than cell-wise defined. Therefore, it makes sense to deal with MLB (and therefore with CIOs) in a different manner. The way how we face this challenge is elaborated in Chapter 4.

- Regardless of the choices for the grouping method in steps b) and c) in Fig. 3.3, from a high-level perspective, we are chaining three ML "boxes": a regression-related task, a dimensionality reduction model, and a clustering stage. Every model must be retrained as long as more information is being gathered from the network or as soon as the network layout changes, so additional loops for updating the ML models must be implemented, which could make the optimization process complex, and this time is due to an extra-burden which is not related to the mobile network optimization whatsoever.

- As we are connecting "black boxes" with no reliability metric apart from the ones given by the individual ML models and the network KPIs, there is a lack of explainability of how the errors propagate from stage to stage in the data

pipeline.

- The approach presented in this chapter could be considered a best-effort method in the sense that we go as far as the collected data allow us. However, there is a lack of optimality understanding in the sense that there are no insights about how far a recommended configuration is from a global optimal (provided it exists).

Based on the given insights and learnings above, we elaborate a hybrid approach that tries to solve some pieces of the coordination problem in a closed form, leaving the rest of the coordination problem to be solved using a ML stage in one shot, as it will be seen in Chapters 4 and 5 for different combinations of SFs.

4 Coordination between CCO and MLB: A Hybrid Approach

This chapter presents an explicit formulation of the joint optimization problem when MLB and CCO are instantiated in a mobile network, as well as a hybrid approach to deal with the underlying optimization task: on the one hand side, an User Association (UA) stage works in a closed form to balance the load in the network and, on the other side, a ML stage is included to boost the optimization of the network coverage (hence the hybrid denomination). Both stages are connected through a global utility, like in the previous chapter, which guarantees the modeling of the compromise between both SFs.

The relevant notation considered throughout this chapter is provided in Table 4.1.

4.1 Introduction

As mentioned before, with the advent of 5G, network lifecycle operations such as initial service deployment, configuration management, network optimization, and self-healing shall be fully automated processes to reduce CAPEX and OPEX and also to allow new players such as campus networks owners, to come into the scene as non-traditional network operators. To this end, SFs were proposed as a first attempt to provide self-adaptation capabilities to mobile networks on different fronts and to reduce the error-prone human intervention. Nevertheless, deploying multiple optimization functions in a network brings demanding challenges in terms of conflicting objectives in coordination. Automatically coordinating all those functions is paramount for network operators or even industry owners in campus networks (as they usually do not have the expertise to carry out network optimization in an agile manner).

Typically, each SF aims at individual goals by modifying coupled network parameters, generally in dissonant directions concerning other SFs, jeopardizing the global stability of the system. A common conflicting scenario between CCO and MLB was already described in Section 2.3.4.

Table 4.1: List of variables

Description	**Symbol**
x-th ML retrain loop	$\mathcal{L}^{x}_{\text{retrain}}$
Network-wide configuration vector	$\boldsymbol{K} \in \mathbb{R}^{\text{K}_{\text{dim}}}$
Network-wide environmental variable vector	$\boldsymbol{E} \in \mathbb{R}^{\text{E}_{\text{dim}}}$
α-fairness cost function	Φ_α
Global utility function	$\mathcal{U} \in \mathbb{R}$
Geometric Mean of Available Resources	GMAR
The best possible configuration vector	$\boldsymbol{K^*} \in \mathbb{R}^{\text{K}_{\text{dim}}}$
User distribution for i-th cell	$\text{uD}_i \in [0, 1]$
Set of cells	$\mathcal{B} = \{1, ..., B\}$
i-th cell's transmit power	$p_i \in \mathbb{R}_+$
Transmit power vector	$\boldsymbol{p} \in \mathbb{R}^B_{++}$
i-th cell utilization	$\rho_i \in (0, 1]$
Cell utilization mapping	$T : \mathbb{R}^B_+ \to \mathbb{R}^B_{++}$
Cell utilization vector	$\boldsymbol{\rho} \in (0, 1]^B$
Link budget from i-th cell to m-th UE	$g_{i,m}(\cdot)$
SINR from i-th cell to m-th UE	$\text{SINR}_{i,m}(\cdot)$
Base station's Bandwidth	BW
Noise Power measured over BW	$\sigma^2 \in \mathbb{R}_+$
Shannon capacity between i-th cell to m-th UE	$c_{i,m}(\cdot)$
m-th UE's requested data rate	$\gamma_m \in \mathbb{R}_{++}$
UE's requested data rate vector	$\boldsymbol{\gamma} \in \mathbb{R}^S_{++}$
i-th cell's user association	$\mathcal{P}_i$
Set of user associations	$\mathcal{P} \coloneqq \{\mathcal{P}_1, ..., \mathcal{P}_i, ..., \mathcal{P}_B\}$
Optimal user associations	$\mathcal{P}^*$
i-th cell's antenna downtilt	$e_i \in \mathbb{D}_e$
Antenna downtilt vector	$\boldsymbol{e} \in \mathbb{D}^B_e$
The best possible antenna downtilt vector	$\boldsymbol{e}^* \in \mathbb{D}^B_e$
Coverage degree	$C(\boldsymbol{e}) \in \mathbb{R}_+$
Signal strength threshold	$\xi_m \in \mathbb{R}_+$
A3-HO Hysteresis	H
A3-HO Time to trigger	TTT
Cell Individual Offset between cells i and j	$\text{CIO}_{\text{i,j}}$
The best possible Cell Individual Offset values	$\mathbf{CIO^*} \in \mathbb{R}^{B \times B}$
Optimal cell for m-th UE	k^*
Current cell for m-th UE	$\overline{k}$
Mismatch between current and optimal cells	$D_i(\mathbf{CIO})$
Maximum CIO value	CIO_{max}
CIO step value	Δ

Our drivers to enhance the proposed optimization framework in this chapter are three-fold, and they are aligned with what was pointed out in Section 3.4:

- first, to minimize the burden of retraining a full chain of ML models like the one presented in Fig. 3.3,
- second, we want to tackle the high dimensionality imposed by the CIO values (related to the typical MLB implementations),
- and lastly, we want to acquire insights into the optimality of the proposed approach for one or both SFs.

To understand the implications of the first issue, we depict in Fig. 4.1 the expected retraining loops for the ML models in dashed lines. Please notice that Fig. 4.1 is an enhanced version of Fig. 3.3. The retraining loops are not necessarily synchronous and are triggered once:

- there is a change in the layout of the network,
- there are new SFs are included in the operation,
- there is more data collected to the network (which will adjust the models), or
- there is a lack of accuracy in the model's predictions.

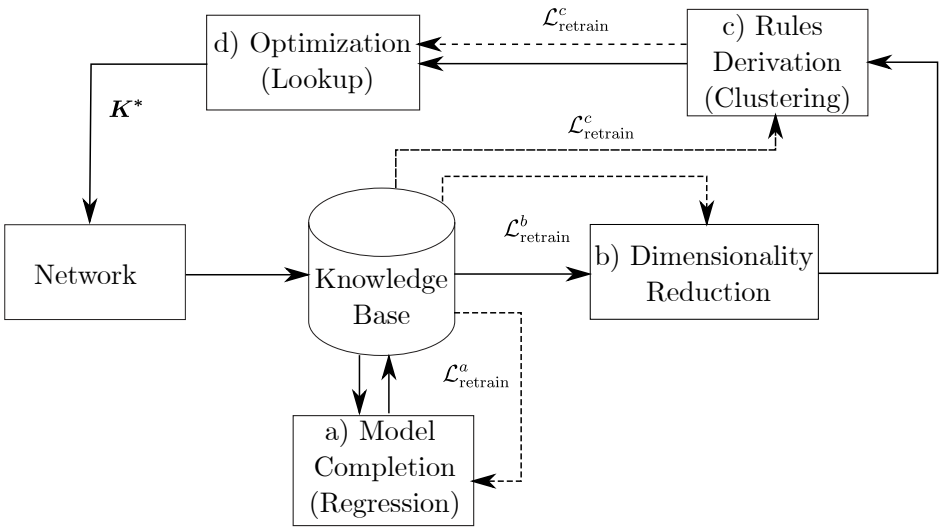


Figure 4.1: Global view of the proposed scheme with ML retraining loops.

We identify three retraining loops that enable changes in the four steps in Fig. 4.1. Please notice that a change in step c) immediately triggers an update in the rules

lookup in step d), therefore we claim that both are controlled by a single re-training loop, namely: $\mathcal{L}^{c}_{\text{retrain}}$. Considering this, we propose to use a single ML Stage, and therefore, a single re-training loop, as shown in Fig. 4.2, although the details for the training process are supplied later in this chapter. This ML stage is in charge of predicting the best possible downtilt vector ($\boldsymbol{e}^*$) as the high-dimensional MLB-related parameters are optimally calculated in a closed form (as will be explained in Sections 4.3.1 and 4.3.2).

Regarding the second motivation from the list above (i.e., the high dimensionality imposed by the CIO values), we claim that the dynamics of the network are ruled by the so-called User Association, UA, (which is governed by several network procedures, depending on the state of the UEs, e.g., initial cell selection, cell re-selection, and handover), because changing the relationship among users and cells affects the interference levels, the load distribution, the HO trigger conditions, etc. As the UA can be controlled using the CIO values in the A3-HO condition in Eq. (2.9), and that condition is a linear inequality, UA fits as a suitable candidate to be solved in a closed form. Accordingly, we aim to find an optimal UA ($\mathcal{P}^*$), which shall be aware of the dynamics between MLB and CCO. From $\mathcal{P}^*$ we intend to obtain the best CIO values, i.e., $\mathbf{CIO}^*$. This is achieved using the shadowed boxes in Fig. 4.2.

As mentioned before, the best antenna tilt vector ($\boldsymbol{e}^*$) is predicted using the ML stage in the outer loop in Fig. 4.2.

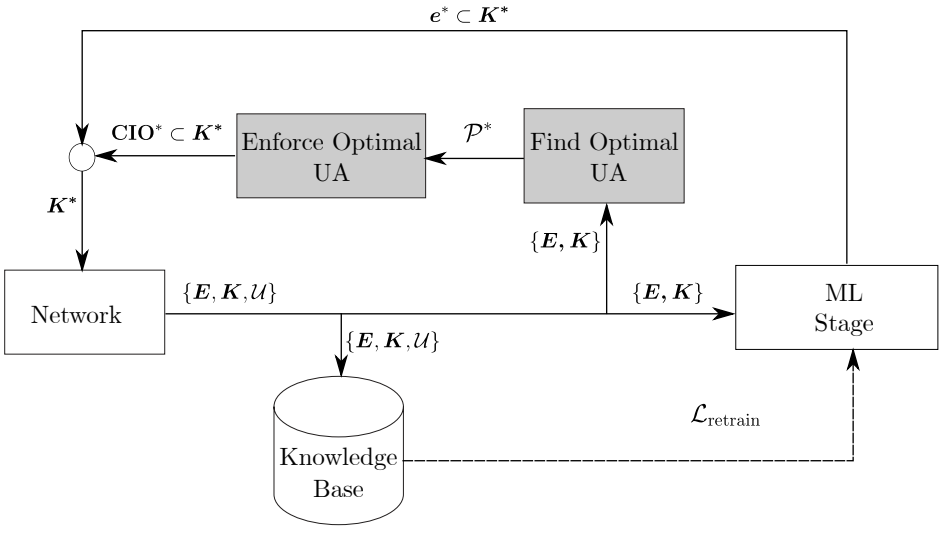


Figure 4.2: Global view of the proposed scheme.

The two loops can run asynchronously, and we claim the decoupling is compliant with the different nature of the SFs, in the sense that, for MLB it is expected to have a faster timescale of operation than CCO [19].

From Fig. 4.2, a part of the best possible configuration is obtained using a ML-based outer loop, whereas the other part of $\boldsymbol{K}^*$ is obtained through the inner loop in a closed-form manner. Therefore, we refer to this framework as a hybrid approach.

A first attempt to harmonize the simultaneous operation of UA and MLB through an explicit optimization problem definition is presented in [82]. However, the solution was tailored to IEEE 802.16m technology rather than LTE/5G. Additionally, the authors assumed that base stations periodically shared their average loads with the UEs, and UEs used that information to make decisions over time. Nevertheless, this assumption is hardly met in real 3GPP-compliant network deployments. In [83], an extension of the ideas in [82], considering cache-enabled network optimization is presented, but the solution is meant to jointly optimize content placement and UA rather than improve the network coverage or capacity. The authors in [53] propose a combination between fixed-point iterations and directed search mechanisms to deal with the coordination between MLB and CCO. We build on top of [53] by adding a machine learning stage to tackle the high-dimensional problem we observed in Chapter 3, as well as to take advantage of the inter-cell dynamics among SF to boost the network performance.

4.1.1 Contributions

In this chapter, we deal with the limitations elaborated in Section 3.4. We propose improving the architecture in Fig. 4.1 by reducing the number of ML retraining loops. The new scheme is depicted in Fig. 4.2 and only contains a single training loop ($\mathcal{L}_{\text{retrain}}$), which will be studied in detail in Section 4.3. The motivation of the inner loop in Fig. 4.2 is two-fold: on one side, it **optimally** solves the load balancing problem in a closed form (provided the constraints in the CIO values, given by the 3GPP, are met), and on the other hand side, we mitigate the problem's dimensionality imposed by the CIO values. Therefore, we claim the hybrid approach is **more data-efficient** than the approach proposed in the previous chapter, meaning that, with the same training set size, the performance is better for the solution in Fig. 4.2. On that note, a comparison is carried out through simulation studies to prove that point.

Additionally, a flexible fairness measure is introduced, that is fully aligned with the minimalist model of the Round Robin scheduler we introduced in Section 2.1.4. This objective function allows a variety of objectives, from maximizing available resources, minimizing delay, or even equalizing the cell utilization across the network. It only takes to modify a parameter in the objective function, namely α.

Furthermore, it is proposed a mechanism to enforce a rather theoretical concept as

the user association ($\mathcal{P}$) into real 3GPP-compliant networks through the A3-HO event (this corresponds to the second shaded box in the inner loop in Fig. 4.2).

Finally, the outer loop in Fig. 4.2, in charge of predicting the best possible antenna downtilt, is proposed as a **multiouput learning problem**, carefully studied throughout all the taxonomy branches available in the literature. Additionally, a deep neural network is proposed to tackle the antenna downtilt prediction. Throughout the rest of this document, we will be interested in landing in the neural network domain for two reasons: the first one is related to the robustness against the curse of dimensionality, and the second one is regarding state-of-art techniques to deal with the cold-start problem.

4.2 Formulation of the Centralized Joint Optimization Problem

In this section, we formulate a joint optimization problem of a network where MLB and CCO are running concurrently. We start selecting a suitable fairness measure and draw some connections with the basic scheduler model presented in Section 2.1.4. Afterward, we explain how the main variables involved in the optimization problem are estimated, specifically the cell utilization and the coverage degree. Subsequently, a joint optimization problem is formally proposed.

4.2.1 A Fairness Measure for Centralized Decision-Making and Conflict Resolution

To select a suitable fairness measure, fully compliant with the scheduler model presented in Section 2.1.4, the resource allocation theory that involves multiple players (base stations) and a central decision-maker (C-SON) is visited [84, 85]. To capture a compromise between fairness to UEs and utilization of resources in the network, we consider the α-fairness function as the objective function to be minimized, and this function is defined by [84]:

$$\Phi_\alpha(\boldsymbol{\rho}) = \begin{cases} \sum_{i=1}^{B} \frac{(1-\rho_i)^{1-\alpha}}{\alpha-1} & \alpha \geq 0, \alpha \neq 1 \\ \sum_{i=1}^{B} -\log(1-\rho_i) & \alpha = 1 \end{cases} \tag{4.1}$$

Out of the available allocation options in the literature (e.g., social welfare, Nash, Kalai–Smorodinsky, or Max–min schemes), the α-fairness was selected since it is the

most versatile and flexible (see [84]) in the sense that Eq. (4.1) supports a family of optimization objectives as α ranges from 0 to ∞, including the following:

- Minimizing Φ_0 reduces to the minimization of the sum of cell utilization. When $\alpha = 0$, the minimization of Eq. (4.1) is equivalent to the maximization of the summation of $(1 - \rho_i)$. As we saw in Section 2.1.4, those terms correspond to the available resources in the network.
- Minimizing Φ_1 allows for maximizing the Geometric Mean of Available Resources (GMAR), defined as GMAR $\coloneqq 10^{-\Phi_1/|B|}$.
- Minimizing Φ_2 corresponds to minimizing the overall average delay. To elaborate on this idea, please consider that $\Phi_2 = \sum_{i=1}^{B} \frac{1}{(1-\rho_i)} = \sum_{i=1}^{B} \frac{\rho_i}{(1-\rho_i)} + 1$. Therefore, for the sake of minimization of Φ_2, it is enough to minimize the first part of the latter summation, which turns out to be (according to Section 2.1.4) the delay due to the queueing system of the scheduler, as expressed in Eq. (2.7).
- As α continues increasing, equalization in the cell utilization is observed, as shown in Fig. 4.3(a) and 4.3(b), in which the cost function in Eq. (4.1) is shown for two cells for the cases of $\alpha = 2$ and $\alpha = 10$ respectively. It is evident that for higher α values, the cells tend to have the same utilization.

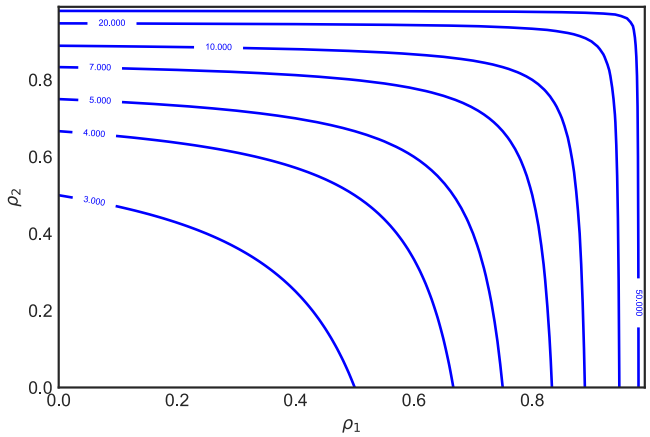
From Eq. (4.1), it is evident that an accurate estimation of the cell utilization is essential. Therefore, in the following section, we use a well-established method for cell utilization estimation [86, 87, 53], which is compliant with the system model proposed in Section 2.1. Additionally, we propose a method to account for the network coverage (for CCO purposes).

4.2.2 Cell Utilization and Coverage Estimation

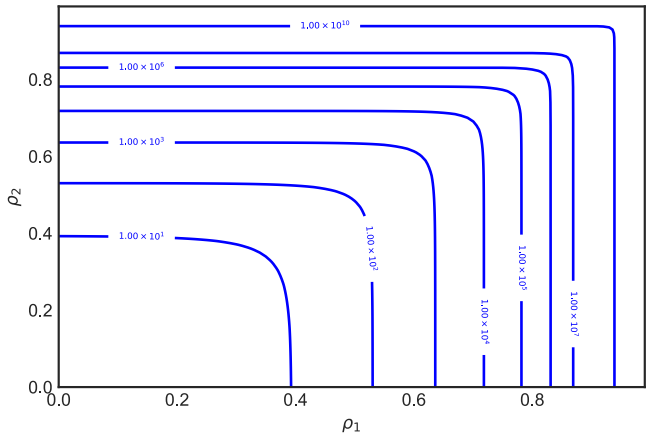
Let $\boldsymbol{e}$ represent the antenna downtilt vector (in degrees), where the i-th coordinate e_i is the tilt of cell i, and it takes values from a discrete set $\mathbb{D}_e$. In this chapter, we make the propagation loss dependent on the antenna downtilt, which is a reasonable assumption as changing the orientation of the cell radiation pattern alters the loss for m-th UE with respect to cell i. Therefore, given the antenna tilt e_i, the propagation loss between cell i and an UE m is denoted by $g_{i,m}(e_i)$.

Replacing Eq. (2.2) into Eq. (2.4), we can write the Shannon capacity as Eq. (4.2).

$$c_{i,m}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P}) = \mathrm{BW} \log_2 \left(1 + \frac{p_i g_{i,m}(e_i)}{\sum_{k \neq i}^{B} p_k g_{k,m}(e_k) \rho_k + \sigma^2} \right) \tag{4.2}$$



(a) $\alpha = 2$



(b) $\alpha = 10$

Figure 4.3: Utilization equalization in a network with two cells.

For an arbitrary (but fixed) power allocation $\boldsymbol{p}$, downtilt vector $\boldsymbol{e}$, data rate requirements vector $\boldsymbol{\gamma}$, and user association $\mathcal{P}$, the cell utilization is estimated by solving the equation system in Eq. (2.5), which is revised in Eq. (4.3):

$$(\forall i \in \mathcal{B})\ \rho_i = \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P})}, \tag{4.3}$$

We are interested in estimating the impact of multiple user associations ($\mathcal{P}$) on the cell utilization ($\boldsymbol{\rho}$) without actually applying any change in the network. That means we are interested in solving Eq. (4.3). The equation system in Eq. (4.3) represents a set of nonlinear equations, which is implicitly defined because the cell utilization ρ_i is on both sides of the equation. The conventional way to solve such an equation system is through the use of Fixed Point Iteration (FPI) algorithms, like the standard one presented in Eq. (4.4):

$$\boldsymbol{\rho}^{(k+1)} = T(\boldsymbol{\rho}^{(k)}),\ \boldsymbol{\rho}^{(1)} \in \mathbb{R}_+^B \tag{4.4}$$

where T is a vector-valued mapping given by:

$$\begin{aligned} T &: \mathbb{R}_+^B \to \mathbb{R}_{++}^B \\ \boldsymbol{\rho} &\mapsto \begin{bmatrix} \sum_{l \in \mathcal{P}_1} \frac{\gamma_l}{c_{1,l}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P})} \\ \sum_{l \in \mathcal{P}_2} \frac{\gamma_l}{c_{2,l}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P})} \\ \vdots \\ \sum_{l \in \mathcal{P}_B} \frac{\gamma_l}{c_{B,l}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P})} \end{bmatrix} \end{aligned} \tag{4.5}$$

A solution to Eq. (4.5) is called a Fixed Point of the mapping T, and it is expressed as $\boldsymbol{\rho} \in \mathrm{Fix}(T)$.

Apart from estimating the cell utilization, we are also interested in estimating the network coverage. Therefore, to account for the coverage in the network, an UE is considered covered as long as the received power by the UE is greater than some specific threshold. For an arbitrary UA ($\mathcal{P}$), we define the coverage degree $C(\boldsymbol{e}) \in [0, 1]$ as the fraction of covered UEs:

$$C(\boldsymbol{e}) \coloneqq \frac{1}{S} \sum_{i \in \mathcal{B}} \sum_{m \in \mathcal{P}_i} \mathbb{1}_{i,m} \tag{4.6}$$

where:

$$(\forall i \in \mathcal{B})(\forall m \in \mathcal{S})\ \mathbb{1}_{i,m} \coloneqq \begin{cases} 1 & [p_i g_{i,m}(e_i)]^{[\mathrm{dBm}]} > \xi_m \\ 0 & \text{otherwise} \end{cases}$$

where ξ_m is the service requirement threshold at UE m in dBm.

4.2.3 Optimization Problem Formulation

Against this formulation, we choose to maximize the GMAR considering utilization and coverage constraints. Formally, the optimization problem is proposed as in Eq. (4.7):

$$\begin{aligned} \underset{\mathcal{P},\boldsymbol{\rho},\boldsymbol{e}}{\text{minimize}} \quad & \Phi_1(\boldsymbol{\rho}) \\ \text{subject to} \quad & \boldsymbol{\rho} \in \mathrm{Fix}(T) \\ & \mathbf{0} \le \boldsymbol{\rho} \le \mathbf{1} \\ & \boldsymbol{e} \in \mathbb{D}_e^B \\ & C(\boldsymbol{e}) \ge \mathrm{C}_{\min} \end{aligned} \tag{4.7}$$

As in every constrained optimization problem, it is not enough to find the minimum point of the objective function ($\Phi_1(\boldsymbol{\rho})$). It is also needed to guarantee that the minimum lies within the feasible set (jointly specified by the constraints of the optimization problem). Hence, the constraints in Eq. (4.7) have been defined to connect to the mobile network dynamics; namely: the first constraint ensures that the objective function is evaluated at the solutions to the system in Eq. (4.5) through FPI, which corresponds to the definition of utilization in Eq. (4.3); the second constraint also follows from the definition of utilization (the PRB usage can not exceed 1); the third constraint limits the antenna's downtilt values to be within a discrete set, whereas the last constraint ensures that the network coverage is satisfactory.

4.3 A Hybrid Solution

The problem in Eq. (4.7) is complex owing to the implicit formulation of the cell utilization, the choice of UE-cell association $\mathcal{P}$, and the effects of the antenna downtilt on the received power, coverage, and cell utilization. However, exploiting the fact that for fixed $\boldsymbol{p}$, $\boldsymbol{e}$, $\boldsymbol{\gamma}$, and $\mathcal{P}$, the optimal utilization vector minimizing Φ_1 is given as the unique solution to $\boldsymbol{\rho} \in \mathrm{Fix}(T)$ as it will be seen in Section 4.3.1 (and already proven in [82], [53]), and by noticing also that having fixed values for $\boldsymbol{e}$, $\boldsymbol{p}$, and $\boldsymbol{\gamma}$

yields the fact that the association $\mathcal{P}$ determines $\boldsymbol{\rho}$ (as in Eq. (4.3)), it is possible to define a utility function aligned with the problem in Eq. (4.7) by considering a compromise between available capacity (GMAR) and coverage. To this end, we introduce $0 \leq \kappa_1, \kappa_2 \leq 1; \kappa_1 + \kappa_2 = 1$, and we define the following utility function:

$$\mathcal{U} \coloneqq \kappa_1 10^{-\frac{\Phi_1(\boldsymbol{\rho})}{B}} + \kappa_2 C(\boldsymbol{e}) \tag{4.8}$$

The first term of the utility function corresponds to the GMAR, whereas the second term is the coverage degree as defined in Eq. (4.6). We propose to maximize $\mathcal{U}$ in Eq. (4.8) as a surrogated version of the original optimization problem in Eq. (4.7) (subject to the same three first constraints).

To maximize $\mathcal{U}$ (i.e., to find $\boldsymbol{e}^*$ and $\boldsymbol{\rho}^*$), several approaches could be followed:

1. Naively and exhaustively searching over the antenna downtilt: first fixing $\boldsymbol{e}$, measuring $C(\boldsymbol{e})$, finding $\boldsymbol{\rho}^*$ according to Section 4.3.1, computing $\mathcal{U}$ and finally selecting the value of $\boldsymbol{e}$ exhibiting the highest utility value ($\boldsymbol{e}^*$). Unfortunately, this approach is intractable due to the high dimensionality of the problem.

2. Alternately, a heuristic search could also be used. Since we have a discrete set for the downtilt candidates, for maximizing Eq. (4.8), which is a combinatorial optimization problem with permutation property [88], Simulated Annealing (SA), as described in Appendix E, is selected as a benchmark in this research. Unfortunately, such an approach is computationally expensive due to the high number of parameters which makes it not that scalable: as explained in Appendix E, to get close to the global optimum, the temperature of the model must not decrease that fast, and the number of solution perturbations per temperature (i.e., i_{max}, see Appendix E) must be high. Thus, running an instance of SA is time-consuming. Additionally, the higher the number of parameters to be estimated, the higher the "exploration" time of the method.

Accordingly, we propose the hybrid approach in Fig. 4.2, which is revised in Fig. 4.4 to include more details about the training of the ML model (see the dashed lines). In Fig. 4.4, an inner loop is envisaged to maximize the first term in Eq. (4.8) (i.e., finding the optimal user association $\mathcal{P}^*$), whereas the outer loop tries to maximize the latter term (i.e., finding the best antenna downtilt vector). As mentioned before, the compromise between both terms is captured through the weights κ_1, κ_2. It is important to notice that both loops work in coordination (not necessarily in sync), meaning that the global utility function in Eq. (4.8) is jointly maximized.

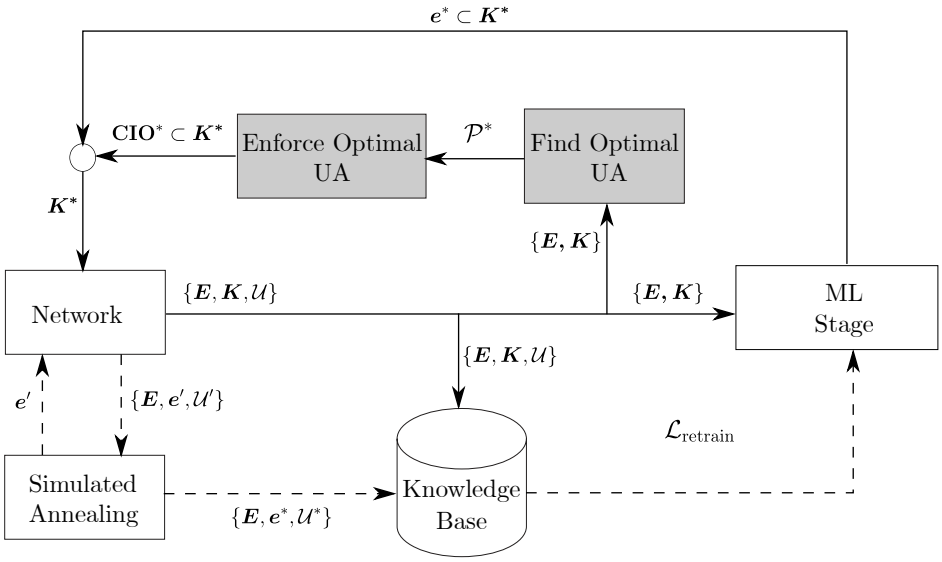


Figure 4.4: Hybrid approach with training pipeline.

As it will be shown in Section 4.3.1, the optimal user association $\mathcal{P}^*$ can be obtained by using a FPI algorithm, and that association can be enforced using a search method based on the A3-HO condition, as explained in Section 4.3.2. The chaining of both methods makes up the inner loop of our hybrid solution.

On the other side, for the outer loop, we propose to use a machine learning stage to directly predict $\boldsymbol{e}^*$ based on the environmental conditions $\boldsymbol{E}$. To train the ML model, it is needed to gather enough information about the tuple: $(\boldsymbol{E}, \boldsymbol{e}^*)$. To accomplish that we use a "smart exploration" approach, based on SA heuristic, as shown in Fig. 4.4. The term offline "smart exploration" refers to the feature of SA to accept solutions that improve the utility as the exploration time passes by. More details about the implementation of SA are given in Section 4.3.3.2, which is devoted to explaining the outer loop of our framework.

One of the main advantages of our method, as we will see later on, is that even if the predicted tilt configurations obtained through the outer loop are not optimal, the inner loop still provides the optimal CIO values for the given tilt configuration $\boldsymbol{e}$. In the following sections, we dive into the details of every box in Fig. 4.4.

4.3.1 Inner Loop: Finding the Optimal Association ($\mathcal{P}*$)

Maximizing Eq. (4.8) over the set of $\mathcal{P}, \boldsymbol{\rho}$, for fixed vectors $\boldsymbol{e}$ and $\boldsymbol{p}$, yields:

$$\begin{aligned} \underset{\mathcal{P},\boldsymbol{\rho}}{\text{maximize}} \quad & \mathcal{U} \\ \text{subject to} \quad & \boldsymbol{\rho} \in \text{Fix}(T) \\ & \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \end{aligned} \tag{4.9}$$

According to [82], [83], [53], the solution to Eq. (4.9) gives a unique policy to associate every UE to a single cell (i.e., $\mathcal{P}^*$). Specifically, UE m should be associated with the cell k^* , with:

$$k^* = \underset{i \in \mathcal{B}}{\text{argmax}} \, (1 - \rho_i)^{\alpha} \, c_{i,m}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P}) \tag{4.10}$$

In simpler words, the m-th UE should be connected to the cell which maximizes the product on the right-hand side of Eq. (4.10).

The association rule in Eq. (4.10) can be interpreted as follows: the best association between an UE and a cell is ruled not only by the Shannon capacity at the UE location (which is a purely SINR-based decision), but it also considers the network utilization. Therefore, the UEs will avoid connecting to highly utilized cells. Hereby lies the load-balancing principle exposed in this chapter.

As we mentioned before, with $\alpha = 0$ or $\alpha = 1$ we pursue the maximization of available resources (either as the summation or as the geometric mean respectively). However, only for $\alpha \geq 1$, a load balancing effect is really considered from Eq. (4.10). Hence, our selection of $\alpha = 1$ in this thesis. Another exciting relation shows up when $\alpha = 1$ as the product in Eq. (4.10) becomes the weighted average rate over time for a processor sharing M/G/1 queue with a Round Robin scheduling policy, as it was established in Section 2.1.4 (please see Eq. (2.8)). Therefore, when $\alpha = 1$, the association policy given by Eq. (4.10) aims at improving the UE throughput (as observed by the scheduler's queue [23]).

The association policy in Eq. (4.10) is used along a FPI to find the best user association ($\mathcal{P}^*$), which is afterward enforced in the network as explained in Section 4.3.2. Our implementation to obtain $\mathcal{P}^*$ is proposed in Algorithm 1.

Please notice that the association rule in Eq. (4.10) is implemented in line 4 of Algorithm 1. The utilization estimation in Eq. (4.3) takes place in line 5, and the Fixed Point Iteration (FPI) is implemented in line 7. However, we do not use the standard form in Eq. (4.4) but rather an interpolated version, as suggested in [82]. Algorithm 1 runs iteratively until the cell utilization estimation converges, as expressed in line 8, where an estimation error (μ) is calculated. The outputs of the algorithm

Algorithm 1: FPI for the calculation of the best UA

INPUT: μ,ϵ,$k = 0$,$\boldsymbol{\rho}^{(0)} \in \mathbb{R}_{+}^{B}$,$\beta$
while $\mu > \epsilon$ **do**
  for all cell $i \in \mathcal{B}$ **do**
    calculate new UA: $\mathcal{P}_i^{(k)} = \left\{ m \in \mathcal{S} \mid i = \underset{j \in \mathcal{B}}{\operatorname{argmax}} \left(1 - \rho_j^{(k)}\right) c_{j,m}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P}) \right\}$;
    calculate new utilization: $T_i(\boldsymbol{\rho}^{(k)}) = \min \left\{ \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{p}, \boldsymbol{e}, \mathcal{P})}, 1 - \epsilon \right\}$;
  end for
  $\boldsymbol{\rho}^{(k+1)} = \beta \boldsymbol{\rho}^{(k)} + (1 - \beta) T(\boldsymbol{\rho}^{(k)})$
  $\mu = \| \boldsymbol{\rho}^{(k+1)} - \boldsymbol{\rho}^{(k)} \|_2, k = k + 1$;
end while
OUTPUT:$\mathcal{P}_i^* = \mathcal{P}_i^{(k)}$,$\boldsymbol{\rho}^* = \boldsymbol{\rho}^{(k)}$,$\Phi_1 = \sum_{i=1}^{B} -\log(1 - \rho_i^{(k)})$, GMAR = $10^{-\Phi_1/|B|}$

are, the optimal user association $\mathcal{P}^*$, the cell utilization imposed by the $\mathcal{P}^*$, and the maximum achievable GMAR.

4.3.2 Inner Loop: Enforcing $\mathcal{P}*$ into the Network

To modify the ongoing UA in the network according to the optimal association $\mathcal{P}^*$ from Algorithm 1, we consider the handover procedure, as suggested in [53]. As it was already explained in Section 2.1.5, an A3-HO is carried out whenever the neighbor cell RSRP becomes stronger than the RSRP of the serving cell by a certain offset. A simplified version of the entry condition for an A3 handover from cell i to cell j is given in Eq. (2.9), which is revisited in Eq. (4.11):

$$[p_i g_{i,m}(e_i)]^{[\mathrm{dBm}]} + H + \mathrm{CIO_{i,j}} < [p_j g_{j,m}(e_j)]^{[\mathrm{dBm}]}, \tag{4.11}$$

where the CIO is a power offset given in [dB]. For the sake of avoiding coverage holes, we consider $\mathbf{CIO} \coloneqq (\mathrm{CIO_{i,j}})^{B \times B}$ with $\mathrm{CIO_{i,j}} = -\mathrm{CIO_{j,i}}$ and $\mathrm{CIO_{i,i}} = 0 \ \forall\, i, j \in \mathcal{B}$. As mentioned in Section 2.1.5, H is the hysteresis value which shows the difference needed between the measurements of the serving and the neighbor cells to trigger the A3 event also in [dB].

Likewise, an UE m will remain associated to cell i as long as:

$$[p_i g_{i,m}(e_i)]^{[\mathrm{dBm}]} + H + \mathrm{CIO_{i,j}} \geq [p_j g_{j,m}(e_j)]^{[\mathrm{dBm}]}$$

therefore, the UA policy ruled by the HO procedure is defined as:

$$\overline{k} = \underset{i}{\text{argmax}}([p_i g_{i,m}(e_i)]^{[\text{dBm}]} + H + \text{CIO}_{\text{i,j}} - [p_j g_{j,m}(e_j)]^{[\text{dBm}]}), \forall j \in \mathcal{B} \setminus i.$$

Thus, $\overline{k}$ denotes the index of the cell that serves UE m according to the current HO setting, whereas k^* in Eq. (4.10) denotes the index of the cell that should serve UE m according to the optimal association $\mathcal{P}^*$. To measure the difference between both associations, let us introduce:

$$\mathbb{1}_i(x) := \begin{cases} 1 & x = i \\ 0 & \text{otherwise} \end{cases}$$

and the distance between both association policies per cell as [53]:

$$D_i(\mathbf{CIO}) := \sum_{m \in \mathcal{S}} |\mathbb{1}_i(k^*) - \mathbb{1}_i(\overline{k})|. \tag{4.12}$$

In other words, Eq. (4.12) counts the number of users for whom the current cell, specified by the A3-HO parameters, is different from the optimal one given by Eq. (4.10). Therefore, as in [53], the CIO values can be adjusted in a way that the mismatch is minimized:

$$\mathbf{CIO}^* = \underset{\mathbf{CIO}}{\text{argmin}} \sum_{i=1}^{B} D_i(\mathbf{CIO}). \tag{4.13}$$

In principle, given a set of possible values for $\text{CIO}_{\text{i,j}}$, the problem in Eq. (4.13) is a complex combinatorial problem. Therefore, a direct search is proposed at this stage, as shown in Algorithm 2: in line 6, the current cell ($\overline{k}$) for the m-th user is obtained. In line 7, the current cell ($\overline{k}$) is compared to the optimal one (k^*). If both cells are equal, there is no adjustment in the CIO value (from the m-th user's perspective) as in line 8. If the cells are not the same, a HO of the m-th user from cell $\overline{k}$ to cell k^* should be triggered; therefore, the minimum needed CIO value is calculated as in line 10. The adjustment in the CIO value is driven by the A3-HO condition. $\text{CIO}_{\text{i,j}}{}^m$ is the offset between cells i and j (from the perspective of UE m) needed to be compliant with the best association. Line 16 in Algorithm 2 is intended to conciliate the final CIO value when more than one user should be transferred from cell $\overline{k}$ to cell k^*, and in line 17, we guarantee that $\text{CIO}_{\text{i,j}} = -\text{CIO}_{\text{j,i}}$ according to the definition.

Please notice that only discrete changes in the CIO values are allowed. Those changes are determined by the input variables Δ and CIO_{max}, which must be compliant with

the 3GPP specifications. According to the standard, a CIO value can be between -24 and 24 dB. However, for this thesis, only values between -12 and 12 dB with steps of 0.1 dB were considered

Algorithm 2: Enforcement of the UA policy

INPUT: $\mathcal{P}^*$,Δ,$\text{CIO}_{\max}$
RESET CIO
for all $j \in \mathcal{B}$ **do**
  $k^* = j$: optimal cell
  for all $m \in \mathcal{P}_j^*$ **do**
    $i = \{k \mid m \in \mathcal{P}_k\}$
    $\overline{k} = i$: current cell
    if $\overline{k}$=k^* **then**
      $\text{CIO}_{\text{i,j}}^{\text{m}} = 0.0$: no HO is needed for m-th user
    else
      $\text{CIO}_{\text{i,j}}^{\text{m}} = \Delta * \text{floor}((\lceil p_j g_{j,m}(e_j) \rceil^{[\text{dBm}]} - \lceil p_i g_{i,m}(e_i) \rceil^{[\text{dBm}]} - H)/\Delta)$
    end if
  end for
end for
for all $i \in \mathcal{B}$ **do**
  for all $j \in \mathcal{B}$ & $i! = j$ **do**
    $\text{CIO}_{\text{i,j}} = \min_{m \in \mathcal{P}_j^*} \{\text{CIO}_{\text{i,j}}{}^{m}\}$
    $\text{CIO}_{\text{j,i}} = -1.0 * \text{CIO}_{\text{i,j}}$
  end for
end for
OUTPUT:$\mathbf{CIO}^* = \mathbf{CIO}$

4.3.3 Outer Loop: a Supervised ML Stage

The outer loop is made of a supervised machine learning model predicting the best possible antenna downtilt vector $\boldsymbol{e}^*$ based on the environmental conditions ($\boldsymbol{E}$). For the joint optimization of MLB and CCO, $\boldsymbol{E}$ corresponds to the set of mean UE rate requirements (γ_m), which can change over time, as well as the user distribution (uD_i), as defined in Section 3.3. Because we have a multioutput prediction task (one output per cell's parameter to be predicted), we land in the so-called multioutput learning domain.

4.3.3.1 Multioutput Learning

In a C-SON scheme, while simultaneously optimizing multiple network parameters, it is possible to formulate a multioutput learning problem in which every output of the prediction corresponds to a parameter of a cell. In our case, every prediction output corresponds to the best possible downtilt value for each cell, i.e., e_i^*. Depending on the cardinality of each output, the multioutput learning problem can be categorized as in Table 4.2 [89, 90].

Number of outputs	Output cardinality	Type of problem
>1	2 (i.e., 0 or 1)	Multilabel classification
>1	>2	Multiclass-multioutput classification
>1	Continuous	Multioutput regression

Table 4.2: Multioutput learning categories.

Since we deal with multi-cell scenarios, with a network layout like the one depicted in Fig. 2.10 with $B = 21$ cells, and to keep the data collection process tractable, we stick with a low cardinality of the targets without loss of generality, even though in real deployments, a higher cardinality (or even continuity) of the output variables is desired. Therefore, we will consider binary outputs, which are extensively used in mobile networks, such as switching on or off a cell, using high or low antenna downtilts, or using high or lower transmit power. Therefore, we land in the multilabel classification world according to Table 4.2.

In the central branch of Fig. 4.5 it is shown a taxonomy of the meta-estimators that could be used to tackle a multilabel classification problem [89, 90].

We supply some specifics about multioutput classifiers and classifier chains in Appendix D. In general, the main difference between both families is that the multioutput classifiers consider that the outputs are statistically independent, and therefore independent classifiers are trained (one for every output). In contrast, the classifier chains drop the inter-output independence assumption, allowing a linkage among individual classifiers. Even though both families are considered in the simulation studies in this thesis, it is expected to have a better performance for the classifiers chains as there are interdependencies among cells in a real mobile network.

Apart from the two families above, deep neural networks are also considered to solve the multioutput learning problem. That decision is natural because FFNNs support multiple inputs, multiple outputs, and multiple cardinalities in the outputs (even continuity).

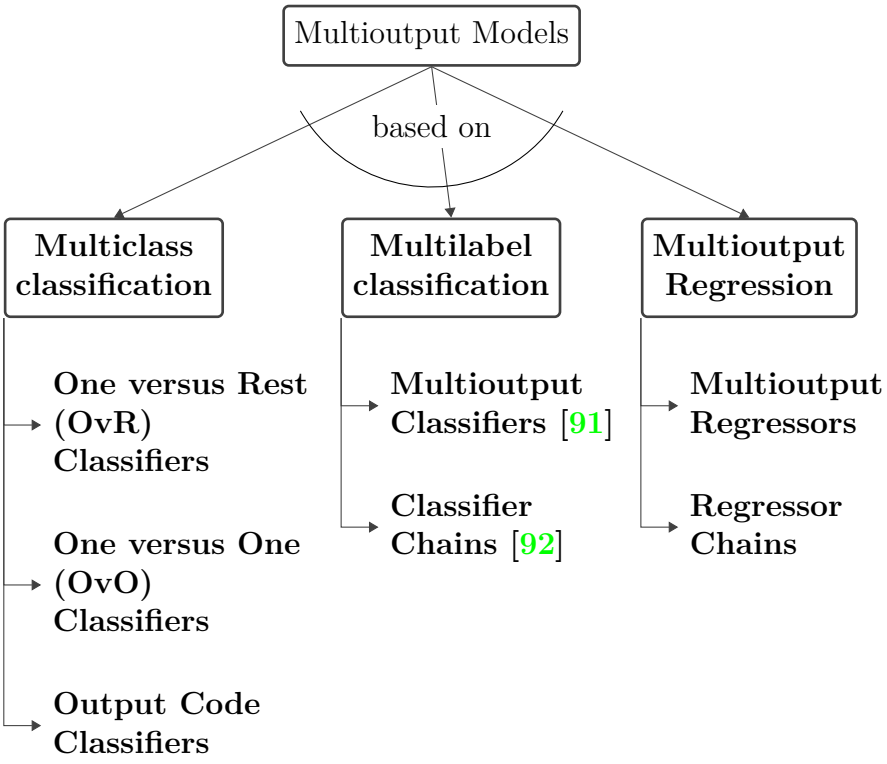


Figure 4.5: Multioutput models taxonomy

The general FFNN architecture is depicted in Fig. 4.6. It is evident the multiple input - multiple output nature of the architecture. To control the cardinality of every output, several activation functions in the output layer should be considered: e.g., when the output is binary, the sigmoid function is the natural selection; when the cardinality is higher than 2, the softmax function should be selected; if continuity is expected a linear activation function should be considered for each output in the output layer.

Regardless of the selected model, its performance depends largely on the amount and quality of the information used to train it. Therefore, the collection of training data is an essential step for the outer loop of our solution.

4.3.3.2 Training Set Collection

Naively, the labeled data for training the ML models could be obtained through an offline exhaustive search procedure given a small number of tilt candidates ($|\mathbb{D}_e|$). If generating the training set is not feasible using an exhaustive search (e.g., due to the high number of parameter permutations), training sets can be constructed offline with time-consuming heuristics such as SA. As shown in Fig. 4.4, the latter is the

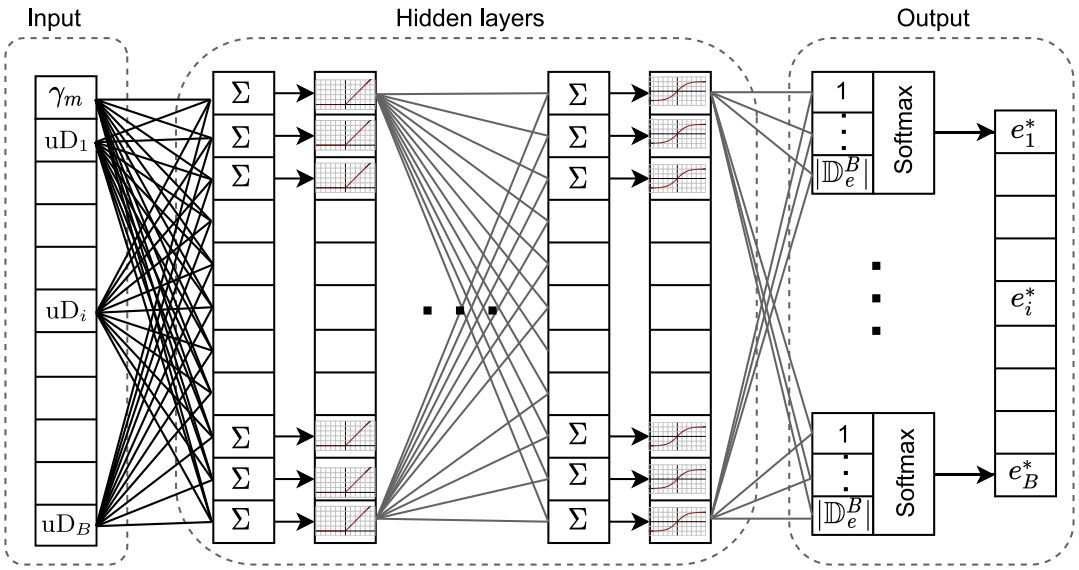


Figure 4.6: Feed forward neural network for multioutput learning problem

approach followed in the document to get labeled data, i.e., $(\boldsymbol{E}, \boldsymbol{e}^*)$. Nevertheless, as we mentioned before, the higher the number of parameters to be explored, the higher the "exploration" time of the heuristic. Since the total number of tilt combinations is given by the permutation with repetitions $|\mathbb{D}_e|^B$, the number of cells should be small or the cardinality of every output to keep a reasonable exploration time. As stated before, we choose the latter option.

Our version of the SA heuristic (detailed in Appendix E) for getting labeled data and as a time-consuming benchmark for maximizing the utility in Eq. (4.8) is presented in Algorithm 3.

4.4 Comparative Evaluation of Algorithms for joint MLB-CCO Optimization

A simulation campaign was conducted with a mobile network comprising 21 cells, as depicted in Fig. 3.7. The main parameters for the simulation are shown in Table 4.3.

Algorithm 3: Simulated Annealing for MLB and CCO Optimization

INPUT: $e_{init}, i_{max}, \tau_{min}, \tau_{max}$
INITIALIZE: $\tau = \tau_{max}, e = e_{init}$
$U(e) \leftarrow$ Compute utility in (4.8)
while $\tau > \tau_{min}$ **do**
  for all $i = 1$ to i_{max} **do**
    $U(e') \leftarrow$ Compute utility in (4.8) for $e' \in \mathbb{D}_e^B \setminus e$
    $\Delta(U) = U(e') - U(e)$
    flag=1
    if $\Delta(U) > 0$ **then**
      Accept $e \leftarrow e'$
    else
      Calculate probability: $\Pr(\Delta(U)) = e^{\frac{\Delta(U)}{\tau}}$
      if $\Pr(\Delta(U)) > \text{rnd}(0,1)$ **then**
        Accept $e \leftarrow e'$
      else
        Reject $e \leftarrow e'$
        flag=0
      end if
    end if
    if flag=1 **then**
      Update $e^* = e$
    end if
  end for
  $\tau = \frac{\tau}{log(i+1)}$
end while
OUTPUT:e^*

Table 4.3: Simulation parameters

Type	**Parameter**	**Value**
System specifics	Inter-site distance	500 [m]
	Pathloss	see Eq. (2.3)
	Shadowing	see Eq. (2.3), Ω = 6[dB]
	p_i (Eq. (4.2))	46 [dBm]
	$\mathbb{D}_e$	$\{9°, 12°\}$
	Tx antennas	gain 15 [dBi], height = 32 [m]
	BW (Eq. (4.2))	10 [MHz]
	σ^2 (Eq. (4.2))	-95 [dBm]
	ξ_m (Eq. (4.6))	-115 [dBm]
	Scheduler	round robin
User specifics	UEs	240
	Mobility model	random walk
	UE positioning	uniform distribution
	UE antennas	gain 2 [dBi], height = 1.5[m]
	γ_m	300-1024 [kbps]
Algorithm specifics	H (Eq. (4.11))	3.0 [dB]
	Δ	0.1 [dB]
	CIO_{max}	12.0 [dB]
	α (Eq. (4.1))	1
	β (Algorithm (1))	0.9
	κ_1,κ_2 (Eq. (4.8))	0.5
	τ_{min}, τ_{max}	0.0001 , 100
	i_{max}	100
Sim. period	After training	30 [s] (except for SA)

To collect the training data, multiple network scenarios with randomly deployed users were simulated. For every scenario, both the distribution of the users (uD_i) and the requested data rate (γ_m) change, emulating the change in the traffic profile throughout the day. For every scenario, a search over the antenna downtilt using SA was carried out to learn the best possible tilt configuration $\boldsymbol{e}^*$. Once a large enough number of scenarios have been simulated, it is possible to train the ML Stage in the outer loop in Fig. 4.4.

Once the ML model is trained, the framework in Fig. 4.4 is ready to optimize MLB

and CCO jointly, providing implicit coordination among them. To check the performance of the joint optimization, the simulation batch duration is set to 30 seconds. At the beginning of every batch, the user distribution (uD_i) and the requested data rate (γ_m) change, and based on these environmental variables ($\boldsymbol{E}$), the best possible downtilt vector is predicted using the outer loop of Fig. 4.4 and applied afterward. Soon after the change in the downtilt, we allow the registration of UEs to the respective cells and the generation of data traffic (γ_m). At the 10th second, once we have a stable estimation of the cell utilization (ρ_i), the user association optimization is triggered using Algorithm 1, and the resulting association ($\mathcal{P}^*$) is enforced thereafter using Algorithm 2.

A subset of $\gamma_m : \{300, 400, 500, 600, 700, 800, 900\}$ [kbps] is used to evaluate the performance of the proposed framework over multiple traffic conditions. Two benchmark schemes are considered to compare the benefits of the proposed framework, namely:

1. SA-based heuristic. This long-lasting heuristic can be used not only to collect the training data but also to predict the best possible downtilt vector ($\boldsymbol{e}^*$).

2. The best fully-implicit optimization mechanism[1] in Chapter 3, using the architecture in Fig. 3.3. We want to investigate whether there is an advantage of using a hybrid approach compared with the fully-implicit method presented in the previous chapter in terms of network performance when both methods are trained with the same amount of information.

A brief discussion about the performance of every method is exposed in the following subsections. Afterward, we present and discuss the performance of the individual boxes of the hybrid approach, and we end up with a network performance comparison among all the schemes.

4.4.1 Simulated Annealing Approach

Apart from collecting data for training, SA can also be used as a long-lasting method to find the best antenna downtilt ($\boldsymbol{e}^*$). That is, SA (as in Algorithm 3) could replace the ML stage in the outer loop. For a specific user distribution[2], and for the different values of γ_m, the heuristic was in charge of maximizing the utility function $\mathcal{U}$ defined as in Eq. (4.8) through the modification of the antenna downtilt vector. The evolution of $\mathcal{U}$ for different values of γ_m is depicted in Fig. 4.7.

[1] This corresponds to the ensemble of SOM and hierarchical clustering.

[2] That was not explored during the training data collection, i.e., a "test" user distribution.

Please recall that for every value of n, i_{max} perturbations of the solutions are allowed (taking place in line 6 of Algorithm 3).

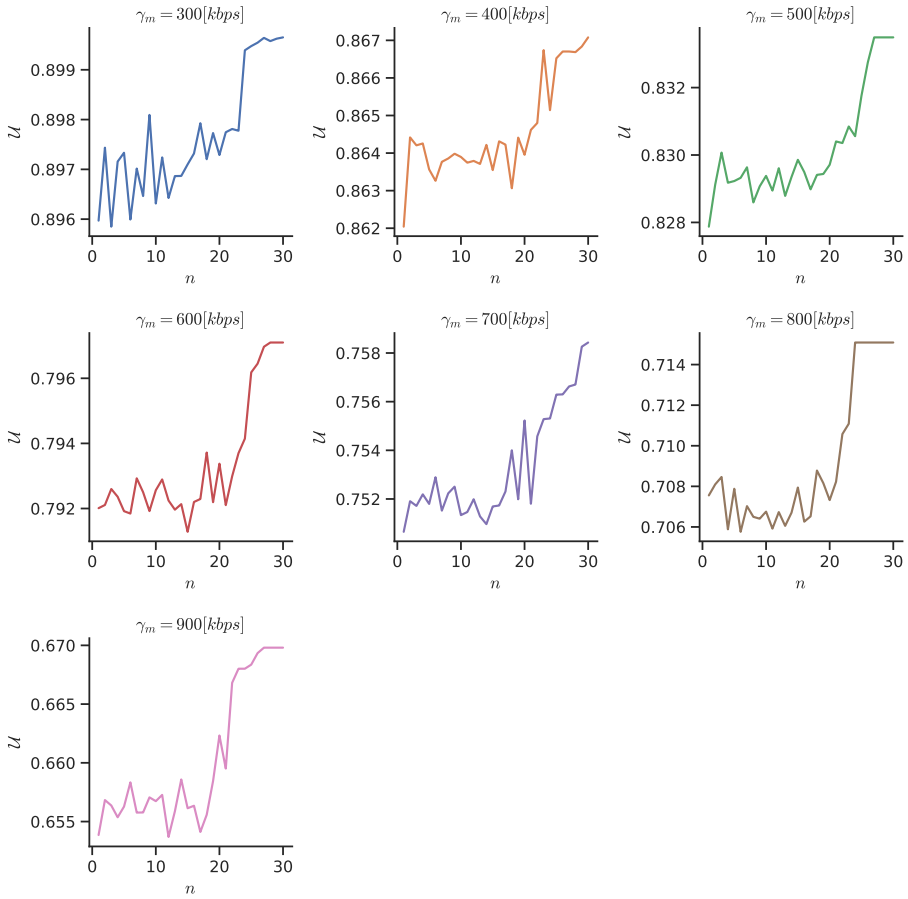


Figure 4.7: Improvement of the utility function $\mathcal{U}$ over the time

We can see that, for every value of γ_m that was considered, the proposed version of SA finds a suitable downtilt configuration provided it was given a high enough number of perturbations (i.e., long simulation time). The impact of the obtained configuration on the network performance will be supplied in Section 4.4.4.

4.4.2 Fully-Implicit Optimization Approach

Regarding the fully-implicit coordination mechanism, we found in Chapter 3 that the best approach was the combination of SOM and Hierarchical Clustering (HC). We select that model as a benchmark considering the following functional domains: the main KPI ($\mathcal{U}$) as in Eq. (4.8), the controllable part corresponds to the downtilt vector ($\boldsymbol{e}$) and CIO values (**CIO**), whereas the environmental part corresponds to the

requested data rate γ_m and user distributions (uD_i).

As in Chapter 3, a three-layer SOM is trained, and the hyperparameters ruling the model behavior are found using a grid search procedure. After finding the best SOM model, a hierarchical cluster is applied, considering only the environmental variables. As indicated in Appendix A, the Elbow method using Within cluster Sum of Squares (WSS) is used to determine the number of clusters. For this specific SF optimization problem, a cluster size of 8 is selected, as shown in Fig. 4.8 (one should choose a number of clusters so that adding another cluster does not reduce that much the total WSS).

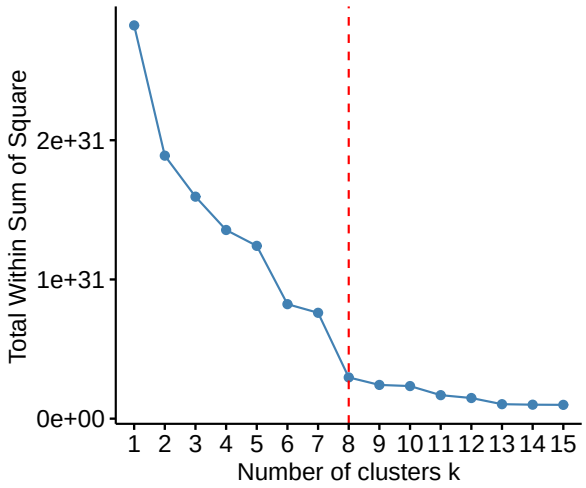


Figure 4.8: Selected number of clusters

Propagating those clusters to the KPI layer in the SOM grid (as explained in Fig. 3.5), we obtain Fig. 4.9.

From Fig. 4.9, we observed that the best value of $\mathcal{U}$ is located in the top left corner (the inner circles with the highest area), whereas the worst utility values are obtained in the central top area of the map.

On the other side, we observed some "islands" in the clustering. That means the clusters are not fully connected. An explanation of that phenomenon is given considering the "quality"[3] of the SOM grid, which shows the mean distance of objects mapped to a unit to the unit signature. The smaller the distances, the better the objects are represented by the signature [77].

The quality grid of the SOM is depicted in Fig. 4.10. As it turns out, the number of samples behind the gray units is zero, i.e., we have empty neurons. One way to solve this issue is to gather more information for training. Another way is to reduce the x_{dim}

[3]Concept introduced in Appendix A

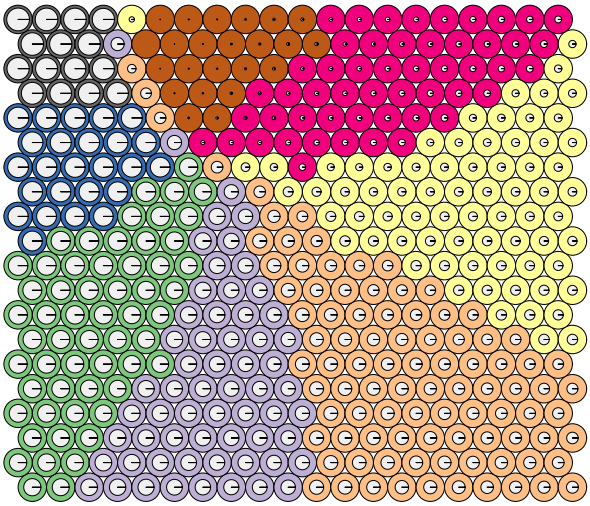


Figure 4.9: 3-layer SOM and hierarchical clustering on the KPI layer

and y_{dim} of the map (see Appendix A). However, as mentioned before, a grid search was carried out to find the appropriate dimensions of the grid with respect to a score metric (as shown in Fig. 3.4). The results of that grid search are shown in Table 4.4.

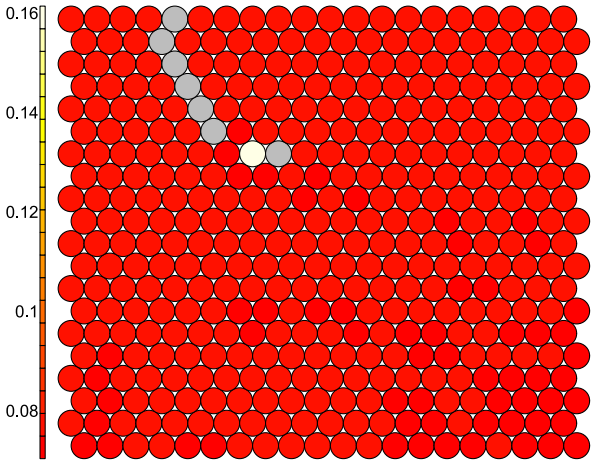


Figure 4.10: Quality metric of the 3-layer SOM

For the sake of fairness to the other methods, we keep the same volume of data for training the SOM, and we keep the dimensions of the grids according to Table 4.4, i.e., x_{dim} = 20 and y_{dim} = 20.

4.4.3 Hybrid Approach

To find the optimal user association ($\mathcal{P}^*$) and map it to the suitable CIO values ($\mathbf{CIO}^*$), it is needed to execute both Algorithm 1 and Algorithm 2. Therefore, it makes sense to evaluate these algorithms separately.

Table 4.4: Grid search over x_{dim} and y_{dim}

x_{dim}	y_{dim}	RMSE	R^2	Mean Absolute Error (MAE)
5	5	0.0348	0.8991	0.0279
5	10	0.0315	0.9174	0.0248
5	20	0.0282	0.9339	0.0218
20	5	0.0275	0.9370	0.0214
10	10	0.0268	0.9404	0.0210
10	20	0.0241	0.9515	0.0186
20	10	0.0247	0.9492	0.0191
20	20	0.0212	0.9622	0.0162

4.4.3.1 Inner loop: Algorithm 1

For a specific user distribution and variable γ_m, the evolution of the cost function Φ_1 from Algorithm 1 as well as GMAR are depicted in Fig. 4.11 and Fig. 4.12 respectively.

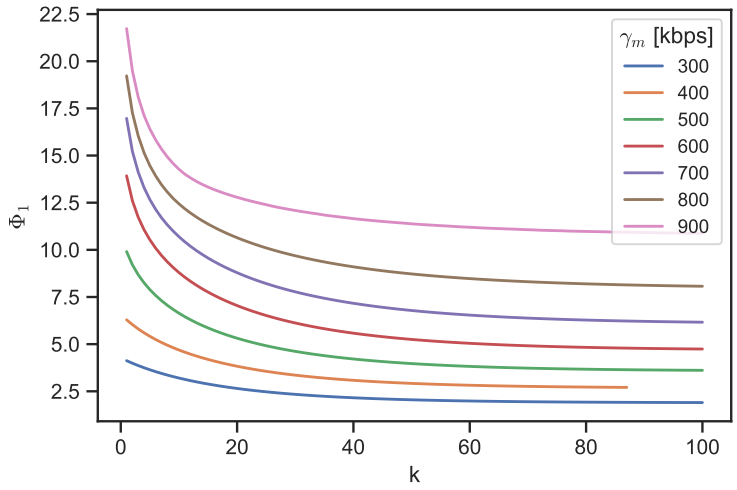


Figure 4.11: Evolution of the cost function: Φ_1

It is evident that after a few iterations of the proposed method, the cost function monotonically converges for every value of γ_m, and so does the GMAR.

4.4.3.2 Inner loop: Algorithm 2

To illustrate the Algorithm 2 dynamics, multiple snapshots of the UA are taken to show three different states: the initial UA, the optimal UA ($\mathcal{P}^*$), and the final UA

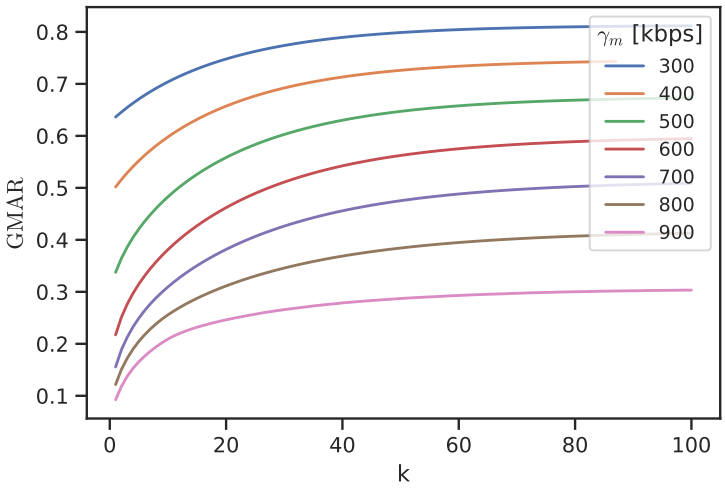


Figure 4.12: Evolution of the GMAR function

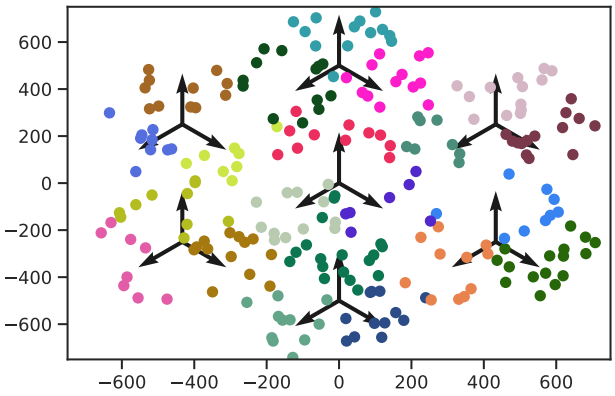
(i.e., after Algorithm 2 execution). Those snapshots are shown in Fig. 4.13 when γ_m was set to 300 [kbps] for the user distribution depicted in the figures.

As you can see from Fig. 4.13(c), not all the UEs can be associated with the desired cell as depicted in Fig. 4.13(b) (please notice the dots surrounded by dashed boxes). This is due to the real constraints imposed on the values Δ and CIO_{max} (given in Table 4.3). Therefore, to measure the mismatch between both the best and the final associations, as well as the mean number of modified adjacencies, and the maximum offset applied to the adjacencies, Table 4.5 is presented.

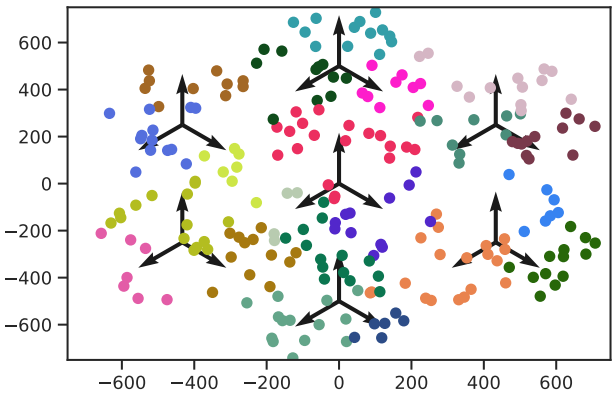
Table 4.5: Performance of the Algorithm 2 for a macro-cell scenario

γ_m [Kbps]	**Matching rate [%]**	**Modified Adjacencies**	$\max\{\text{CIO}_{i,j}\}$ [dB]
300	99.16	13	5.0
400	99.58	9	5.0
500	99.58	9	2.9
600	99.58	8	3.0
700	100	10	3.9
800	100	11	3.9
900	99.58	2	5.5

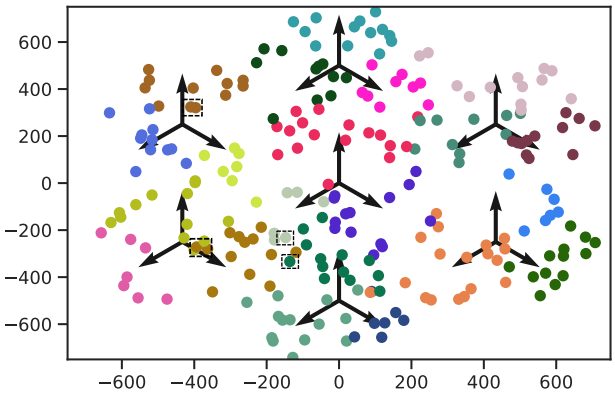
As seen from the second column of Table 4.5, the matching rates between the final and optimal associations are pretty high under reasonable changes of CIO, which implies that Algorithm 2 exhibits a consistent behavior for the considered values of



(a) Initial user association



(b) Optimal user association $\mathcal{P}^*$ (from Algorithm 1)



(c) Final user association (after the execution of Algorithm 2).

Figure 4.13: UA optimization and enforcement for γ_m = 300[kbps]

γ_m.

4.4.3.3 Impact on the Network Performance of the Inner Loop

From the network perspective, it is possible to see to what extent the chaining of the shaded boxes in Fig. 4.2 improves the performance. Let us consider the mean SINR profile for multiple values for γ_m, as shown in Fig. 4.14.

According to Fig. 4.14, an improvement in SINR is evident after TTT has expired (see Section 2.1.5) and all the HOs (product of the UA optimization and enforcement) were executed. Therefore we conclude that the inner loop of the proposed framework improves the network performance.

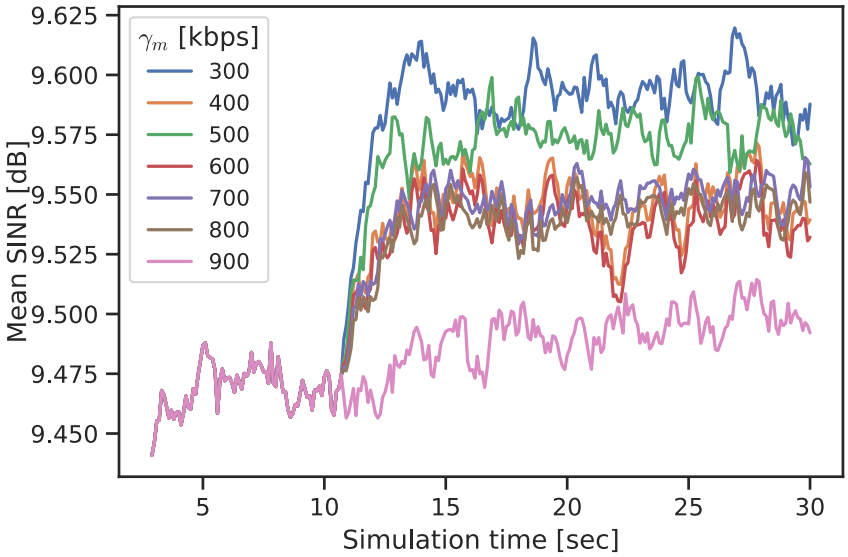


Figure 4.14: Improvement of the SINR over time

4.4.3.4 Outer Loop

The hybrid approach, based on ML for the prediction of $\boldsymbol{e}^*$ and the chaining of Algorithm 1 and Algorithm 2 to find $\mathbf{CIO}^*$, is also tailored to achieve the best performance during the prediction of $\boldsymbol{e}^*$ using the most suitable ML Stage in Fig. 4.2. As mentioned before, several multioutput models were considered according to Fig. 4.5. The metric considered to select the best model was the "micro-averaged" version of the Area Under the Receiver Operating Characteristic (ROC) Curve (see Appendix D).

Based on that selection, a grid search was carried out to find the appropriate hyperparameters per model. A deep explanation of the considered models and their main

hyperparameters is not within the scope of this thesis. In Table 4.6, it is possible to see the results of the grid search procedure. The considered hyperparameters for each model are not shown. Please notice that the best model with respect to the selected score metric is the chain classifier made of logistic regressors in all the targets. That makes sense considering that $\mathbb{D}_e = \{9°, 12°\}$, i.e., a binary output.

Table 4.6: Grid search for different multioutput strategies

Type	Model	Area Under ROC Curve
Multioutput classifier	Gradient Boost	0.7565
	K-Neighbors	0.7539
	Random Forest	0.7010
	Extremely Randomized Trees	0.6625
	Multi-layer Perceptron	0.6113
Chain classifier	Logistic Regression	0.81
	K-Neighbors	0.8021
	Ridge Classifier	0.8020
	Random Forest	0.7804
	Extremely Randomized Trees	0.7802
	Quadratic Discriminant Analysis	0.7619
	Nearest centroid classifier	0.7502
Deep Learning	FFNN (see Fig. 4.6)	0.6785

As stated before, apart from the multioutput models, a FFNN with a deep structure (see Fig. 4.6) was also considered. As $\mathbb{D}_e = \{9°, 12°\}$, we only have two options for the tilt of every cell. Therefore, the output layer in Fig. 4.6 uses sigmoidal activation functions. Please remember that when more than two categories of tilt need to be predicted, the activation function should be softmax; if a continuous value is needed, a linear activation function should be selected.

The deep Artificial Neural Network (ANN) did not perform as expected for this scenario. The hypothesis behind this is that deep ANNs could not capture the dynamics of the studied SFs with the amount of available training data. As a matter of fact, in Chapter 5, we collect 50% more labeled data, and we see that the FFNN outperforms the same candidates. However, in real networks, we often deal with limited reference data (ground truth) because its acquisition is extremely time-consuming and costly.

4.4.4 Performance Comparison

Bearing in mind the results in the previous subsections, we carry out a performance comparison of the considered approaches. The following methods are compared: INITIAL corresponds to a static configuration obtained during the exploration done by SA when $n = 20$ (see Fig. 4.7), FPI corresponds to activating MLB only (i.e., the inner loop in Fig. 4.2), SA corresponds to using both loops in Fig. 4.2, being the outer loop driven by SA, SOM+HC represents the solution proposed in the previous chapter, i.e., Fig. 3.3, and finally HYBRID represents the scenario in Fig. 4.2 with the ML Stage being a chain of logistic regressors solving a multioutput learning problem.

In Fig. 4.15 it is shown the SINR in the network. It is evident that the best performance is achieved with the HYBRID approach; please notice that the median, .75-percentile, and the mean, are the highest for almost all the traffic scenarios. The second-best performance is obtained with the implicit method except for some scenarios (e.g., $\gamma_m \in [400, 500, 600]$) in which SA outperforms it. The same holds for UE data rates, as shown in Fig. 4.16.

From the results above, we can deduce that the HYBRID approach (proposed in Fig. 4.2) is superior to the fully-implicit approach (proposed in Fig. 3.3 and implemented using SOM+HC) when both use the same amount of data to train the involved ML models. The HYBRID approach is more data-efficient than the fully-implicit approach.

Due to the user accommodation after the HOs execution, it is expected an impact on the cell utilization. According to the cost function in Eq. (4.1), a join reduction in the cell utilization is foreseen (please recall that one component of $\mathcal{U}$ is GMAR). Since a high cell utilization ($\rho_i = 1$) means the network might not handle new sessions, we are interested in the higher values of the cell utilization distribution (rather than a central tendency metric like the mean). At the same time, we want to avoid using the maximum cell utilization value because it is susceptible to outliers. Therefore, we select the .75-Percentile as the figure of merit. The .75-percentile of the cell utilization is shown in Table 4.7. The HYBRID approach alleviates the high cell utilization through an α-fair distribution of users across the network.

Finally, in terms of coverage, Fig. 4.17 shows the coverage degree as defined in Eq. (4.6). Once again, the best numbers are obtained with the HYBRID method.

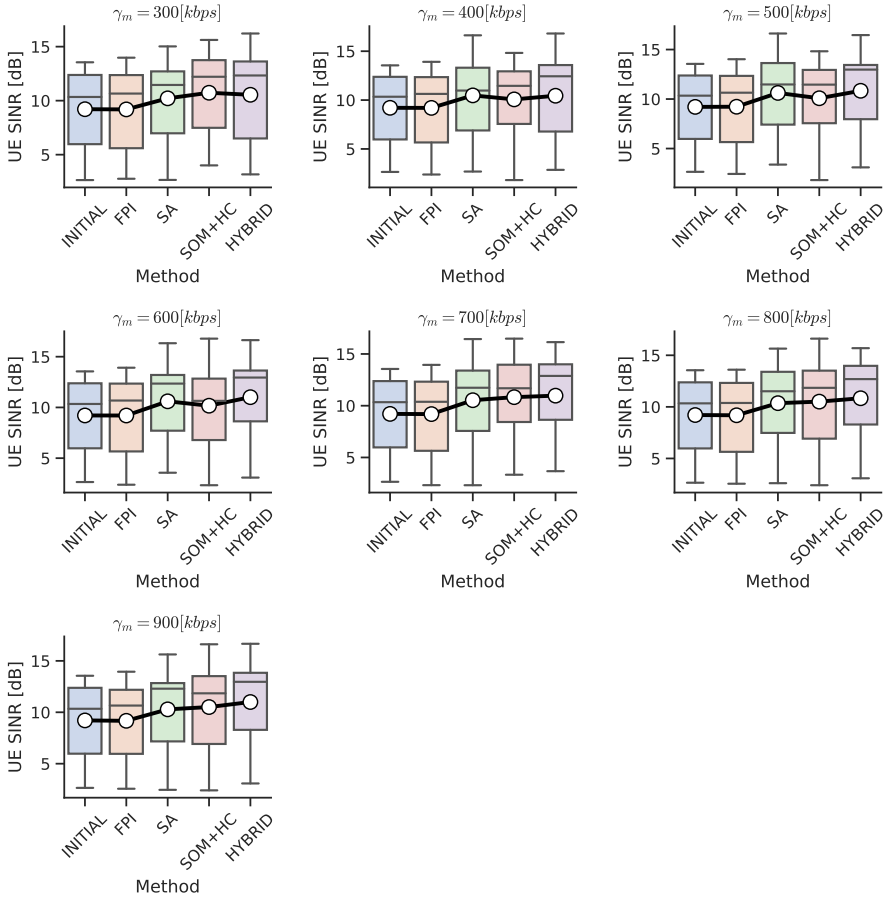


Figure 4.15: Mean UE SINR

Table 4.7: Impact on the cell utilization (and therefore available resources)

γ_m	INITIAL	FPI	SA	SOM+HC	HYBRID
300	0.3888	0.3854	0.3418	0.3545	0.3483
400	0.5177	0.5035	0.4404	0.4601	0.4404
500	0.6466	0.6254	0.5385	0.5730	0.5168
600	0.7755	0.7522	0.6317	0.6663	0.6132
700	0.9044	0.9080	0.7554	0.7430	0.7361
800	1.0000	1.0000	0.8524	0.9134	0.8454
900	1.0000	1.0000	0.9590	1.0000	0.9474

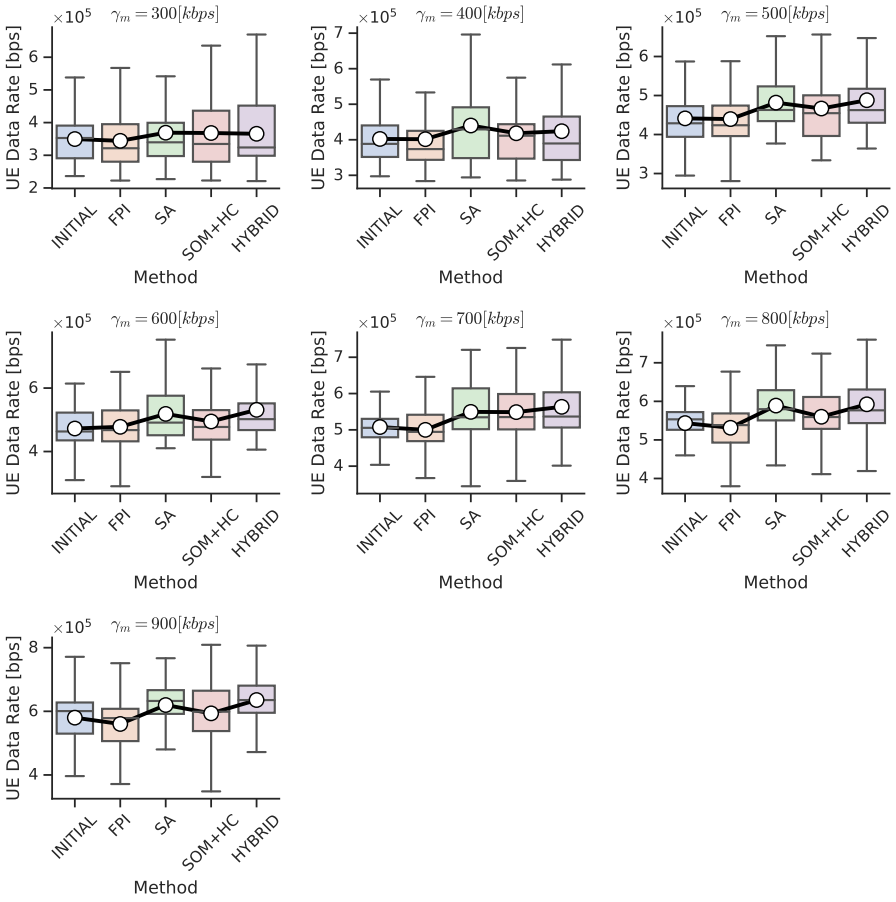


Figure 4.16: Mean UE data rate

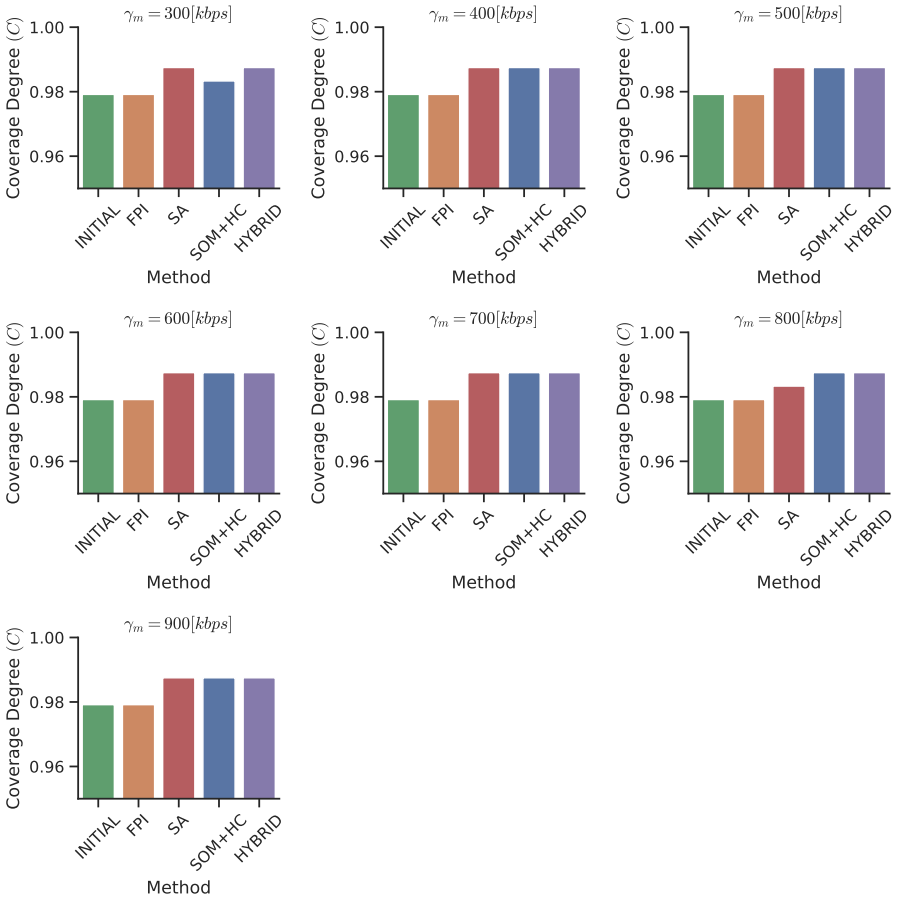


Figure 4.17: Coverage degree $C(\boldsymbol{e})$

4.5 Summary

A hybrid network optimization framework has been presented, which applies to network scenarios in which the knowledge of propagation loss values and traffic signatures is available. We assume that this information is attainable, e.g., once the MDT function is active in the network. We proposed a ML stage that predicts in an outer loop a suitable downtilt configuration from data rate requirements and users distribution information, whereas a fixed point algorithm finds the optimal user association in an inner loop given a predicted downtilt.

One of the advantages of the proposed method is that it predicts the downtilt non-iteratively, which is an essential feature for networks that should adapt quickly to the surrounding changes. Another advantage is that the fixed point algorithm guarantees an optimal user association given any prediction from the ML model. Therefore we have gained some optimality insights we were missing according to Section 3.4.

We demonstrated the advantages of jointly optimizing multiple conflicting functions in terms of network performance. The method has been shown beneficial regarding a utility function combining both coverage and composite network capacity indicators.

The main limitation of the proposed architecture is the unavailability of labeled data. Even though we have trained a ML stage using a heuristic (SA) that runs multiple scenarios during a long enough period of time, it is quite likely that in real network deployments, the amount of information for training a ML model is unavailable, or there is not such a degree of freedom to run a heuristic like SA. Of course, this limitation is also valid for the approach proposed in the previous chapter and (for the sake of fairness) for all the possible solutions aiming to solve the SF coordination problem in a centralized data-driven manner.

In that sense, we claim that there are already workarounds to face the lack of labeled data, e.g., *transfer learning* theory for FFNN models, in which the training of a neural network is carried out using information from a "similar" domain[4] in which enough information is available, not necessarily in the mobile communications world. With this technique, the deeper layers of a neural network trained with the data from the *source domain* are fine-tuned with the scarce real network information, a.k.a. *target domain*. This way, the FFNN will not have to learn from scratch all the low-level features and structure of the problem, it will only have to learn the higher-level structures (which requires less labeled data) [76].

When it comes to situations in which, unfortunately, it is not possible to find a

[4]The more similar the tasks are, the more layers in the neural network it is possible to reuse.

model trained on a similar task (i.e., transfer learning is no longer a feasible option), it could be possible to use *unsupervised pretraining* [76] using Restricted Boltzman Machines, Autoencoders or Generative Adversarial Networks, keeping the lower layers of those models and adding an output layer which could be fine-tuned using small labeled datasets.

Even though the above solutions seem to be tailored to neural network models, and for this specific chapter, the vanilla FFNN was not the best model (see Table 4.6), in [9] we carry out a study in which a FFNN was used to coordinate MLB and CCO in a smaller network with only 9 cells in a campus network scenario. Additionally, in Chapter 5 we will collect more labeled data at the expense of more simulation time, and we will see that a FFNN model becomes the most suitable model to execute the outer loop in Fig. 4.2

5 An extension of the Hybrid Approach: Including ICIC and ES

In this chapter, we present the application of the hybrid solution proposed in the previous one (see Fig. 4.4) to a different set of SFs, namely: MLB, ICIC, and ES. This solution is intended to optimize the SFs in a hybrid manner: on the one hand side, a ML stage predicts part of the best possible configuration in one shot using an outer loop, whereas an inner loop tries to solve, in a closed form, the high dimensional load balancing problem.

The relevant notation considered throughout this chapter is provided in Table 5.1.

5.1 Introduction

Modern cellular systems based on OFDMA, such as LTE (in downlink) or 5G (in both uplink and downlink), are conceived as technologies with a Frequency Reuse (FR) of 1 to achieve the highest utilization of the spectrum. FR allows all cells to transmit over the whole set of time-frequency resources (improving the spectral efficiency) but yields to ICI, where an UE is affected by interfering signals coming from the neighbor cells transmitting in the same resources (a.k.a. co-channel cells). This situation is even worse for users located in the cell borders due to the bad channel conditions expected as the distance from the serving transceiver increases. With a rise in the interference levels comes a degradation of the SINR, and therefore a throughput reduction.

To mitigate ICI, multiple schemes are available either in the time domain, such as Almost Blank Subframe (ABS)-based ICIC, or frequency reuse-n approaches, such as HFR, FFR, and SFR [17]. In this study, we focus on the latter category, especially on SFR (see Section 2.1.2). SFR provides the best spectrum reuse because the entire available system bandwidth is used in each cell (i.e., reuse 1). To reduce ICI, SFR sets up edge subbands that are different for any two neighbor cells, usually combined with lower power on the signals towards the cell centers (see Fig. 2.3).

The use of SFR in mobile networks leads to natural tradeoffs among performance

Table 5.1: List of variables

Description	**Symbol**
Network-wide configuration vector	$\boldsymbol{K} \in \mathbb{R}^{\mathrm{K_{dim}}}$
Network-wide environmental variable vector	$\boldsymbol{E} \in \mathbb{R}^{\mathrm{E_{dim}}}$
α-fairness cost function	Φ_α
Global utility function	$\mathcal{U} \in \mathbb{R}$
Geometric Mean of Available Resources	GMAR
The best possible configuration vector	$\boldsymbol{K^*} \in \mathbb{R}^{\mathrm{K_{dim}}}$
User distribution for i-th cell	$\mathrm{uD}_i \in [0, 1]$
Set of cells	$\mathcal{B} = \{1, ..., B\}$
Set of neighbor cells for i-th cell	$\mathcal{N}(i)$
i-th cell's transmit power	$p_i \in \mathbb{R}_+$
Transmit power vector	$\boldsymbol{p} \in \mathbb{R}^B_{++}$
i-th cell utilization	$\rho_i \in (0, 1]$
Cell utilization mapping	$T : \mathbb{R}^B_+ \to \mathbb{R}^B_{++}$
Cell utilization vector	$\boldsymbol{\rho} \in (0, 1]^B$
Link budget from i-th cell to m-th UE	$g_{i,m}(\cdot)$
SINR from i-th cell to m-th UE	$\mathrm{SINR}_{i,m}(\cdot)$
Shannon capacity between i-th cell to m-th UE	$c_{i,m}(\cdot)$
Reduction power factor in the i-th cell center	$\eta^c_i \in \mathbb{D}_p \subseteq \mathbb{R}_+$
Vector of power factors in cell centers	$\boldsymbol{\eta^c} \in \mathbb{D}^B_p \subseteq \mathbb{R}^B_+$
Best possible vector of power factors in cell centers	$\boldsymbol{\eta^{c*}} \in \mathbb{D}^B_p \subseteq \mathbb{R}^B_+$
Reduction power factor at the i-th cell edge	$\eta^e_i \in \mathbb{D}_p \subseteq \mathbb{R}_+$
Vector of power factors in cell edge	$\boldsymbol{\eta^e} \in \mathbb{D}^B_p \subseteq \mathbb{R}^B_+$
m-th UE's requested data rate	$\gamma_m \in \mathbb{R}_{++}$
UE's requested data rate vector	$\boldsymbol{\gamma} \in \mathbb{R}^S_{++}$
i-th cell's user association	$\mathcal{P}_i$
Set of user associations	$\mathcal{P} \coloneqq \{\mathcal{P}_1, ..., \mathcal{P}_i, ..., \mathcal{P}_B\}$
Optimal user associations	$\mathcal{P}^*$
A3-HO Hysteresis	H
A3-HO Time to trigger	TTT
Cell Individual Offset between cells i and j	$\mathrm{CIO_{i,j}}$
Best possible Cell Individual Offset values	$\mathbf{CIO^*} \in \mathbb{R}^{B \times B}$
Optimal cell for m-th UE	k^*
Edge SINR in i-th cell	$\mathrm{edgeSINR}_i \in \mathbb{R}$
Edge interference in i-th cell	$\mathrm{edgeIf}_i \in \mathbb{R}$
Interference created by the ith cell	$\mathrm{createdIf}_i \in \mathbb{R}$
Edge interference in i-th cell's neighborhood	$\mathrm{neighEdgeIf}_i \in \mathbb{R}$
UEs associated with the i-th cell's center	$\mathcal{P}^c_i$
UEs associated with the i-th cell's edge	$\mathcal{P}^e_i$
UEs location-dependent bandwidth factor	$\mathrm{BWF}_{i,m}$

metrics such as coverage for edge users, network throughput, and spectral efficiency. That compromise becomes more complex if the load imbalance among neighboring cells is considered because it causes inefficient resource utilization and low throughput. Accordingly, it is evident a tight relationship between ICIC and MLB functions, as well as the necessity of considering a joint optimization scheme between both SFs.

To the best of our knowledge, explicit coordination schemes between ICIC and MLB are only mentioned in [93] and [55]. The authors in [93] developed a tailor-made method to jointly optimize the aforementioned SFs considering FFR. Even though FFR is comparatively better in terms of coverage outage probability, SFR exhibits higher bandwidth efficiency. We claim that the outage probability could be optimized by a different SF, i.e., CCO (some insights for that kind of deployment were shown in the previous chapter and are also available in [18]). Therefore, we lean towards the use of SFR in this study. The authors in [55] proposed a combination between fixed-point iterations and a non-derivative heuristic, namely Nelder and Mead algorithm, to deal with the coordination between MLB and ICIC in a Heterogeneous Network. Unfortunately, that heuristic works well only on problems of relatively small dimensions (up to 10 decision variables) [94].

For the sake of completeness, regarding implicit coordination schemes (although for different SF), [68] proposes a tailing coordination scheme based on network performance predictions using ML followed by a posteriori multi-objective optimization process, which searches for a set of suitable configurations. In [68], it is observed that the performance of the ML models depends on the network size as the number of regressors increases with the number of cells imposing scalability issues on the solution.

Similarly to the previous chapter, in this one we combine the best of both explicit and implicit approaches to get rid of the dimensionality curse[1], as well as to boost the optimization process using a ML stage.

5.1.1 Contributions

In this chapter, we explore the applicability of the framework presented in Chapter 4 to consider the joint optimization of MLB and ICIC, the latter implemented as a frequency domain solution, namely SFR, which was introduced in Section 2.1.2. To our best knowledge, by the time of writing this document, this is the first work dealing with the joint optimization of MLB and SFR-based ICIC SFs. We compare multiple

[1] Especially when it comes to finding the appropriate values for the CIO which, as has been mentioned before, are the worst offenders in terms of dimensionality since they are meant to work in an edge-wise manner rather than cell-wise.

network dimensions of the proposed architecture in Fig. 5.1 against a fully-implicit coordination architecture as in Fig. 4.1 and a benchmark heuristic.

We propose using a neural network capable of predicting the best possible power factor vector ($\boldsymbol{\eta}^{c*}$) in one shot and show how this ML model admits an extension to ES SF, which is in charge of switching off some cells when the load conditions are not that high. We are interested in landing in the neural network domain to boost the future use of techniques to mitigate the cold-start problem that have been tailored for deep neural networks.

5.2 Formulation of the Centralized Joint Optimization Problem

As we did in the previous chapter, we start formulating an optimization problem. We select a fairness measure and present how to estimate the main variables involved in the optimization problem considering a dynamic SFR scheme working on an OFDMA network.

5.2.1 A Fairness Measure for Centralized Decision-Making and Conflict Resolution

Like we did in Chapter 4, and for the same reasons, the α-fairness function is selected as the cost function. The α-fairness function is revisited in Eq. (5.1).

$$\Phi_\alpha(\boldsymbol{\rho}) = \begin{cases} \sum_{i=1}^{B} \frac{(1-\rho_i)^{1-\alpha}}{\alpha-1} & \alpha \geq 0, \alpha \neq 1 \\ \sum_{i=1}^{B} -\log(1-\rho_i) & \alpha = 1 \end{cases} \tag{5.1}$$

Once again, our objective is to maximize the GMAR considering cell utilization and interference constraints, i.e., we select $\alpha = 1$. Unlike the previous chapter which used a combined metric to model both MLB (with GMAR) and CCO (with $C(\boldsymbol{e})$), we claim that for jointly optimizing MLB and ICIC it is enough to minimize Φ_1. The reasons behind that selection are two-fold:

- According to Eq. (4.10), with $\alpha = 1$, it is possible to maximize the available resources and have a load-balancing effect. Therefore, MLB dynamics are modeled with Φ_1. As we saw in the previous chapter, the association rule given by Eq. (4.10) implies a user association based on throughput, assuming a processor

sharing M/G/1 queue with a Round Robin scheduling policy, as it was established in Section 2.1.4 (please see Eq. (2.8)). Therefore, Eq. (4.10) is compliant with the objective of improving the throughput of the users in the cell's border, which is a major objective of ICIC.

- As it will be shown in Fig. 5.6, minimizing Φ_1 also reduces the interference levels created by the i-th cell and imposed on its neighborhood. Therefore, we claim the dynamics of ICIC are covered by Φ_1.

From Eq. (5.1), it is evident that an accurate estimation of the cell utilization is essential. In the following section, we use a well-established method for cell utilization estimation, which is compliant with the system model proposed in Section 2.1.4, and we adapt it to introduce a dynamic SFR scheme.

5.2.2 Cell Utilization Estimation

We consider an OFDMA network with a dynamic SFR scheme. As mentioned in Section 2.1.2, by considering a SFR scheme on the system, the coverage area of the cell is divided into two mutually exclusive parts, the center, and the edge. In each cell, the frequency bandwidth (BW) is divided into three adjacent orthogonal subbands. Two subbands are allocated to the center users, and the third subband is allocated to the edge users, as shown in Fig. 2.3.

We assume that the maximum transmit power for the i-th cell is p_i in Watts, with $\boldsymbol{p}$ being the (maximum) downlink power vector. As mentioned in Section 2.1.2, in SFR, every cell applies the reduction power factors η_i^c and η_i^e at the center and the edge respectively. Therefore, the transmit power for resources assigned to edge users is $\eta_i^e p_i$, whereas for the resources assigned to center users is $\eta_i^c p_i$. Let $\boldsymbol{\eta^c}$ be the power factor vector at the center of the cells, where the i-th coordinate η_i^c is the center power factor of cell i, and it takes values from a discrete set $\mathbb{D}_p$ with cardinality $|\mathbb{D}_p|$. Additionally, $\boldsymbol{\eta^e}$ is the power factor vector at the edge of the cells, where the i-th coordinate η_i^e is the edge power factor of cell i, and it also takes values from a discrete set $\mathbb{D}_p$ with cardinality $|\mathbb{D}_p|$.

Given the antenna tilt e_i, the propagation loss between cell i and an UE m is denoted by $g_{i,m}(e_i)$. Unlike the previous chapter, in this study, we will assume the tilt for all antennas is fixed; therefore, we drop the dependency of the losses to the downtilt. On the other hand, as we require high throughput and good coverage for users at the edge, we additionally fix the edge power to maximum: $(\forall i \in \mathcal{B})\eta_i^e = 1$ as in [18], [95].

Like in Chapter 4, replacing Eq. (2.2) into Eq. (2.4), we can write the Shannon capacity as Eq. (5.2), which is adapted here to consider SFR.

$$c_{i,m}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P}) = \mathrm{BWF}_{i,m}\mathrm{BW}\log_2\left(1 + \frac{\mathrm{PF}_{i,m}p_i g_{i,m}}{\sum_{k\neq i}^{B}\mathrm{PF}_{k,m}p_k g_{k,m}\rho_k + \sigma^2}\right) \tag{5.2}$$

where σ^2 is the noise power over the transmission bandwidth BW, and $\boldsymbol{\rho}$ is the cell utilization vector. Additionally, an user location-dependent bandwidth factor ($\mathrm{BWF}_{i,m}$) and power factor ($\mathrm{PF}_{i,m}$) are introduced and defined as follows:

$$(\forall i \in \mathcal{B})(\forall m \in \mathcal{S})\ \mathrm{BWF}_{i,m} = \begin{cases} 2/3 & m \in \mathcal{P}_i^c \\ 1/3 & \text{otherwise,} \end{cases}$$

$$(\forall i \in \mathcal{B})(\forall m \in \mathcal{S})\ \mathrm{PF}_{i,m} = \begin{cases} \eta_i^c & m \in \mathcal{P}_i^c \\ 1 & \text{otherwise,} \end{cases}$$

where $\mathcal{P}_i^c$ corresponds to the central portion of the i-th cell[2].

As we have already seen, for an arbitrary maximum power allocation $\boldsymbol{p}$, downtilt $\boldsymbol{e}$, data rate requirements $\boldsymbol{\gamma}$, and user association $\mathcal{P}$, the cell utilization is estimated by solving the equation system in Eq. (2.5), which is revised in Eq. (5.3):

$$(\forall i \in \mathcal{B})\ \rho_i = \sum_{l\in\mathcal{P}_i}\frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P})}, \tag{5.3}$$

As in Chapter 4, we want to estimate the impact of multiple user associations ($\mathcal{P}$) on the cell utilization vector ($\boldsymbol{\rho}$) without actually applying any change in the network. Therefore, we are interested in solving Eq. (5.3). The equation system in Eq. (5.3) is a set of nonlinear equations, which is implicitly defined because the cell utilization ρ_i is on both sides of the equation. The conventional way to solve such an equation system is through the use of FPI algorithms, like the standard one presented in Eq. (5.4):

$$\boldsymbol{\rho}^{(k+1)} = T(\boldsymbol{\rho}^{(k)}),\ \boldsymbol{\rho}^{(1)} \in \mathbb{R}_+^B \tag{5.4}$$

[2]Identifying the nominal cell range of a cell as well as the boundary between the cell center and edge is not an easy endeavor. In this chapter, the association of a UE to any of those regions is achieved by computing a normalized distance based on the separation between the cell and UE and the horizontal angle between transmitter and receiver.

where T is a vector-valued mapping given by:

$$
\begin{aligned}
T : \mathbb{R}_{+}^{B} &\to \mathbb{R}_{++}^{B} \\
\boldsymbol{\rho} &\mapsto \begin{bmatrix} \sum_{l \in \mathcal{P}_1} \frac{\gamma_l}{c_{1,l}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P})} \\ \sum_{l \in \mathcal{P}_2} \frac{\gamma_l}{c_{2,l}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P})} \\ \vdots \\ \sum_{l \in \mathcal{P}_B} \frac{\gamma_l}{c_{B,l}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P})} \end{bmatrix}
\end{aligned}
\tag{5.5}
$$

A solution to Eq. (5.5) is called a Fixed Point of the mapping T, and it is expressed as $\boldsymbol{\rho} \in \mathrm{Fix}(T)$.

5.2.3 Optimization Problem Formulation

Formally, the optimization problem is given by:

$$
\begin{aligned}
\underset{\mathcal{P}, \boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}}{\text{minimize}} \quad & \Phi_1(\boldsymbol{\rho}) \\
\text{subject to} \quad & \boldsymbol{\rho} \in \mathrm{Fix}(T) \\
& \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \\
& \boldsymbol{\eta}^{\boldsymbol{c}} \in \mathbb{D}_p^B
\end{aligned}
\tag{5.6}
$$

The first constraint ensures that the objective function is evaluated at the solutions to the system in Eq. (5.5), which corresponds to the definition of cell utilization in Eq. (5.3). The second constraint also follows the definition of utilization. The third constraint limits the center power factor in every cell to be within a discrete set $\mathbb{D}_p$.

5.3 A Hybrid Solution

The problem in Eq. (5.6) is complex due to the implicit formulation of the cell utilization, the choice of UE-cell associations $\mathcal{P}$, and the effects of the antenna downtilt on the received power, coverage, and cell utilization. However, exploiting the fact that for a fixed $\boldsymbol{\eta}^{\boldsymbol{c}}$, $\boldsymbol{e}$, $\boldsymbol{\gamma}$, and $\mathcal{P}$, the optimal utilization vector minimizing Φ_1 is given as the unique solution to $\boldsymbol{\rho} \in \mathrm{Fix}(T)$, as it will be seen in Section 5.3.1 (and it has been proven in [82], [53], [55]), and considering that the optimization over $\mathcal{P}$ (and $\boldsymbol{\rho}$) and the optimization over $\boldsymbol{\eta}^{\boldsymbol{c}}$ are differently structured optimization problems [55], we propose to use the hybrid architecture presented in Chapter 4, which is revisited in Fig. 5.1,

to solve maximize the utility function given by Eq. (5.7), which corresponds to the GMAR definition.

$$\mathcal{U} \coloneqq 10^{-\frac{\Phi_1(\rho)}{B}} \tag{5.7}$$

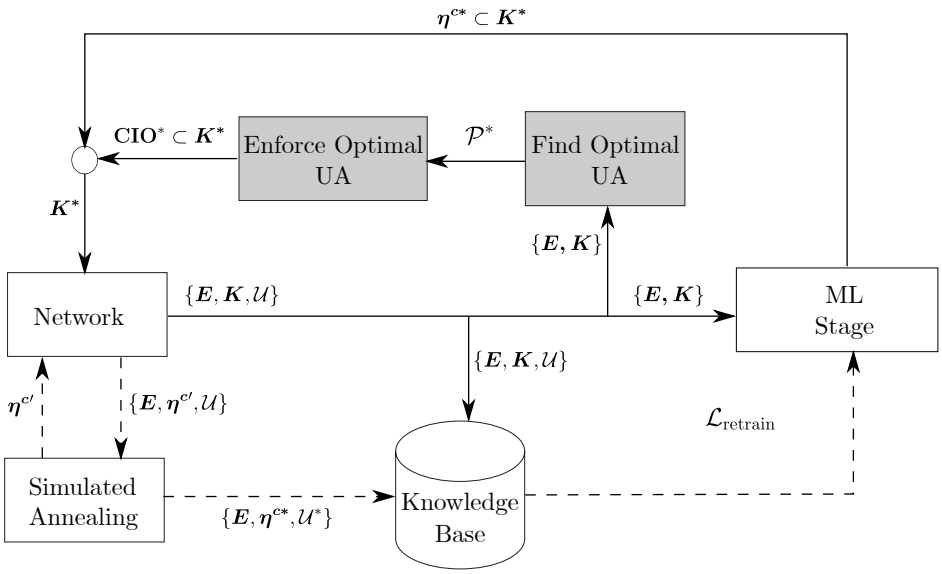


Figure 5.1: Global view of the proposed scheme.

The framework in Fig. 5.1 is designed to maximize Eq. (5.7).

The inner loop finds and enforces the optimal user association ($\mathcal{P}^*$), whereas the outer loop predicts the best possible power factors $\boldsymbol{\eta}^{\boldsymbol{c*}}$ in one shot. It is important to mention that both loops work in coordination, meaning that the global utility function in Eq. (5.7) is jointly maximized. As shown in Section 5.3.1, $\mathcal{P}^*$ is obtained using a FPI algorithm, and the resulting association is enforced in the network using the A3-HO condition.

For the outer loop, we propose to use a ML stage to directly predict $\boldsymbol{\eta}^{\boldsymbol{c*}}$ based on the environmental conditions $\boldsymbol{E}$. In order to train the ML model, it is needed to gather enough information about the tuple: $(\boldsymbol{E}, \boldsymbol{\eta}^{\boldsymbol{c*}})$. Therefore, as we did in Chapter 4, we use a "smart exploration" approach based on the SA heuristic, as shown in dashed lines in Fig. 5.1.

As in Chapter 4, one of the main advantages of our method is that even if the predicted power factors (obtained through the outer loop) are not optimal, the inner loop still provides the suitable CIO values for the predicted $\boldsymbol{\eta}^{\boldsymbol{c}}$.

5.3.1 Inner Loop: Finding and Enforcing $\mathcal{P}^*$

For a fixed power factor vector $\boldsymbol{\eta}^{\boldsymbol{c}}$, maximizing Eq. (5.7) over the set of $\mathcal{P}, \boldsymbol{\rho}$ yields:

$$\begin{aligned} \underset{\mathcal{P},\boldsymbol{\rho}}{\text{maximize}} \quad & \mathcal{U} \\ \text{subject to} \quad & \boldsymbol{\rho} \in \text{Fix}(T) \\ & \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \end{aligned} \tag{5.8}$$

As we already saw in Section 4.3.1, and according to [82], [83], [53], the solution to Eq. (5.8) gives a policy to associate every UE to a single cell (i.e., $\mathcal{P}^*$) [82]. Specifically, UE m should be associated with the cell k^*, with :

$$k^* = \underset{i \in \mathcal{B}}{\text{argmax}} \, (1 - \rho_i) \, c_{i,m}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P}) \tag{5.9}$$

Since ICIC aims to reduce the ICI, especially at the edge of the cell, improving the throughput of the edge UE, it makes sense to let the user association be driven by the association rule in Eq. (5.9). Additionally, according to Eq. (5.9), users should be associated with cells with lower utilization to experience an increased throughput, which is the main objective for MLB. Therefore, it is evident that the problem formulation is consistent with a SF coordination framework between ICIC and MLB. Thus, as mentioned before, the definition of the utility function as in Eq. (5.7) is enough to achieve the joint optimization of ICIC and MLB.

The association rule in Eq. (5.9) is used along FPI to find an optimal user association. Our implementation is proposed in Algorithm 4.

Algorithm 4: FPI for the calculation of the optimal user association

1: INPUT: μ,ϵ,$k = 0$,$\boldsymbol{\rho}^{(0)} \in \mathbb{R}_+^B$,$\beta$
2: **while** $\mu > \epsilon$ **do**
3: **for all** gNB $i \in \mathcal{B}$ **do**
4: calculate cell area: $\mathcal{P}_i^{(k)} = \left\{ m \in \mathcal{S} \mid i = \underset{j \in \mathcal{B}}{\text{argmax}} \left(1 - \rho_j^{(k)}\right) c_{j,m}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P}) \right\}$;
5: calculate new utilization: $T_i(\boldsymbol{\rho}^{(k)}) = \min \left\{ \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{\eta}^{\boldsymbol{c}}, \mathcal{P})}, 1 - \epsilon \right\}$;
6: **end for**
7: $\boldsymbol{\rho}^{(k+1)} = \beta \boldsymbol{\rho}^{(k)} + (1 - \beta) T(\boldsymbol{\rho}^{(k)})$
8: $\mu = \| \boldsymbol{\rho}^{(k+1)} - \boldsymbol{\rho}^{(k)} \|_2$, $k = k + 1$;
9: **end while**
10: OUTPUT:$\mathcal{P}_i^* = \mathcal{P}_i^{(k)}$,$\boldsymbol{\rho}^* = \boldsymbol{\rho}^{(k)}$,$\Phi_1 = \sum_{i=1}^{B} - \log(1 - \rho_i^{(k)})$, $\mathcal{U} = 10^{-\Phi_1/|B|}$

As the reader can appreciate, the association rule in Eq. (5.9) is implemented in

line 4 of Algorithm 4, the cell utilization estimation in Eq. (5.3) takes place in line 5, and the FPI is implemented in line 7 using an interpolated version rather than the standard one presented in Eq. (5.4). The outputs of the algorithm are the desired user association $\mathcal{P}^*$, the cell utilization imposed by the $\mathcal{P}^*$, and the maximum achievable GMAR.

To enforce $\mathcal{P}^*$ in the network, we use the method explained in Section 4.3.2. Specifically, we use Algorithm 2. It is possible to reuse the enforcement method of the previous chapter as it is taking place in the Radio Resource Control (RRC) layer, which is on top of layers 1 and 2, which are the ones that are mainly involved with the introduction of SFR.

5.3.2 Outer Loop: a Supervised ML Stage

To naively maximize $\mathcal{U}$ over $\boldsymbol{\eta^c}$, an exhaustive search could be carried out: first fixing $\boldsymbol{\eta^c}$, finding $\boldsymbol{\rho}^*$ according to Section 5.3.1, computing GMAR and finally selecting the value of $\boldsymbol{\eta^c}$ exhibiting the highest GMAR value ($\boldsymbol{\eta^{c*}}$). Unfortunately, such an approach is computationally expensive due to the high number of parameters, making it not tractable. Therefore, as we did in the previous chapter, we propose a ML-driven approach in the outer loop in Fig. 5.1, which predicts the best possible power factors $\boldsymbol{\eta^{c*}}$ based on the environmental conditions ($\boldsymbol{E}$). For the joint optimization of MLB and ICIC, $\boldsymbol{E}$ corresponds to the set of mean UE rate requirements (γ_m), which can change over time and the user distribution (uD_i), as defined in Section 3.3.

Because we have a multioutput prediction task, we again consider the classical multioutput learning domain, as explained in Section 4.3.3.1, only that in this case, every model predicts the power factor in the respective cell: η_i^c. Apart from the multioutput models, we also consider a FFNN like the one presented in Fig. 5.2.

Regardless of the selected ML model, its performance depends largely on the amount and quality of the information used to train it. As mentioned before, obtaining labeled data (i.e., with KPI information) is expensive in real networks. If generating the training set is not feasible using an exhaustive search, the training set can be generated with the output of a time-consuming heuristics like SA. As shown in dashed lines in Fig. 5.1, we obtained labeled data, i.e., $(\boldsymbol{E}, \boldsymbol{\eta^{c*}})$ using SA. However, the higher the number of parameters to be optimized (and therefore explored), the higher the "exploration" time of the heuristic. Considering that the total number of power factor combinations is given by the permutation with repetitions $|\mathbb{D}_p|^B$, the number of cells (B) or the cardinality of every output should be small to keep a reasonable exploration

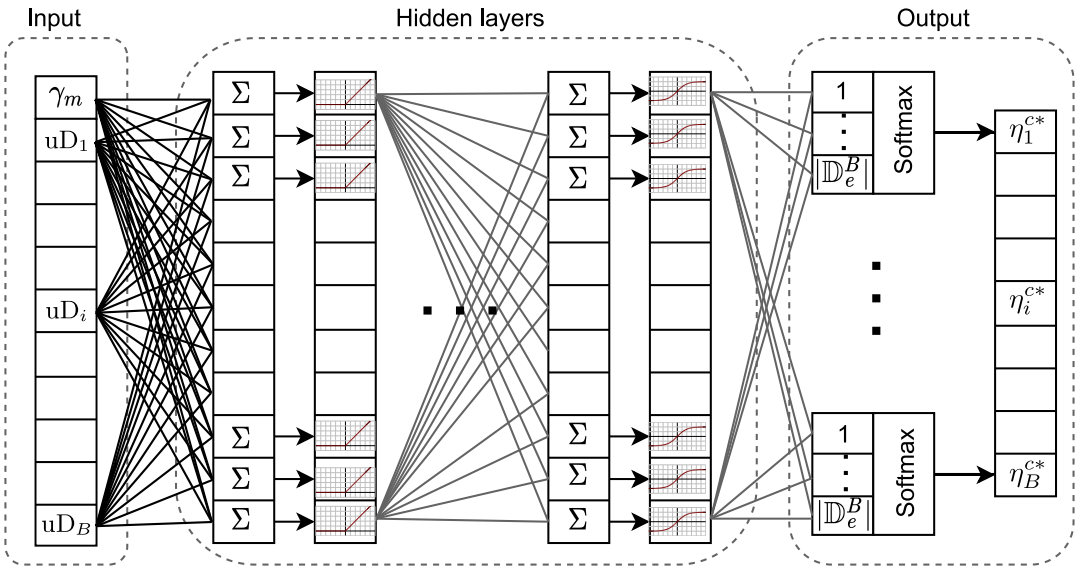


Figure 5.2: Feed forward neural network for a multioutput learning problem.

time. As we did in the previous chapter, we consider the latter option. The modified version of SA to get labeled data is presented in Algorithm 5.

5.4 Comparative Evaluation of Algorithms for joint MLB-ICIC Optimization

Multiple simulations were carried out using an OFDMA system-level simulator, considering 7 gNB each with 3 cells with the layout depicted in Fig. 3.7. Further details of the simulation parameters are given in Table 5.2.

To collect the training data, multiple network scenarios with randomly deployed users were simulated. For every scenario, both the distribution of the users (uD$_i$) and the requested data rate (γ_m) change, emulating the change in the traffic profile throughout the day. For every scenario, a search over the power factors using SA was carried out to learn the best possible tilt configuration $\boldsymbol{\eta}^{\boldsymbol{c*}}$. Once a large enough number of scenarios have been simulated, it is possible to train the ML stage in the outer loop in Fig. 5.1. There is a twist in this chapter though compared to the previous one, we collect 50% more information compared to the training set size.

As we did in the previous chapter, once the ML model is trained, the solution in Fig. 5.1 is ready to optimize MLB and ICIC jointly. To check the performance of the joint optimization, the simulation batch duration is set to 30 seconds. At the beginning of every batch, the user distribution (uD$_i$) and the requested data rate (γ_m) change,

Algorithm 5: Simulated Annealing for MLB and ICIC Optimization

INPUT: $\eta^{c}_{init}, i_{max}, \tau_{min}, \tau_{max}$
INITIALIZE: $\tau = \tau_{max}, \eta^{c} = \eta^{c}_{init}$
$U(\eta^{c}) \leftarrow$ Compute utility in (5.7)
while $\tau > \tau_{min}$ **do**
  for all $i = 1$ to i_{max} **do**
    $U(\eta^{c'}) \leftarrow$ Compute utility in (5.7) for $\eta^{c'} \in \mathbb{D}^{B}_{p} \setminus \eta^{c}$
    $\Delta(U) = U(\eta^{c'}) - U(\eta^{c})$
    flag=1
    if $\Delta(U) > 0$ **then**
      Accept $\eta^{c} \leftarrow \eta^{c'}$
    else
      Calculate probability: $\Pr(\Delta(U)) = e^{\frac{\Delta(U)}{\tau}}$
      if $\Pr(\Delta(U)) > \mathrm{rnd}(0,1)$ **then**
        Accept $\eta^{c} \leftarrow \eta^{c'}$
      else
        Reject $\eta^{c} \leftarrow \eta^{c'}$
        flag=0
      end if
    end if
    if flag=1 **then**
      Update $\eta^{c*} = \eta^{c}$
    end if
  end for
  $\tau = \frac{\tau}{log(i+1)}$
end while
OUTPUT:$\boldsymbol{\eta}^{\boldsymbol{c*}}$

Table 5.2: Simulation parameters

Type	**Parameter**	**Value**
System specifics	Inter-site distance	500 [m]
	Pathloss	See Eq. (2.3)
	Shadowing	See Eq. (2.3), Ω = 6[dB]
	p_i (Eq. (5.2))	46 [dBm]
	$\mathbb{D}_p$	$\{0.5, 0.8\}$
	Tx antennas	gain 15 [dBi], height = 32 [m]
	BW	10 [MHz]
	σ^2 (Eq. (5.2))	-95 [dBm]
	Scheduler	Round Robin
User specifics	Number of users	240
	Mobility model	Random Walk
	UE positioning	uniform distribution
	UE antennas	gain 2 [dBi], height = 1.5[m]
	γ_m	100-800 [kbps]
Algorithm specifics	H (Eq. (4.11))	3.0 [dB]
	Δ	0.1 [dB]
	CIO_{max}	12.0 [dB]
	α (Eq. (4.1))	1
	β (Algorithm 4)	0.9
	τ_{min}, τ_{max}	0.0001 , 100
	i_{max}	100
Sim. period	After training	30 [s] (Except for SA)

and based on these environmental variables ($\boldsymbol{E}$), an appropriate power factor vector is predicted using the outer loop of Fig. 5.1 and applied afterward. Soon after the change in the power factor, we allow the registration of UEs to the respective cells and the generation of data traffic. At the 10-th second, once we have a stable estimation of the cell utilization (ρ_i), the user association optimization is triggered using Algorithm 4. The resulting association ($\mathcal{P}^*$) is enforced afterward using Algorithm 2.

A subset of $\gamma_m : \{100, 200, 300, 400, 500, 600, 700, 800\}$ [kbps] is used to evaluate the performance of the proposed framework over multiple traffic conditions. As we did in the previous chapter, two benchmark schemes are considered to compare the benefits of the proposed framework, namely:

1. SA-based heuristic. This long-lasting heuristic can be used not only to collect the training data but also to predict a suitable power factor vector ($\boldsymbol{\eta}^{c*}$).

2. The best fully-implicit optimization mechanism in Chapter 3, corresponding to the grouping of SOM and hierarchical clustering, using the architecture in Fig. 3.3.

A brief discussion about the performance of every method is given in the following subsections. Afterward, we show the network performance comparison among all the schemes.

5.4.1 Simulated Annealing Approach

Apart from collecting data for training, SA (as in Algorithm 5) can also be used as a (long-lasting) method to find the suitable power factors ($\boldsymbol{\eta}^{c*}$). That is, SA could replace the ML stage in the outer loop. For a specific user distribution[3], and for the different values of γ_m, the heuristic was in charge of maximizing the utility function $\mathcal{U}$ defined as in Eq. (5.7) through modifying the power factor vector. The evolution of $\mathcal{U}$ for different values of γ_m is depicted in Fig. 5.3.

We can see that for every value of γ_m that was considered, the proposed version of SA finds a suitable configuration provided it was given a high enough number of iterations (n) and solution perturbations. Please recall that for every value of n, multiple perturbations of the solutions are allowed ($i_{\max}$ in Table 5.2). The impact of the obtained configurations on the network performance will be supplied in Section 5.4.4.

[3] That was not explored during the training data collection, i.e., a "test" user distribution.

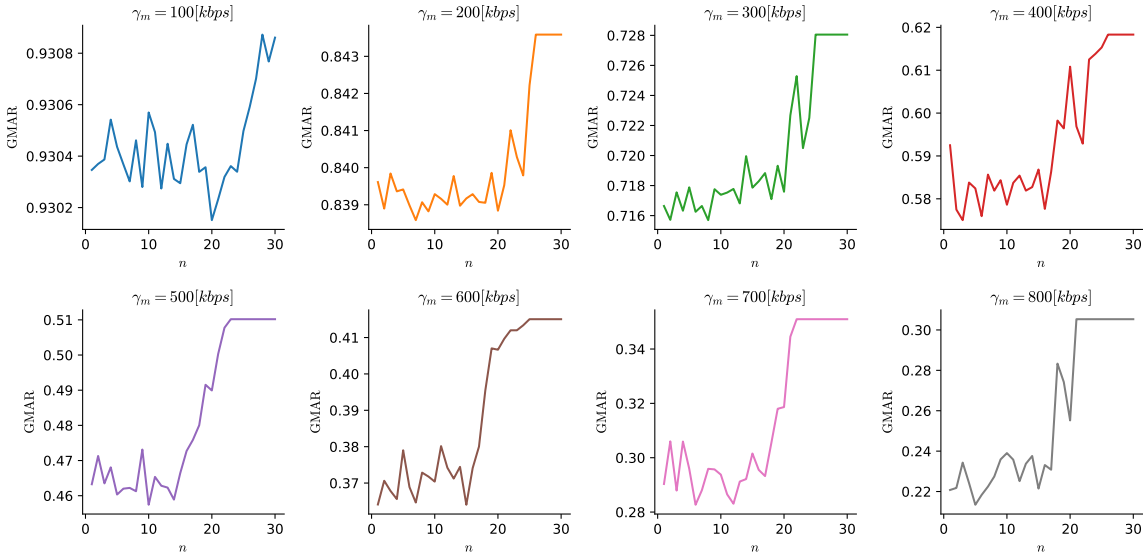


Figure 5.3: Improvement of the GMAR over the time

5.4.2 Fully-Implicit Optimization Approach

Regarding the fully-implicit coordination approach, we found in Chapter 3 that the best approach was the combination of SOM and HC. In this chapter, we consider the following functional domains: the main KPI ($\mathcal{U}$) as in Eq. (5.7), the controllable part corresponds to the power factor vector ($\boldsymbol{\eta}^c$) and CIO values (**CIO**), whereas the environmental part corresponds to the requested data rate (γ_m) and user distributions (uD_i).

As in Chapters 3 and 4, a three-layer SOM is trained, and the hyperparameters ruling the model behavior, i.e., x_{dim} and y_{dim}, are found using a grid search procedure. The results are available in Table 5.3. The best combination corresponds to $\text{x}_{\text{dim}} = 5$ and $\text{y}_{\text{dim}} = 10$. As we have more training data and also the number of units in the SOM is lesser than in Chapter 4, it is expected that we do not have that many problems with empty units. As a matter of fact, the quality plot of the map (which shows the mean distance of objects mapped to a unit to the neuron signature [77]) is depicted in Fig. 5.4(a). Please recall that the smaller the distances, the better the objects are represented by the neuron's signature [77]. Compared with the SOM in Fig. 4.10, in this case, we do not have gray units (empty neurons).

After finding the best SOM model, hierarchical clustering is applied considering only the environmental variables. As indicated in Appendix A, the Elbow method using WSS is used to determine the number of clusters. A cluster size of 8 is selected, as shown in Fig. 5.4(b).

Table 5.3: Grid search for SOM representation

x_{dim}	y_{dim}	RMSE	R^2	MAE
5	5	0.1065	0.8539	0.0841
5	10	0.1060	0.8553	0.0840
5	20	0.1079	0.8500	0.0846
10	5	0.1033	0.8627	0.0817
10	10	0.1059	0.8554	0.0836
10	20	0.1464	0.7231	0.1154
20	5	0.1077	0.8505	0.0850
20	10	0.1541	0.6943	0.1213
20	20	0.1784	0.5905	0.1411

Propagating the 8 clusters to the KPI layer in the SOM grid (as explained in Fig. 3.5), we obtain Fig. 5.4(c). We can see from Fig. 5.4(c) that the optimization in the low-dimensional corresponds to moving towards the right-most bottom corner at step sizes given by the clusters.

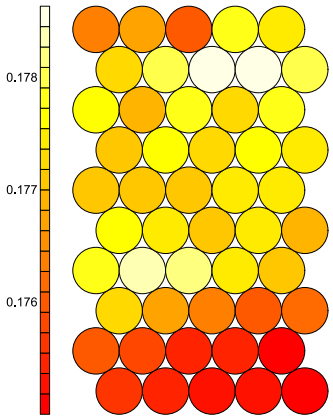
5.4.3 Hybrid Architecture

5.4.3.1 Inner Loop

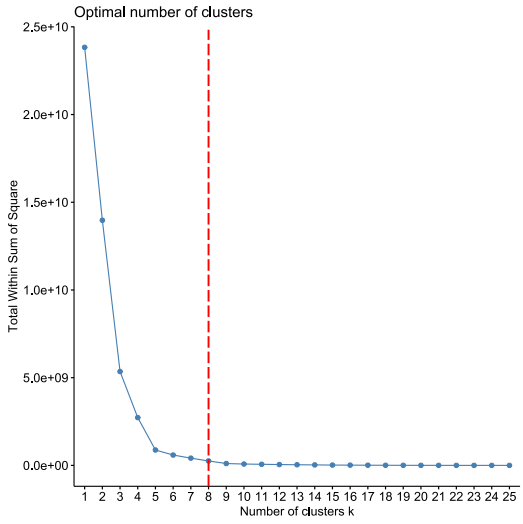
Because the behavior of the cost function and the GMAR, which are calculated using Algorithm 4, are monotonically decreasing as in Section 4.4, we focus directly on the performance of Algorithm 2 while enforcing the user association $\mathcal{P}^*$.

To gain insights about the changes that are taking place under the hood in the network with the enforcement of $\mathcal{P}^*$, Table 5.4 shows the average matching rate between the optimal ($\mathcal{P}_i^*$) and the final associations in the network (i.e., after the execution of Algorithm 2), the average number of modified adjacencies, as well as the maximum offset value that was applied to an adjacency. As it can be seen from the second column of the table, the matching rates are almost perfect under reasonable changes of CIO, which implies that Algorithm 2 exhibits consistent behavior for the considered values of γ_m.

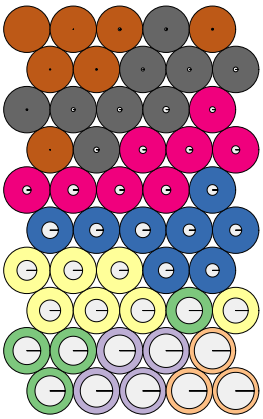
From Table 5.4, we observe that Algorithm 2 strives to do the best it can with the constraints given in Table 5.2, i.e., the values for Δ and CIO_{max}.



(a) Quality metric of the 3-layer SOM



(b) Selected number of clusters



(c) 3-layer SOM and Hierarchical clustering on the KPI layer

Figure 5.4: SOM and hierarchical clustering

Table 5.4: Performance of the Algorithm 2

γ_m [Kbps]	**Matching %**	**Mod. Adjacencies**	max{$CIO_{i,j}$} [dB]
100	100	6	7.5
200	100	5	6.2
300	100	4	3.1
400	100	4	2.4
500	99.58	5	3.8
600	100	6	7.5
700	100	9	7.5
800	99.58	12	8.6

5.4.3.2 Impact on the Network Performance of the Inner Loop

From the network perspective, it is possible to see to what extent the chaining of the shaded boxes in Fig. 5.1 improves the performance. For a specific instance of $\boldsymbol{\eta}^{\boldsymbol{c}}$ predicted in the outer loop, the impact on the average cell SINR for the considered γ_m values is depicted in Fig. 5.5.

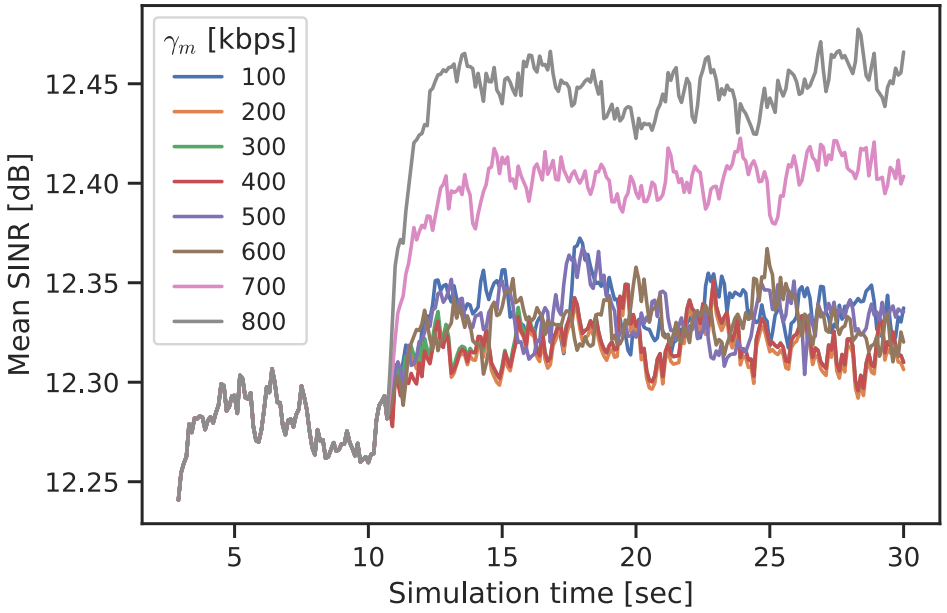


Figure 5.5: Mean network SINR.

As indicated in Fig. 5.5, once the HOs are executed after triggering Algorithm 4 and Algorithm 2 (once TTT has expired), it is possible to see an increment in the mean cell SINR profile, especially when γ_m is higher.

Another significant impact on the network performance is evident when the level of interference created by one cell (and imposed on the UEs of other cells) is accounted for, namely: $\text{createdIf}_i := \frac{\sum_{m \notin \mathcal{P}_i} \text{PF}_{i,m} p_i g_{i,m}}{|\{m \notin \mathcal{P}_i\}|}$. The average created interference for the different values of γ_m is depicted in Fig. 5.6.

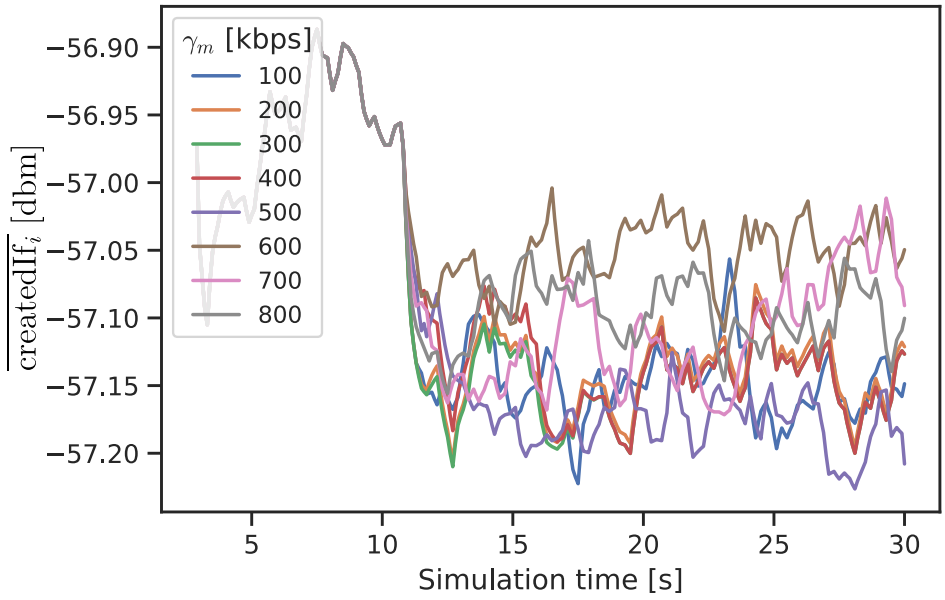


Figure 5.6: Mean interference level generated (createdIf_i).

It is evident that after the 10th second, the ICI is alleviated. One important takeaway appears here: with the sole definition of the utility as in Eq. (5.7) and the user association being driven by $\mathcal{P}^*$ from Algorithm 4, it is possible to reduce ICI. Thus, while minimizing Φ_1, we guarantee joint optimization of ICIC and MLB.

Please notice that we have not modified $\boldsymbol{\eta}^{\boldsymbol{c}}$ yet, so we claim that with an additional adjustment of $\boldsymbol{\eta}^{\boldsymbol{c}}$, we can improve (even further) the ICI levels. The hybrid approach is also tailored to achieve the best performance during the prediction of $\boldsymbol{\eta}^{\boldsymbol{c*}}$ using the most suitable ML Stage in Fig. 5.1.

5.4.3.3 Outer Loop

To discover the convenient model for the ML stage in Fig. 5.1, several ML models were considered for predicting $\boldsymbol{\eta}^{\boldsymbol{c*}}$, including neural networks (like the one in Fig. 5.2, and multioutput models (see Appendix D).

A grid search to find the appropriate hyperparameters per model was carried out based on the "micro-averaged" version of the area under the ROC curve. A deep explanation of the considered models and their main hyperparameters is not within the scope of this thesis, but we kindly refer the reader to [76].

In Table 5.5, it is possible to see the performance of the best parameters per model (the considered hyperparameters for each model are not shown in the table).

Table 5.5: Grid search for different multi-learning strategies

Type	Model	Area Under ROC Curve
Multioutput classifier	Gradient Boost	0.7263
	K-Neighbors	0.7150
	Random Forest	0.6673
	Extremely Randomized Trees	0.6455
	Multi-layer Perceptron	0.7312
Chain classifier	Logistic Regression	0.7146
	K-Neighbors	0.6714
	Ridge Classifier	0.7106
	Random Forest	0.6755
	Extremely Randomized Trees	0.6558
	Quadratic Discriminant Analysis	0.6338
	Nearest centroid classifier	0.6220
Deep Learning	FFNN (see Fig. 4.6)	0.7521

According to Table 5.5, in this opportunity, the best model happens to be a FFNN[4]. Since $\mathbb{D}_p = \{0.5, 0.8\}$ (see Table 5.2), we only have two options for the tilt of every cell. Therefore, the output layer in Fig. 4.6 uses sigmoidal activation functions[5].

5.4.4 Performance Comparison

For a comprehensive assessment of the hybrid framework, a study in steady-state is carried out to fairly compare the selected schemes. In that sense, we define some counters to compare the different approaches in this section.

We say the m-th user is in the center of the i-th cell if $m \in \mathcal{P}_i^c$. Likewise, the m-th user is at the edge of the i-th cell if $m \in \mathcal{P}_i^e$. The number of users in the center of the i-th cell is given by $|\mathcal{P}_i^c|$ whereas the number of users in the edge of the i-th cell is given by $|\mathcal{P}_i^e|$. Let $\mathcal{N}(i)$ represent the set of neighbor cells for the i-th cell. The following KPIs are used to evaluate the benefits of the proposed approach in this chapter:

[4]The grid search procedure suggested a neural network with 3 hidden layers, each with 30 neurons with a *softsign* activation function and *adam* optimizer.

[5]When more than two categories of power factors need to be predicted, the activation function should be softmax. In contrast, if a continuous value is required, a linear activation function should be selected.

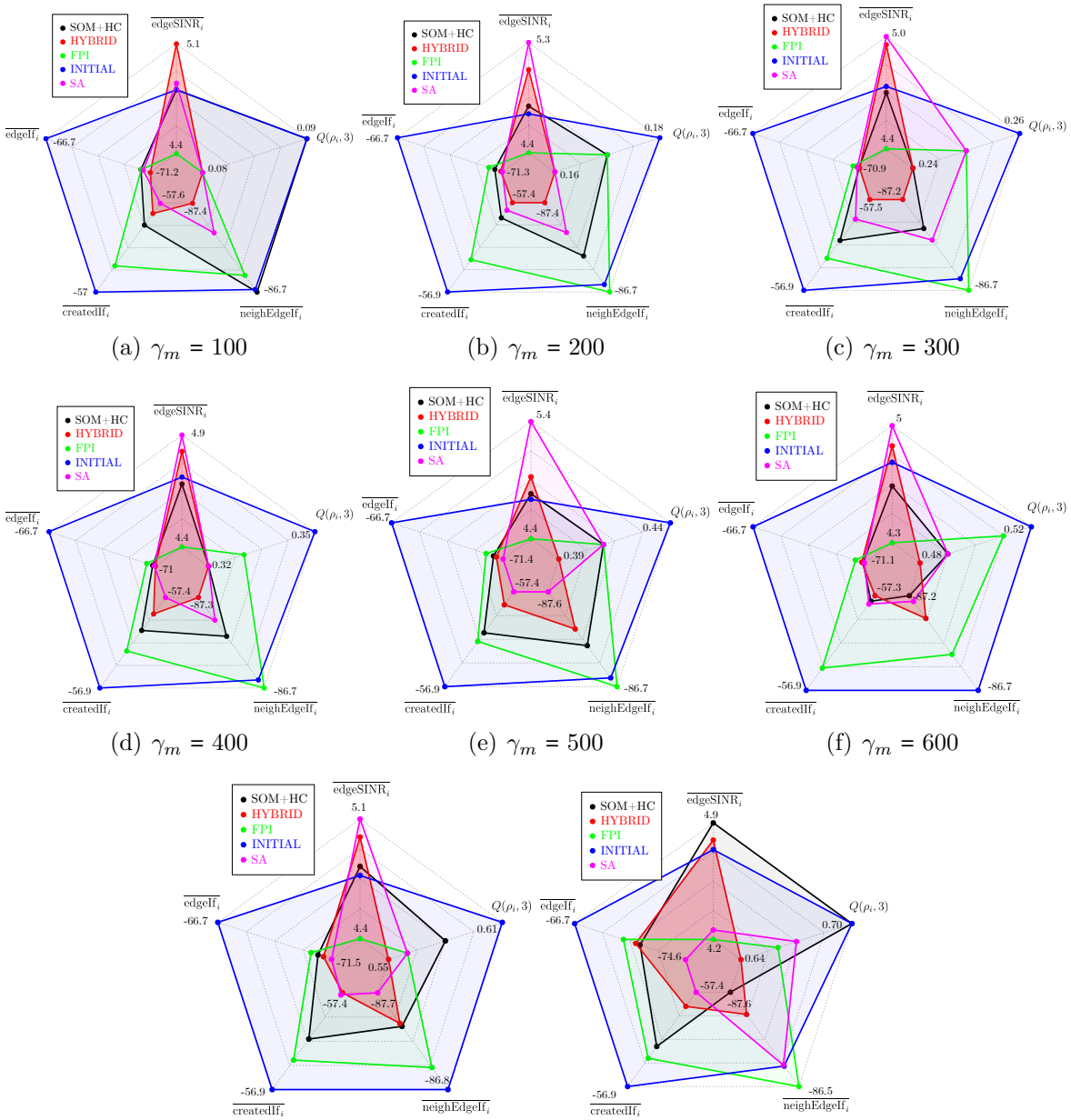
1. Edge SINR in i-th cell (to be maximized): defined as the mean SINR perceived by the UEs located at the edge of the i-th cell. Formally: $\text{edgeSINR}_i \coloneqq \frac{\sum_{m \in \mathcal{P}_i^e} \text{SINR}_m}{|\mathcal{P}_i^e|}$
2. Edge interference in i-th cell (to be minimized): accounts for the total received signal strength from all the surrounding cells excluding the serving cell i (from now on If) measured by an UE located at the edge of the i-th cell. Formally: $\text{edgeIf}_i \coloneqq \frac{\sum_{m \in \mathcal{P}_i^e} \text{If}_m}{|\mathcal{P}_i^e|}$
3. Interference created by the i-th cell (to be minimized): the name is self-explanatory. Formally: $\text{createdIf}_i \coloneqq \frac{\sum_{m \notin \mathcal{P}_i} \text{PF}_{i,m} p_i g_{i,m}}{|\{m \notin \mathcal{P}_i\}|}$
4. Edge interference in the neighborhood of the i-th cell (to be minimized): defined as the interference perceived by the users located at the edge of the neighbor cells of the i-th cell. Formally: $\text{neighEdgeIf}_i \coloneqq \frac{\sum_{j \in \mathcal{N}(i)} \sum_{m \in \mathcal{P}_j^e} \text{If}_m}{\sum_{j \in \mathcal{N}(i)} |\mathcal{P}_j^e|}$
5. .75-Percentile of the cell utilization (to be minimized): $Q(\rho_i, 3)$

The following methods are compared: INITIAL corresponds to a static configuration obtained during the exploration done by SA when $n = 20$ (see Fig. 5.3); FPI corresponds to activating only the inner loop in Fig. 5.1; SA corresponds to using both loops in Fig. 5.1, being the outer loop driven by SA; SOM+HC represents the fully-implicit optimization approach proposed in Chapter 3 (i.e., Fig. 3.3); finally HYBRID represents the scenario in Fig. 5.1 with the ML Stage being a FFNN solving the multioutput learning problem in the outer loop. The mean values of the above KPIs over all the cells for multiple values of γ_m are shown in Fig. 5.7.

From Fig. 5.7, it is evident that the HYBRID method (the red-filled polygons) proposed in this chapter outperforms all the other methods since it is ranked first (or second) in almost all the scenarios for all the considered performance counters. Whenever the HYBRID method was beaten, the winner was the long-lasting SA heuristic. Therefore, we can deduce that sometimes the outer loop of the hybrid approach does not perform as expected. This could be explained if we consider the performance in Table 5.5 which should be close to 1 in an ideal scenario. Please remember that the ground truth for training the neural network comes from many executions of SA before the online execution of the hybrid approach on never-seen-before network scenarios.

5.4.5 Final Comments on Energy Savings

It is worth mentioning that the proposed solution is extendable to consider the ES function, as suggested in Fig. 5.8, in which the median of the power factors (η_i^c) has



(a) γ_m = 100 (b) γ_m = 200 (c) γ_m = 300

(d) γ_m = 400 (e) γ_m = 500 (f) γ_m = 600

(g) γ_m = 700 (h) γ_m = 800

Figure 5.7: Network performance evaluation

been calculated from the ML model. It is clear that for lower values of γ_m, some cells are set with the lowest value of power in the set $\mathbb{D}_p$ (i.e., 0.5), whereas the maximum value (0.8) will be set as γ_m increases. Therefore, if we extend $\mathbb{D}_p$ to consider zero as a valid power factor, some cells could be switched off when the user's traffic demand is low. That is precisely what is expected from ES according to Section 2.2. Please notice that the dips in the curve for higher values of γ_m represent underperformance of the prediction done by the FFNN.

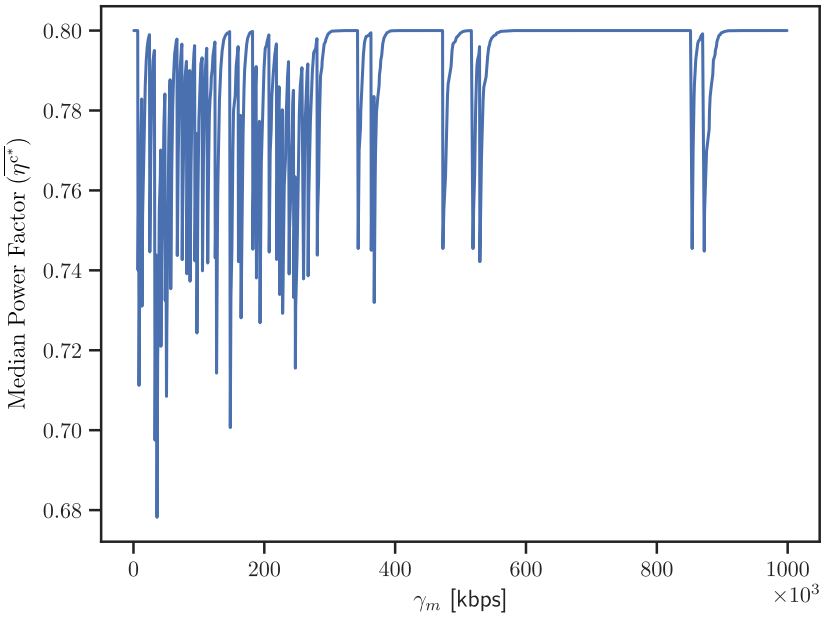


Figure 5.8: Median of η_i^c over all the cells.

5.5 Summary

A machine learning-driven optimization technique has been presented, which is applicable to mobile network scenarios, given the knowledge of propagation loss values and traffic signatures. We assume that this information is attainable, i.e., a function like MDT is active in the network.

We introduced a machine learning stage that predicts the power factor configuration from data rate requirements and user distribution information, whereas a fixed point algorithm finds the optimal user association given a predicted power configuration. One of the advantages of the proposed method is that it predicts the power configuration non-iteratively, which is an important factor for networks that should adapt quickly to the surrounding changes.

Another advantage is that the fixed point algorithm guarantees the best user association given any power configuration prediction (regardless of whether it is optimal or not). The method has been shown beneficial regarding the level of interference created, composite network capacity indicators, and energy savings.

As in previous chapters, the main limitation of the proposed solution is to obtain the amount of labeled data that is needed to train the ML Stage in the outer loop of Fig. 5.1. We have seen that chain classifiers are more data-efficient than neural networks (at least for the scenarios considered in this document) in the sense that they outperform neural network models when the amount of training data is not that big (see Section 4.4.3.4). Nevertheless, we have mentioned in Section 4.5 that there are methods to deal with the lack of labeled data that are well suited for neural network models, like *transfer learning* or *unsupervised pre-training*.

Another challenge is related to the SFR scheme implementation per se, and it comes to the difficulty of defining a border between the edge and the center of the i-th cell. In fact, simply identifying the cell coverage is a complicated task. In this chapter, we have considered a normalized distance, calculated based only on the distance and angle in the azimuth plane between UEs and gNBs, and compared that normalized distance to a fixed threshold, known as Edge-to-Centre Boundary (ECB). When the normalized distance happens to be larger than ECB, the UE is considered to be in the edge ($m \in \mathcal{P}_i^e$) or the center ($m \in \mathcal{P}_i^c$) otherwise. We claim that different strategies should be considered to account for the radio losses rather than the geographic location of the UEs with respect to the base stations.

6 Conclusions and Future Work

Throughout this document, we have presented different architectures for jointly optimizing several SFs in a centralized manner. These architectures are fully compliant with the specifications of SDOs as well as their perspectives in terms of design, requirements, and expected research directions. We have proven that it is possible to coordinate multiple SFs in a fully-implicit manner using a functional decomposition of the problem in three domains: controllable, environmental, and utility planes (provided there is enough data for training). This kind of approach is especially useful in network management scenarios in which the network owner has no deep knowledge about network optimization, which could be the case in campus networks or N-PLMNs or in scenarios in which the network optimization becomes that interleaved that closed-form solutions or classical optimization heuristics are impossible to use.

We have shown that the associated high dimensional optimization problem can be rendered in a low dimensional space in which the optimization process becomes human-readable, and therefore it can be formulated as a simple lookup procedure in the low dimensional space (based on the current environmental conditions of the network) followed by controlled changes in the parameter set.

We demonstrated that it is possible to use this fully-autonomic approach by chaining multiple ML models, provided there is a *large enough*[1] amount of data in the OSS. Typically that is the case if the operators store FCAPS information in a data lake. However, we are well aware that the stored data could not have that much variance, especially in the controllable part. To elaborate a bit more on this regard, please consider the following hypothetical scenarios:

- If a suboptimal configuration was accidentally set in a real network and there was QoS degradation, a rollback should occur as soon as possible (limiting the number of samples gathered during misconfiguration).
- In network deployments where an online exploration of parameters is not viable, the operators will probably keep the default configuration, usually provided by

[1] What large enough means in this domain still needs to be characterized.

the vendors.

Based on these scenarios, we can deduce that the datasets from a real network could be imbalanced or sparsely populated. Additionally, it is evident that getting more labeled data is expensive in the real world (no operator is willing to trade user unsatisfaction to get online insights about inappropriate configuration sets). The tradeoff between spending more resources collecting more information or developing more sophisticated ML models has already been discussed [96, 97]. Fortunately, there are promising methods (especially tailored for neural networks) to face the lack of training data, namely *transfer learning* and *unsupervised pretraining*, which should be visited in future work.

Strong assumptions were also made considering the environmental variables, for instance, about the availability of the mean velocity of a user in a specific cell. We claim variables like that could be calculated through proxy variables, for instance, the permanency time in a cell (which could be obtained from Call Detail Records - (CDR) information) and the approximate cell size.

Solving the coordination problem in a fully-implicit and centralized way also imposed a high dimensionality on the framework in case we want to take advantage of the global visibility of the variables (gaining observability of intra- and inter-cell conflicts) and set the complete parameter set at once. Therefore, that approach comes with the curse of dimensionality, which is an issue for many ML models. Fortunately, models like perceptrons and neural networks are robust to that problem. That is why throughout the whole thesis, we tried to land in the neural network domain, even though, for training a deep neural network, more training data is required due to the number of parameters (weights in the neural network that need to be found), especially for fully connected networks.

To reduce the dimensionality of the joint optimization problem, we modified the fully-implicit approach and proposed a hybrid framework in which the high-dimensional MLB dynamics are characterized in a closed-form manner (finding a fixed point in the so-called *load estimation problem* [98]) whereas other SFs (like CCO, ICIC, and ES) are optimized using ML. Of course, both paths are interleaved in a single optimization task using a common utility function. While jointly optimizing the utility function, it is possible to achieve implicit coordination among the SFs.

We observed that the hybrid approach outperformed the fully-implicit framework and the selected benchmark schemes assuming the same amount of information, meaning that the hybrid approach is more data-efficient for this type of problem. It is essential to keep in mind that while optimizing mobile networks using ML, the scarcity

of data for training is almost as important as the scarcity of radio resources.

For future work, it could be interesting to solve the so-called *power estimation problem* [98, 99]. To illustrate the formulation of this problem, the cell utilization estimation is presented again in Eq. (6.1).

$$(\forall i \in \mathcal{B})\ \rho_i = \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{p}, \mathcal{P})}, \tag{6.1}$$

Multiplying both sides of Eq. (6.1) by p_i/ρ_i, we end up with a system of non-linear equations implicitly defined with respect to the power allocation ($\boldsymbol{p}$):

$$(\forall i \in \mathcal{B})\ p_i = \frac{p_i}{\rho_i} \sum_{l \in \mathcal{P}_i} \frac{\gamma_l}{c_{i,l}(\boldsymbol{\rho}, \boldsymbol{p}, \mathcal{P})}, \tag{6.2}$$

As shown in [99], it is possible to solve this system using FPI, as we have done throughout this document. We claim that solving that problem gives a closed-form way of dealing with ICIC and ES since the power vector estimation can be carried out by finding the fixed point of a so-called standard interference mapping as proposed in [99], in which the implicit nonlinear system in Eq. (6.2) is solved for the power vector (rather than the utilization vector). The reasoning behind this is that it has been proven that the data rate requirement of users (γ_m) can be satisfied with lower transmit energy if we allow the cell utilization to increase (we sacrifice the load balancing to some extent). Additionally, acceleration methods for the FPI are proposed in [98] and could be helpful, especially if the network size increases.

In addition, for future work, it is interesting to consider different values for α in the α-fairness function used in Chapters 4 and 5. Of particular interest is the scenario when $\alpha = 2$. As we saw in Section 4.2.1, if $\alpha = 2$, the delay associated with the scheduling should be reduced. To prove this, the inner loop in our hybrid approach is applied to the network layout we used throughout this thesis for several values of α. During the first two seconds of simulation, all the UEs are attached to the network and the application traffic is triggered. At the second 2, all the CIO values in the network are intentionally shuffled to simulate a hypothetical suboptimal state. Five seconds later (i.e. second 7), the inner loop is triggered. The results considering the delay measured in the Packet Data Convergence Protocol (PDCP) layer are shown in Fig. 6.1. It is evident that minimizing Φ_2 has an impact on the latency as measured in the upper layers. We claim this could be useful for delay-sensitive traffic, e.g., URLLC scenarios in 5G.

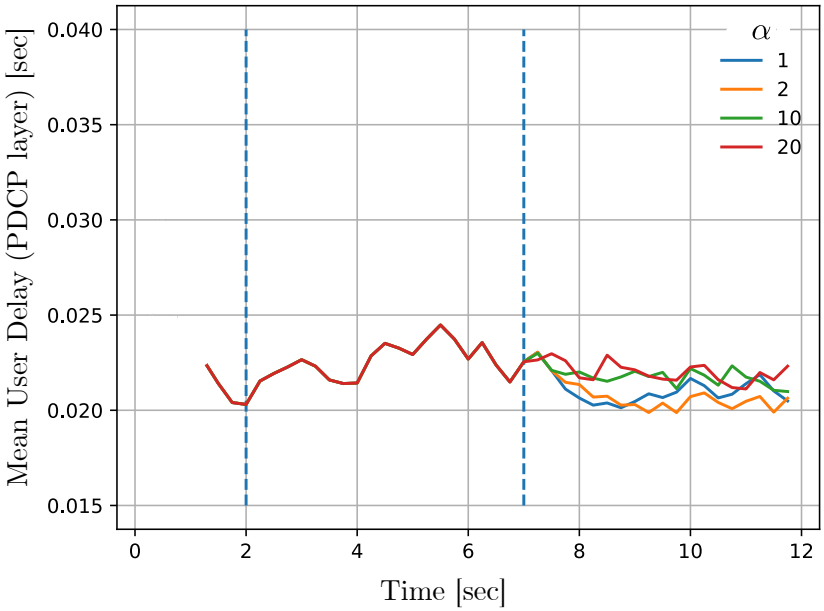


Figure 6.1: Mean user delay, as seen in the PDCP layer.

Even though we only focus on mobility optimization (i.e. MRO) in Chapter 3[2], we claim it is possible to incorporate MRO into the hybrid approach. To do so, we further constrain the optimization task to find the best possible amount of "overlap" of the received power signatures from two neighboring cells. The overlap should be large enough to avoid RLFs due to too-early HOs but should not be that large, in order to avoid RLFs due to too-late HOs. Formally, the overlap region (O_i) of the i-th cell with its neighbors is defined as:

$$(\forall i \in \mathcal{B}) O_i \coloneqq \frac{1}{\mid \mathcal{P}_i \mid} \sum_{m \in \mathcal{P}_i} \mathbb{1}_{im} \tag{6.3}$$

with

$$(\forall i \in \mathcal{B})(\forall m \in \mathcal{S})\ \mathbb{1}_{im} \coloneqq \begin{cases} 1 & \exists\, i' \neq i \ \mid\ |p_i g_{im} - p_{i'} g_{i'm}| \leq H_i, i' \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

$\mathbb{1}_{im}$ in Eq. (6.4), checks whether the received power from (at least) two different cells i, i' is good enough so any of both cells can be selected as a destination cell in potential HO procedure. Therefore, to consider HO optimization, we could formulate the global optimization problem as in Eq. (6.5).

[2]The main reason is that nowadays, around 95% of internet data volume is generated by static or quasi-static users.

$$
\begin{aligned}
\underset{\mathcal{P},\boldsymbol{\rho},\boldsymbol{e},\boldsymbol{H}}{\text{minimize}} \quad & \Phi_\alpha(\boldsymbol{\rho}) \\
\text{subject to} \quad & \boldsymbol{\rho} \in \text{Fix}(T_{\boldsymbol{p},\boldsymbol{e},\boldsymbol{\gamma},\mathcal{P}}) \\
& \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1} \\
& \boldsymbol{e} \in \mathbb{D}_e^B \\
& C(\boldsymbol{e}) \geq \text{C}_{\text{min}} \\
& \mathbf{O_{min}} \leq \mathbf{O}(\boldsymbol{e}, \boldsymbol{H}) \leq \mathbf{O_{max}}
\end{aligned}
\tag{6.5}
$$

An option to deal with the optimization proposed in Eq. (6.5) would be to solve a surrogate optimization problem in which a utility function ($\mathcal{U}$) is enhanced to consider the overlap degree, as we did in Chapter 4. With this formulation, we claim the proposed hybrid approach covers the coordination of all the canonical self-optimization SFs.

Of course, the dynamics among the variables in the problem in Eq. (6.5) are more interleaved now and the parameter space is larger, so more exploration for training the outer loop of the hybrid approach would be needed. Therefore more runtime is necessary in order to get meaningful training data. The challenge is two-fold now: sacrificing the runtime to gather more training data and finding methods for learning from smaller sets.

Appendix

A Self-Organizing Maps and Hierarchical Clustering

Self-Organizing Maps (SOM) in the SF coordination scenario are used to translate a high-dimensional optimization problem to a low-dimensional space in which the "redefined" optimization problem can be posed as a simple lookup. This is possible due to the fact that the dimensionality reduction is not naive (it is not a classical unsupervised dimensionality reduction method), in the sense that it also considers information about the target variable to accommodate the information in the low-dimensional space (what is called a supervised or semi-supervised dimensionality reduction). We claim the redefinition of the joint optimization problem is feasible because it is possible to reuse a parameter set ($\boldsymbol{K}$) in multiple environmental conditions ($\boldsymbol{E}$). Likewise, multiple combinations of different environmental ($\boldsymbol{E}$) and controllable variables ($\boldsymbol{K}$) could yield the same KPI values ($\mathcal{U}$). As a matter of speaking, we "cluster" the $\boldsymbol{K}$-$\boldsymbol{E}$-$\mathcal{U}$ information in a supervised manner, i.e., without losing information about the underlying model $h(\cdot)$.

SOM corresponds to a low-dimensional projection and visualization technique that compresses information while preserving the most important topological relationships of the input data. They are a special kind of ANN that can handle high-dimensional datasets with little a priori information or assumptions concerning the statistical distribution of the input data. It exhibits a shallow architecture consisting of an input layer and a two-dimensional (2D) grid as the output layer whose dimensions are $\mathrm{x}_{\mathrm{dim}}$ and $\mathrm{y}_{\mathrm{dim}}$ respectively, as shown in Fig. A.1.

Every neuron is fully connected to the n variables of the input layer. For the sake of clarity, only the connections to 2 output neurons are shown in Fig. A.1. The weights of the connections between a neuron and the input layer correspond to the "signature" of the neuron. Every unit clusters input data samples with the same distribution as the neuron's signature.

Additionally, the neurons on the grid are interconnected to each other through a neighborhood relationship. Therefore changes in the signature of one output neuron

also affect the neurons in its neighborhood[1].

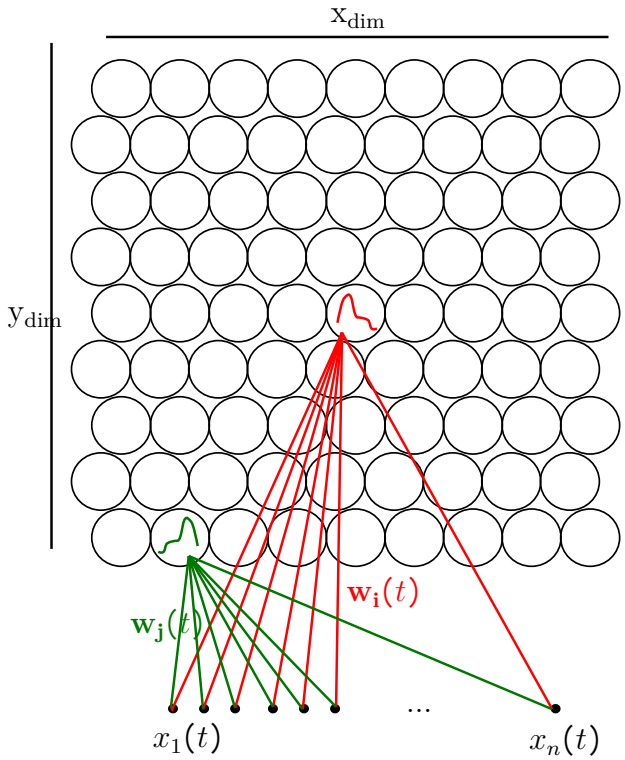


Figure A.1: SOM - Fully connected output and input layers.

For the sake of simplicity and without loss of generality, we start explaining the training of an unsupervised SOM (i.e., without considering the target variable), and afterward, we include the target information to train what is called a Super-Organizing Map. Generally speaking, the training process of an unsupervised SOM is an iterative process as shown in Algorithm 6.

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be the training set with m samples and n number of features or dimensions. Let $t \in \mathbb{Z}_+$ be a training iteration index. The set of neurons in the grid is represented by the set $\mathcal{C}$. The initialization approach of a SOM taking place in line 3 in Algorithm 6 mainly affects the training speed [100]. Once a data point is received ($\mathbf{x}(t)$), a so-called Best Matching Unit (BMU) ($c(\mathbf{x}) \in \mathcal{C}$) is found. A BMU corresponds to the neuron with the closest signature[2] to the input data ($\mathbf{x}(t)$). The closeness notion here depends on a predefined distance metric. Although in line 6 in Algorithm 6, the Euclidean norm has been used, multiple distances can be considered, e.g., Manhattan, Tanimoto, or Mahalanobis distances [100].

[1] This unique feature is used to control the overfitting level of the SOM model [100].
[2] As mentioned before, a neuron's signature corresponds to the weights vector in Fig. A.1.

The learning rate of the SOM ($\delta(t)$) is a function that decreases from a value δ_0 with an increasing number of iterations. The update of $\delta(t)$ takes place in line 7 in Algorithm 6. Afterward, the so-called Neighborhood Function ($\zeta(t)$) is also adapted as in line 8 in Algorithm 6. Let $d_{c,i}$ be the distance (w.r.t the predefined distance metric) between the ith neuron in the grid and the BMU $c(\mathbf{x})$, the so-called Neighborhood Distance Weight function ($h_{c,i}(t)$) is adjusted as in line 9 in Algorithm 6[3].

Finally, after each iteration, all the weights in the grid are adapted considering the current data sample $\mathbf{x}(t)$. The type of update in line 10 in Algorithm 6 is called an online update [100].

Algorithm 6: Training Process of an Unsupervised SOM

1: INPUT: $\mathbf{X}, t_{max}, \delta_0, \delta_{\text{end}}, \zeta_0$
2: INITIALIZE: learning step $t = 1$
3: INITIALIZE: randomly initialize the units signatures $(\forall i \in \mathcal{C})\mathbf{w_i}(t) \in \mathbb{R}^n$
4: **while** $t < t_{max}$ **do**
5: Get random input data point: $\mathbf{x}(t) \in \mathbb{R}^n \subset \mathbf{X}$
6: Find BMU: $c(\mathbf{x}) \in \underset{i \in \mathcal{C}}{\operatorname{argmin}} \parallel \mathbf{x}(t) - \mathbf{w_i}(t) \parallel_2$
7: Update Learning Rate: $\delta(t) = \delta_0 \left(\frac{\delta_{\text{end}}}{\delta_0}\right)^{\frac{t}{t_{max}}}$
8: Update Neighborhood Function: $\zeta(t) = \zeta_0 \left(1 - \frac{t}{t_{max}}\right)$
9: Update Neighborhood Distance Weight: $h_{c,i}(t) = e^{-\frac{d_{c,i}^2}{2\delta(t)^2}}$ $(\forall i \in \mathcal{C})$
10: Update Unit's Signatures: $\mathbf{w_i}(t+1) = \mathbf{w_i}(t) + \delta(t)h_{c,i}(t)[\mathbf{x}(t) - \mathbf{w_i}(t)]$ $(\forall i \in \mathcal{C})$
11: **end while**
12: OUTPUT: Trained Unsupervised SOM

After reaching the maximum number of iterations t_{max}, the unsupervised SOM is fully trained. A graphical interpretation of the main steps in Algorithm 6 is shown in Fig. A.2.

Since the number of nodes on the grid ($|\mathcal{C}| = \mathrm{x_{dim}y_{dim}}$) can be larger than the number of samples in the training set, several nodes that are not linked to any data point of that respective dataset could exist. A measure of this phenomenon can be obtained through the so-called "quality" plot [77] of a SOM, which is typically available in statistical packages. Naturally, it is desired to have grids of high quality. We face a situation where low-quality SOM affects the SF coordination approach in Section 4.4.

In order to include information about the target variable, i.e., to generate a supervised clustering model, it is needed to gather information for both dependent ($\mathbf{y} \in \mathbb{R}^m$)

[3] Multiple strategies for updating $\delta(t)$, $\zeta(t)$, and $h_{c,i}(t)$ are available in [100].

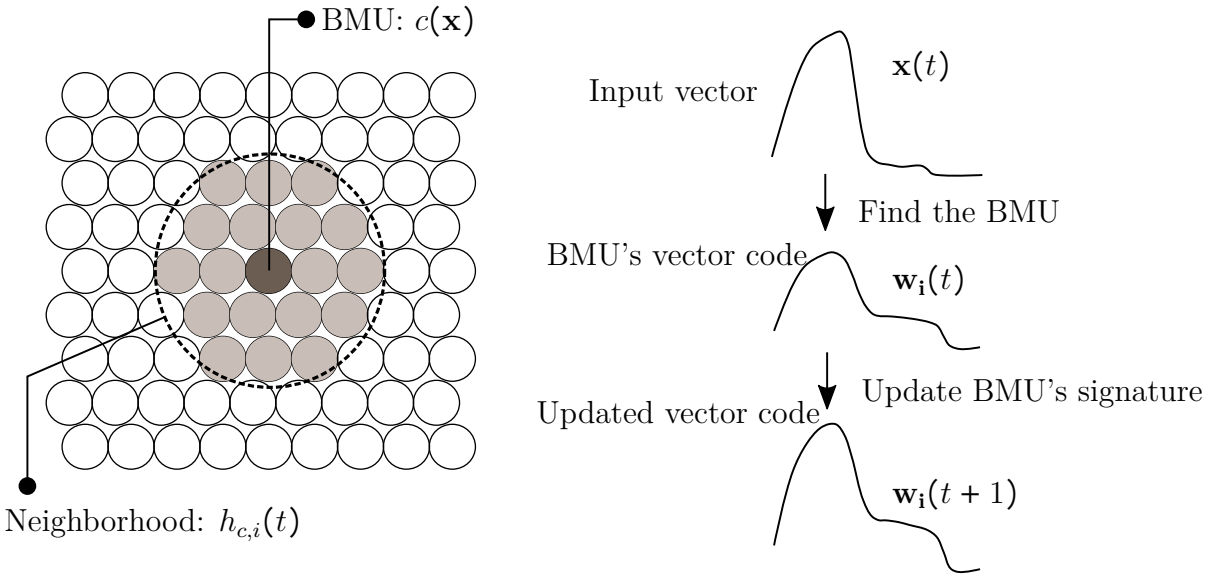


Figure A.2: SOM - training process

and independent variables ($\mathbf{X} \in \mathbb{R}^{m \times n}$). To train a supervised SOM, a second grid (characterizing $\mathbf{y}$) is chained to the trained SOM in Algorithm 6. The training of the second stage is shown in Algorithm 7. Even though at first glance, it could look pretty similar to the one presented in Algorithm 6, there are slight differences in terms of the dimensionality of the signatures, as well as the determination of the BMU. The main considerations are posed as footnotes below the algorithm.

Algorithm 7: Training Process of a Supervised SOM

1: INPUT: Trained Unsupervised SOM from Algorithm 6,$\mathbf{X}, \mathbf{y}, t_{max}, \delta_0, \delta_{\text{end}}, \zeta_0$
2: INITIALIZE: learning step $t = 1$
3: INITIALIZE: Initialize signatures Supervised SOM[a] $(\forall i \in \mathcal{C}) w_i(t) \in \mathbb{R}$
4: **while** $t < t_{max}$ **do**
5: Get random input data point: $\mathbf{x}(t) \in \mathbb{R}^n \subset \mathbf{X}, y(t) \in \mathbb{R} \subset \mathbf{y}$
6: Find BMU[b]: $c(\mathbf{x}) \in \underset{i \in \mathcal{C}}{\operatorname{argmin}} \parallel \mathbf{x}(t) - \mathbf{w_i}(t) \parallel_2$
7: Update Learning Rate: $\delta(t) = \delta_0 \left(\frac{\delta_{\text{end}}}{\delta_0}\right)^{\frac{t}{t_{max}}}$
8: Update Neighborhood Function: $\zeta(t) = \zeta_0 \left(1 - \frac{t}{t_{max}}\right)$
9: Update Neighborhood Distance Weight: $h_{c,i}(t) = e^{-\frac{d_{c,i}^2}{2\delta(t)^2}}$ $(\forall i \in \mathcal{C})$
10: Update Unit's Signatures[c]: $w_i(t+1) = w_i(t) + \delta(t) h_{c,i}(t)[y(t) - w_i(t)]$ $(\forall i \in \mathcal{C})$
11: **end while**
12: OUTPUT: Trained Supervised SOM

[a] In this stage, the signature for every unit in the supervised SOM is not a vector, as we have a single output scenario

[b] Please note that the BMU is obtained considering only the independent variables, i.e., $\mathbf{x}(t)$, which means the BMU is selected considering the already trained unsupervised SOM

[c] Note that the weight that is updated here corresponds to the second grid, therefore it is a single value per unit rather than a codevector

By combining the unsupervised and the supervised SOM, the former is used to select the BMU for each data point while the latter links the selected BMU to a specific regression estimate.

All in all, the SOM training process can be understood as the coexistence of "competition" and "cooperation" stages. During the competition, neurons in the grid compete with each other to become the BMU, whereas, during the cooperation, a topological neighborhood is created and updated such that the BMU will be located at the center of the neighborhood [101].

The aforementioned SOM algorithms can be applied to large data sets. The computational complexity scales linearly with the number of data samples (m) [102]; it does not require huge amount of memory — basically just the signature vectors and the current training vector. Nevertheless, the complexity scales quadratically with the number of map units, i.e., x_{dim} and y_{dim} [102].

Just as we did for chaining two grids (the unsupervised and the supervised maps), it would be possible to stack more grids and carry out an incremental training process, ending up with a so-called Super-Organizing Map. The number of layers depends on

the functional separation of the data and the specifics of the problem to solve. As we see in Chapters 3, 4, and 5, we split the automatic coordination among SFs into three functional domains: controllable, non-controllable, and utility domains. As we see in Chapters 3, 4, and 5, with a Super-Organizing Map we obtain insights about how to move in the low dimensional space to improve a utility function. However, that movement is constrained to environmental conditions (since they are beyond the operator's control). Therefore, to estimate a step size (in the best possible direction), an additional clustering stage can be carried out, as explained below.

Combined SOM-Ward Clustering Analysis

An additional clustering layer could be stacked on top of a Super-Organizing Map. Clustering the SOM rather than clustering the data directly has two benefits [102]:

1. Reduction of the computational cost. In [102], the reduction of the computational load is estimated in $\sqrt{m}/15$, with m the number of samples in the training set.

2. Noise reduction[4]. The signatures in the SOM grid[5] are "local averages" of the data and, therefore, less sensitive to random variations than the original data.

A well-studied representative of these methods is the SOM-Ward clustering grouping. Several studies have shown the effectiveness of the SOM-Ward combination compared with state-of-the-art techniques [101], [102], [103] .

From the taxonomy perspective, the SOM-Ward clustering grouping is considered a hierarchical agglomerative (bottom-up) method. The hierarchical term expresses that a kind of tree is generated, in which every leaf corresponds to a cluster. The term agglomerative (or bottom-up) means that the tree starts being populated from the bottom and reaches the root corresponding to the whole dataset. Usually, that dynamic is represented using a dendogram, like the one presented in Fig. A.3.

Let r and s represent two clusters, n_r and n_s the number of samples associated with each cluster, whereas $\mathbf{w_r}$ and $\mathbf{w_s}$ correspond to the center of gravity of the clusters. The process of generating the clustering on top of the SOM grid, using the so-called Ward distance, is shown in Algorithm 8.

[4]Many clustering techniques require the clusters to be compact and well separated. However, in real applications, this is rarely the case. Instead, the gaps between clusters are blurred due to the noise, the clusters overlap, and the outliers.

[5]Also known as "prototypes" in [102]

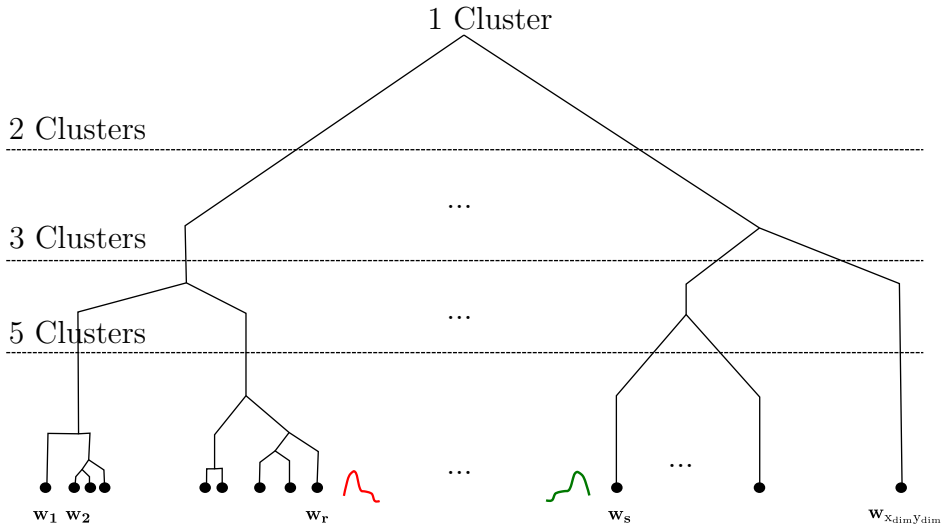


Figure A.3: Dendogram

Algorithm 8: Process to obtain a Dendogram out of a SOM

1: INPUT: Trained Unsupervised SOM from Algorithm 6
2: INITIALIZE: $t = 1$
3: INITIALIZE: Each unit in SOM is a cluster
4: **while** $t > 1$ **do**
5: Compute distance among clusters: $d_{r,s}(t) := \frac{n_r(t)n_s(t)}{n_r(t)+n_s(t)} \parallel \mathbf{w_r}(t) - \mathbf{w_s}(t) \parallel_2$
6: Merge the two clusters that are closest to each other[a].
7: Update the cardinality: $n_r(t+1) := n_r(t) + n_s(t)$
8: Update center of gravity: $\mathbf{w_r}(t+1) := \frac{n_r(t)\mathbf{w_r}(t)+n_s(t)\mathbf{w_s}(t)}{n_r(t)+n_s(t)}$
9: t = get number of clusters left
10: **end while**
11: OUTPUT: Dendogram for clustering the low-dimensional representation

[a] For instance, let cluster r absorb cluster s

Usually, some modifications could be implemented regarding the line 5 in Algorithm 8 to consider some specifics about SOM. On the one hand side, to cope with empty neurons, an adjustment to the distance is shown in Eq. (A.1).

$$d'_{r,s}(t) := \begin{cases} 0 & n_r(t) = 0 \text{ or } n_s(t) = 0 \\ \frac{n_r(t)n_s(t)}{n_r(t)+n_s(t)} \parallel \mathbf{w_r}(t) - \mathbf{w_s}(t) \parallel_2 & \text{otherwise} \end{cases} \tag{A.1}$$

Additionally, to take into account the topological properties of the SOM grid, two neurons that are not adjacent in the grid are never considered to be merged, see Eq. (A.2).

$$d''_{r,s}(t) \coloneqq \begin{cases} d'_{r,s} & \text{if } r \text{ and } s \text{ are adjacent in the grid} \\ \infty & \text{otherwise} \end{cases} \tag{A.2}$$

Of course, multiple versions of Eq. (A.1) and Eq. (A.2) are available, and they can be tailored to accomplish a specific task, e.g., rather than assigning ∞ distance whenever two clusters are not adjacent, it would be possible to assign a distance that increases exponentially concerning the relative position between clusters s and r. In that manner, the connectivity of the resulting clusters is obtained more smoothly.

However, the output dendogram from Algorithm 8 does not provide a unique clustering, as shown in Fig. A.3. Fortunately, there are many techniques to find the appropriate number of clusters ($|\boldsymbol{C_k}|$) and prune the dendogram, for instance, the Elbow, Siluethe, Gap Statistic methods among others [104], which try to follow the classical definition of clustering: a partitioning that minimizes distances within and maximizes distances between clusters. Throughout this thesis, we use the well-known Elbow method, which looks at the total WSS, which measures the compactness of the clustering.

B Uniform Manifold Approximation and Projection - UMAP

UMAP is a relatively modern technique that excels at reducing the dimensionality from an extremely high number of dimensions[1] and it is commonly used for visualization, like in the case of SOM in the Appendix A. It was introduced in [105], and it is made of two steps: first, the construction of a high-dimensional graph followed by force-directed placement of the graph into a low-dimensional space.

Once again, let $\mathbf{X} \coloneqq \{x_i\} \in \mathbb{R}^{m \times n}$ be the training set with m samples and n number of features and $d(x_i, x_j)$ a distance metric between two instances of the training set x_i and x_j, i.e., $d : \mathbf{X} \times \mathbf{X} \to \mathbb{R}_+$.

The general steps of UMAP are shown in Algorithm 9, which offers a slight adaptation from [106]. First, the generation of a weighted high-dimensional (k-neighbor) graph $\mathcal{G}$ takes place from lines 2 to 7 in Algorithm 9. The main hyperparameter for this part of the algorithm is k (the number of nearest neighbors in the high dimensional space every point is allowed to have), which models a tradeoff between conserving the local structure (i.e., specifics of the data set, since a relatively low value for k results in many small and independent clusters) versus the global structure of the data (i.e., the main characteristics of the data set). If k is high, the global structure is conserved, whereas if k is low, the local structure of the input data is conserved. Finding the suitable k is usually carried out through grid search procedures, as we see in Section 3.3.1.3.

[1] To get an idea, the "hello world" example for UMAP consists of reducing the dimensionality of the well-studied MNIST dataset which has an input dimension of 784 into 2D or 3D keeping the topological characteristics of the input data.

Algorithm 9: UMAP Algorithm

1: INPUT: $\mathbf{X}$, k, a, b, ε
2: Find a set of k neighbor points: $\boldsymbol{\theta_i}$ $\forall i = 1, ..., m$
3: Find nearest neighbor and the distance: $(\forall i)(\forall \mathbf{x_j} \in \boldsymbol{\theta_i})\xi_i := \min(d(\mathbf{x_i}, \mathbf{x_j}))$
4: Solve for χ_i $\forall i = 1, ..., m$ in: $\sum_{\mathbf{x_j} \in \boldsymbol{\theta_i}} e^{\frac{-\max(0, d(\mathbf{x_i}, \mathbf{x_j}) - \xi_i)}{\chi_i}} = \log_2 k$
5: Find matrix $\mathbf{A}^a$ with elements: $w(\mathbf{x_i}, \mathbf{x_j}) = e^{\frac{-\max(0, d(\mathbf{x_i}, \mathbf{x_j}) - \xi_i)}{\chi_i}}$
6: Calculate $\mathbf{B}^b$: $\mathbf{B} = \mathbf{A} + \mathbf{A^T} - \mathbf{A} \circ \mathbf{A^T}$ [c]
7: Create Graph $\mathcal{G} := (\mathcal{V}, \mathcal{E}, \mathbf{B})^d$
8: Mapping to low-dimensional space: $\mathcal{H} = \text{GraphLayout}(\mathcal{G}, \text{a}, \text{b}, \varepsilon)$
9: OUTPUT: Graph $\mathcal{H}$

[a] One could interpret A_{ij} as the probability that a directed edge from $\mathbf{x_i}$ to $\mathbf{x_j}$ exists
[b] B_{ij} is the probability that at least one of the two directed edges (from $\mathbf{x_i}$ to $\mathbf{x_j}$ and from $\mathbf{x_j}$ to $\mathbf{x_i}$) exists
[c] $\circ$ is the Hadamard (or pointwise) product
[d] The vertices $\mathcal{V}$ of the graph $\mathcal{G}$ correspond to the set $\mathbf{X}$ whereas the weight of the edges $\mathcal{E}$ is given by the adjacency matrix $\mathbf{B}$

Second, to project $\mathcal{G}$ to a low-dimensional space keeping the topology properties, a force-directed graph layout algorithm is used, obtaining a new induced graph $\mathcal{H}$. Without losing generality, and to keep Algorithm 9 simple, we aggregate the whole second stage of UMAP into a single function called $\text{GraphLayout}(\mathcal{G}, \text{a}, \text{b}, \varepsilon)$ (line 8 in Algorithm 9). If implementation details are needed, [105] could be a starting point. Let $\mathbf{x_i'}$, $i = 1, ..., m$ be the coordinates of an input data point in the low-dimensional space, i.e., $\mathbf{x_i'} \in \mathbb{R}^{n'}$, with $n' << n$, and let a, b, and ε be algorithm hyperparameters, then the attractive force between two vertices i and j ($\mathbf{F^a_{i,j}}$) is defined as in Eq. (B.1), whereas the repulsive force ($\mathbf{F^r_{i,j}}$) is defined as in Eq. (B.2).

$$\mathbf{F^a_{i,j}} := \frac{-2ab||\mathbf{x_i'} - \mathbf{x_j'}||_2^{2(b-1)}}{1 + ||\mathbf{x_i'} - \mathbf{x_j'}||_2^2} w(\mathbf{x_i}, \mathbf{x_j})(\mathbf{x_i'} - \mathbf{x_j'}) \tag{B.1}$$

$$\mathbf{F^r_{i,j}} := \frac{2b}{(\varepsilon + ||\mathbf{x_i'} - \mathbf{x_j'}||_2^2)(1 + a||\mathbf{x_i'} - \mathbf{x_j'}||_2^{2b})} (1 - w(\mathbf{x_i}, \mathbf{x_j}))(\mathbf{x_i'} - \mathbf{x_j'}) \tag{B.2}$$

$\text{GraphLayout}(\mathcal{G}, \text{a}, \text{b}, \varepsilon)$ proceeds by iteratively applying the predefined attractive and repulsive forces at each vertex during a long enough number of iterations or until convergence with respect to some metric is achieved[2]. This metric is usually the edge-wise cross-entropy between graphs $\mathcal{G}$ and $\mathcal{H}$, which is meant to be minimized [105],

[2] This mechanism resembles the competition-collaboration harmony in SOM (see Appendix A).

[107]. GraphLayout$(\mathcal{G}, \mathrm{a}, \mathrm{b}, \varepsilon)$ aims at solving a non-convex optimization problem [105], and the convergence to a local minimum is guaranteed by slowly decreasing the attractive and repulsive forces in a similar way to the reduction of the learning rate in SOM training (see Appendix A) or the temperature cooling down scheme in simulated annealing (see Appendix E).

Hierarchical Density-Based Spatial Clustering of Applications with Noise -HDBSCAN

As we did with the SOM-Ward clustering grouping in Appendix A (and for the same reasons), we study the grouping of UMAP and HDBSCAN, which has been proven recently to exhibit outstanding performance while clustering noisy data in a low-dimensional space [107, 108]. HDBSCAN was initially proposed in [109], and it is an extension of the clustering algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) which was proposed in [110, 111]. HDBSCAN only needs an input parameter: $\mathrm{m_{pts}}$, which can be interpreted as the number of neighboring points in a low-dimensional space. Let $\mathbf{X}' \coloneqq \left\{x_i'\right\} \in \mathbb{R}^{m \times n'}$ be the data set in the low-dimensional space induced by $\mathcal{H}$ (output from Algorithm 9). To formally introduce HDBSCAN, the following definitions are taken from [109]:

Definition B.0.1 (Core Distance)**.** The core distance of an instance $\mathbf{x_i'} \in \mathbf{X}'$ w.r.t. $\mathrm{m_{pts}}$, $\mathrm{d_{core}}(\mathbf{x_i'})$, is the distance from $\mathbf{x_i'}$ to its $\mathrm{m_{pts}}$-nearest neighbor (incl. $\mathbf{x_i'}$).

Definition B.0.2 (Mutual Reachability Distance)**.** The mutual reachability distance between two objects $\mathbf{x_i'} \in \mathbf{X}'$ and $\mathbf{x_j'} \in \mathbf{X}'$ w.r.t. $\mathrm{m_{pts}}$ is defined as $\mathrm{d_{mreach}}(\mathbf{x_i'}, \mathbf{x_j'}) = \max\left\{\mathrm{d_{core}}(\mathbf{x_i'}), \mathrm{d_{core}}(\mathbf{x_j'}), d(\mathbf{x_i'}, \mathbf{x_j'})\right\}$.

Definition B.0.3 (Mutual Reachability Graph)**.** It is a complete graph, $\mathcal{G}_{\mathrm{m_{pts}}}$, in which the objects of $\mathbf{X}'$ are the vertices, and the weight of each edge is the mutual reachability distance w.r.t. $\mathrm{m_{pts}}$ (see Definition B.0.2) between the respective pair of objects.

Considering the previous definitions, the HDBSCAN method is sketched as in Algorithm 10, which was adapted from [112].

Algorithm 10: HDBSCAN Algorithm

1: INPUT: $\mathbf{X}'$, $\mathrm{m_{pts}}$
2: Compute the core distances: $\mathrm{d_{core}}(\mathbf{x_i'})\ \forall\ x_i' \in \mathbf{X}'$ (see Definition B.0.1)
3: Compute the Mutual Reachability Graph: $\mathcal{G}_{\mathrm{m_{pts}}}$ (see Definition B.0.3)
4: Compute a Minimum Spanning Tree (MST)[a] out of $\mathcal{G}_{\mathrm{m_{pts}}}$
5: Extend the MST by adding for each vertex a "self-edge" with the core distance $\mathrm{d_{core}}(\mathbf{x_i'})\ \forall\ x_i' \in \mathbf{X}'$ of the corresponding object as weight.
6: Generate a dendogram out of the extended MST [112]
7: Condense the dendogram using the "Excess of Mass" Method [109]
8: Obtain the final clusters using the stability concept: $S(\boldsymbol{C_k})$
9: OUTPUT: Set of clusters: $\{\boldsymbol{C_k}\}$

[a] A Spanning Tree is a subset of the graph which covers all vertices with the minimum possible edges. A MST is a spanning tree with minimum cost in a weighted graph. There are multiple methods to obtain the MST like Prim's, Kruskal's, or Dual-Tree Boruvka methods, however, a detailed description is out of the scope of this document.

As we did in Appendix A, a dendogram is generated from lines 1 to 6 in the Algorithm 10. Having the dendogram, we could use a technique to prune it as we did in Appendix A, i.e., we could draw a horizontal line through the dendogram and select the clusters that it cuts through. As a matter of fact, that is how DBSCAN works.

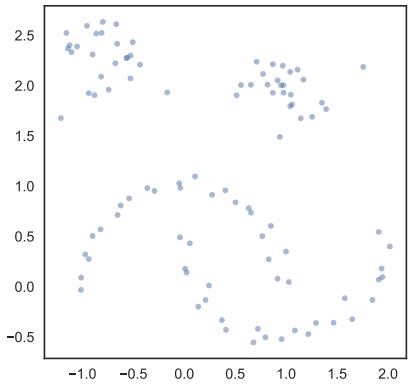
Fortunately, HDBSCAN offers a way to condense the dendogram and automatically generate a smaller tree with a little more data attached to each branch. This is done through the "Excess of Mass" method proposed in [109], which requires a hyperparameter called: *minimum cluster size*. Once again, a grid search is needed to select the best values for this parameter, as shown in Section 3.3.1.3.

To extract the final set of clusters ($\{\boldsymbol{C_k}\}$), a stability metric is defined based on the so-called Λ-values ($\Lambda \coloneqq 1/(\text{distance on dendogram})$) as in Eq. (B.3) [109]:

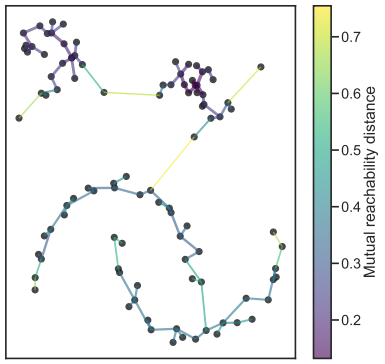
$$S(\boldsymbol{C_k}) \coloneqq \sum_{\mathbf{x_i'} \in \boldsymbol{C_k}} \Big(\Lambda_{\max}(\mathbf{x_i'}, \boldsymbol{C_k}) - \Lambda_{\min}(\boldsymbol{C_k})\Big) \tag{B.3}$$

where $\Lambda_{\min}(\boldsymbol{C_k})$ is the minimum density level at which $\boldsymbol{C_k}$ exists[3], and $\Lambda_{\max}(\mathbf{x_i'}, \boldsymbol{C_k})$ is the density level beyond which object $\mathbf{x_i'}$ no longer belongs to cluster $\boldsymbol{C_k}$. Intuitively, Eq. (B.3) computes the "area" of every cluster, and we want to keep the ones that exhibit the greatest areas of "ink" in the plot (compared with the parent's areas). Using dummy data, and for the sake of clarity, the steps above are reproduced using a statistical package to illustrate the whole process as in Fig. B.1

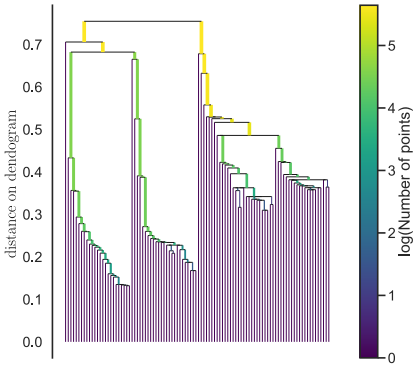
[3] That means, the Λ-value when the cluster split off and became its own cluster



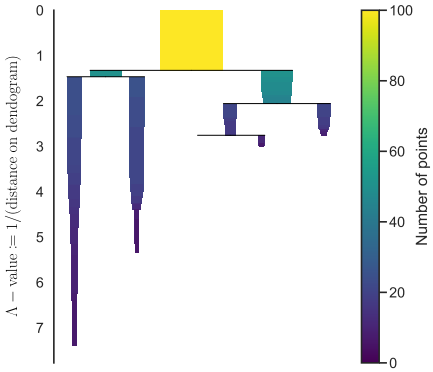
(a) $\mathbf{X}'$ in 2D, i.e., $n' = 2$



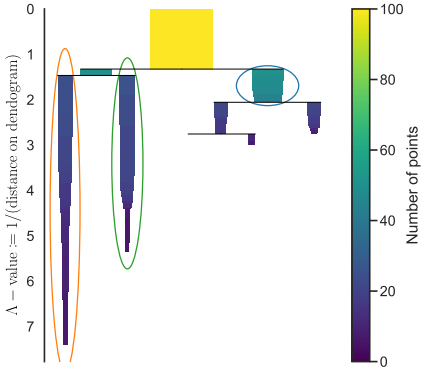
(b) MST as in line 5 in Algorithm 10



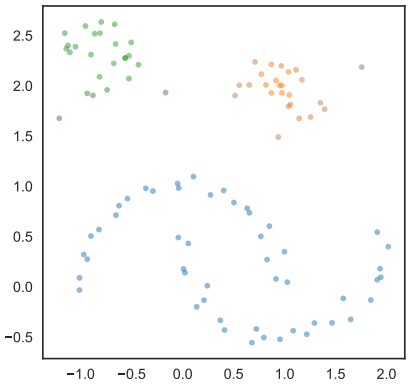
(c) Dendogram as in line 6 in Algorithm 10



(d) Condensed dendogram: line 7 Alg. 10



(e) High stability clusters: line 8 Alg. 10



(f) Three clusters and noisy samples

Figure B.1: HDBSCAN in a nutshell

C Reinforcement Learning

RL is a category within machine learning designed to solve a Markov Decision Process (MDP). It is quite different from traditional supervised or unsupervised learning techniques since it is designed to let an agent learn online through interaction with an environment. RL aims to find an optimal *policy* that maps environmental conditions to the best possible actions to maximize a numerical reward (or minimize a penalty).

Let $\mathcal{S}$ be the state space of the system, let $\mathcal{A}$ be the set of actions an agent can take, let $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ be a transition function, let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a reward function, and let $\Gamma \in [0, 1]$ be a reward discount factor. At each time step ($t = 0, 1, ...$), the agent receives information about the environment state through $s_t \in \mathcal{S}$; based on that state, the agent takes an action $a_t \in \mathcal{A}$, forcing to move the system to a different state $s_{t+1} \in \mathcal{S}$ while obtaining a reward $r_{t+1} \in \mathbb{R}$.

During the training process, the agent updates the so-called agent's *policy*: π[1] to maximize the total reward in the long term, as defined in Eq. (C.1) [113], [114]:

$$R_t \coloneqq \sum_{k=0}^{\infty} \Gamma^k r_{t+k+1} \tag{C.1}$$

If Γ is close to zero, the method is called *myopic* [114] (which means it aims at immediate high rewards), whereas if Γ is close to 1, the algorithm maximizes future rewards. Additionally, let us define the *value* of a state $s \in \mathcal{S}$ as the expected future return associated to follow the policy π starting from s, as in Eq. (C.2).

$$V^{\pi}(s) \coloneqq \mathbb{E}_{\pi}\{R_t | s_t = s\} = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \Gamma^k r_{t+k+1} | s_t = s\right\} \tag{C.2}$$

Likewise, the *state-action value* (a.k.a *Q-values*) is defined as the expected reward if we choose action a in state s and follow the policy π afterward, as in Eq. (C.3)

[1] Short for $\pi(s, a)$, which represents the probability of taking the action a at time t ($a_t = a$) when the system is in state s ($s_t = s$)

$$Q^{\pi}(s,a) \coloneqq \mathbb{E}_{\pi}\left\{R_t | s_t = s, a_t = a\right\} = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \Gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \tag{C.3}$$

Let $P^a_{ss'}$ be the probability of transitioning from state s to s' given the agent has chosen action a, as in Eq. (C.4).

$$P^a_{ss'} \coloneqq \Pr\left\{s_{t+1} = s' | s_t = s, a_t = a\right\} \tag{C.4}$$

Let $R^a_{ss'}$ be the expected reward after transitioning to state s' from state s while applying action a, as in Eq. (C.5).

$$R^a_{ss'} \coloneqq \mathbb{E}\left\{r_{t+1} | s_t = s, a_t = a\right\} \tag{C.5}$$

In order to intuitively understand the uncertainty factors in a state transition, please consider Fig. C.1 which shows a stochastic transition from state s to state s' after taking action a in the initial state. Please notice that the red state (the so-called Q-state) is a theoretical abstraction used to show the uncertainty both in the action to take (a) as well as in the landing state (s').

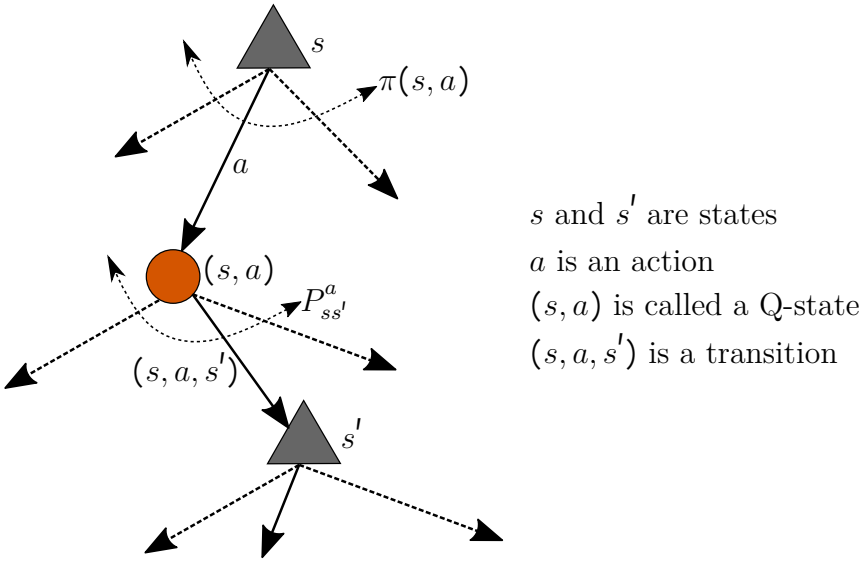


Figure C.1: Transition dynamics in a MDP.

Based on Fig. C.1, it is possible to reformulate Eq. (C.2) and Eq. (C.3) in a different manner:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(s,a) \sum_{s' \in \mathcal{S}} P^a_{ss'}\left[R^a_{ss'} + \Gamma V^{\pi}(s')\right] \tag{C.6}$$

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} P^{a}_{ss'} \left[R^{a}_{ss'} + \Gamma V^{\pi}(s') \right] \tag{C.7}$$

Eq. (C.6) and Eq. (C.7) are called the *Bellman Equations* for the *value function* and *state-action function*, respectively. The first sum on the right-hand side of Eq. (C.6) goes over all the possible actions in state s (see the upper part of Fig. C.1). The second sum in Eq. (C.6) is taken over all the possible next states (see the bottom part of Fig. C.1). The term in squared brackets in both equations corresponds to the immediate reward $R^{a}_{ss'}$ plus the future discounted rewards from the next state onwards ($\Gamma V^{\pi}(s')$).

Let $V^{*}(s) \coloneqq \max_{\pi} V^{\pi}(s) \forall s \in \mathcal{S}$ be the optimal value function and $Q^{*}(s,a) \coloneqq \max_{\pi} Q^{\pi}(s,a) \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$ be the optimal state-action function. It is possible to show that [115]:

$$V^{*}(s) = \max_{a \in \mathcal{A}} Q^{*}(s,a) \tag{C.8}$$

If a^{*} is an action at which the maximum of Eq. (C.8) is attained, then the optimal policy is given by $\pi^{*}(s) = a^{*}$ [115].

Therefore, if an agent learns the Q-values, it can optimally decide what to do next based on its current state. However, according to Eq. (C.7), finding $Q^{*}(s,a)$ implies the knowledge about $P^{a}_{ss'}$ and $R^{a}_{ss'}$ (these quantities are known as the system model) and solving a system of equations afterward.

Unfortunately, the system model is hardly available in real scenarios. Therefore, some model-free methods to estimate $Q^{*}(s,a)$ are available within the domain of Time Difference Learning (TDL). Q-Learning is a well-established model-free approach to estimate the optimal state-action function [115].

The recursive equation used by Q-Learning is as in Eq. (C.9).

$$Q_{t+1}(s_t,a_t) = (1-\delta_t) Q_t(s_t,a_t) + \delta_t \left\{ r_t + \Gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) \right\} \tag{C.9}$$

where $\delta_t \in (0,1]$ is an adaptive learning rate. In Algorithm 11 it is shown the complete Q-Learning method [115].

Algorithm 11: Q-Learning Algorithm

INPUT: $t_{\max}$
INITIALIZE: $t = 1$, randomly assign $Q_t(s_t, a_t)$
while $t < t_{\max}$ **do**
    Observe the current state: s_t
    Choose an action a_t based on an ϵ-greedy policy[a]
    Observe the subsequent state: s_{t+1}
    Observe the received payoff: r_t
    Update $Q_{t+1}(s_t, a_t)$ using Eq. (C.9)
    $s_t = s_{t+1}$
end while
OUTPUT: $Q^*(s, a) = Q_{t+1}(s_t, a_t)$

[a] At each iteration t, the agent chooses a random action a_t with a probability ϵ or an optimal action given by $a_t = \pi(s_t)$ with a probability $1 - \epsilon$. ϵ monotonically decreases from 1 at the beginning of the iterations to 0, forcing the agent to take only optimal actions once the agent is trained with the optimal policy.

D Multilabel Learning Problem

A multilabel learning task receives as input a training set $\{(\mathbf{x_1}, \mathbf{y_1}), ..., (\mathbf{x_m}, \mathbf{y_m})\}$ which has m samples and aims to find a model $h : \mathcal{X} \rightarrow \mathcal{Y}$, with $\mathcal{Y} = \{0, 1\}^B$ that predicts the binary value for each output, indicating a high (1) or low (0) value for a cell parameter. Under this definition, the number of parameters to optimize is given by B (the number of cells), and the size of the optimization space is given by 2^B. A description of the major multilabel classification families (namely, multioutput classifiers and chain classifiers) is given below.

Multioutput Classifiers

The strategy behind multioutput classifiers in Fig. 4.5 consists of fitting one traditional classifier per output (cell) [91, 89, 90]. For a multioutput classification problem with B outputs, B binary classifiers are trained using $\mathcal{X}$, as indicated in Eq. (D.1).

$$h_j : \mathcal{X} \rightarrow \{0, 1\} \tag{D.1}$$

Standard classifiers can be used in this family, e.g., decision tree, k-neighbors, multilayer perceptrons, random forest, and ridge classifiers, among many others. Rather than visiting the details of any of those well-established classifiers, we spend some time discussing the evaluation metrics for a multilabel task because those metrics allow us to pick the best model among the vast number of candidates. If details about individual models are needed, [76] is a good starting point.

The so-called *label-based* evaluation measures decompose the evaluation process into separate evaluations for each model (h_j), which are subsequently averaged over all the outputs [91]. Any metric for binary evaluation could be used here, such as accuracy, the *area under the Receiver Operating Characteristic* (ROC) curve, precision, or recall. Therefore, we review some basic definitions:

- let TP be the true positive samples (that were correctly predicted as 1),
- let FP be the false positive samples (that were wrongly predicted as 1),

- let FN be the false negative samples (that were wrongly predicted as 0),
- and let TN be the true negative samples (that were correctly predicted as 0).

The well-known classification performance metrics are defined as follows:

- The classification precision is defined as [76]: $\text{precision} \coloneqq \frac{\text{TP}}{\text{TP+FP}}$
- The recall (a.k.a sensitivity) [76]: $\text{recall} \coloneqq \frac{\text{TP}}{\text{TP+FN}}$
- The F1 metric, which measures the compromise between precision and recall through the harmonic mean between both [76], $F1 \coloneqq \frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}}$
- The ROC curve plots the recall in the vertical axis against the False Positive Rate (FPR) [76], defined as: $\text{FPR} \coloneqq \frac{\text{FP}}{\text{FP+TN}}$. FPR is equal to 1-True Negative Rate (TNR), which is the ratio of negative instances that are correctly classified as negative. The TNR is also called specificity. Hence, the ROC curve plots recall versus 1-specificity. A perfect classifier will have an Area Under ROC Curve equal to 1, whereas a purely random classifier will have an Area Under ROC Curve equal to 0.5.

However, the above list applies only to single-output learning tasks. As we saw, all of them depend on the tuple $(\text{TP}, \text{TN}, \text{FP}, \text{FN})$. Let $H(\text{TP}, \text{TN}, \text{FP}, \text{FN})$ generalize any of the previous metrics. To apply them in a multilabel scenario, an aggregation is needed. There are two main types of aggregation (averaging), the macro-averaging (see Eq. (D.2)) and micro-averaging (see Eq. (D.3)) [91]:

$$H_{\text{macro}} = \frac{1}{B}\sum_{j=1}^{B} H(\text{TP}_j, \text{TN}_j, \text{FP}_j, \text{FN}_j) \tag{D.2}$$

$$H_{\text{micro}} = H\left(\sum_{j=1}^{B}\text{TP}_j, \sum_{j=1}^{B}\text{TN}_j, \sum_{j=1}^{B}\text{FP}_j, \sum_{j=1}^{B}\text{FN}_j\right) \tag{D.3}$$

The micro-averaging differs from macro-averaging for precision, recall, and area under the ROC curve metrics [91].

To select the adequate metric, a rule of thumb is used: one should prefer the F1 metric whenever the positive class (1) is rare or when one cares more about the false positives (FP) than the false negatives (FN). Otherwise, use the ROC curve [76]. Throughout this document (especially in Chapters 4 and 5), we use the micro-averaged version of the Area Under ROC Curve since we have no a priori knowledge about the

rareness of the positive class, and we care about both false positives and false negatives to the same extent.

Regardless of the chosen model, the multioutput classifiers exhibit a drawback: they cannot take advantage of possible dependencies among the different outputs. If the value of one output may (statistically) depend on the value of others[1], then predicting all outputs in an interweaved manner should indeed be better than predicting them separately. Herein the chain classifiers cover that gap, as shown below.

Chain Classifiers

The other family of multilabel classifiers in Fig. 4.5 is the chain classifiers. In [92], the output-independence assumption is dropped, and it is proposed a novel chaining method that can model output correlations while maintaining acceptable computational complexity. Unlike multioutput classifiers, in which isolated classifiers are trained per output, in the chain classifiers family, there is a linkage among individual classifiers.

For a multilabel classification problem with B outputs, B binary classifiers are trained using an extension of $\mathcal{X}$ (based on the classifiers of the previous outputs[2]), in the sense of Eq. (D.4) (please compare with the approach in Eq. (D.1)):

$$h_j : \mathcal{X} \times \{0,1\}^{j-1} \rightarrow \{0,1\} \tag{D.4}$$

The training procedure is shown in Fig. D.1, which is adapted from [116], where it has been assumed that $B = 5$. From Fig. D.1, it is observed that the first classifier is trained with the original attributes and predicts a binary output, which will be used to extend the training set of the second classifier, which in turn predicts an additional binary output, and so on.

Once the models are trained, and for prediction purposes, the first output is predicted exclusively by the first classifier using the actual features: $\mathbf{x} \in \mathcal{X}$, while the second output is predicted using $\mathbf{x} \in \mathcal{X}$ and the prediction of the previous classifier, and so on. The final prediction is made up using Eq. (D.5). This is further illustrated in Fig. D.2.

$$\mathbf{y} = (h_1(\mathbf{x}), h_2(\mathbf{x}, h_1(\mathbf{x})), h_3(\mathbf{x}, h_1(\mathbf{x}), h_2(\mathbf{x}, h_1(\mathbf{x}))), ...) \tag{D.5}$$

[1] That is what we expect in real mobile networks.

[2] A sort of training order is imposed on the chain beforehand. The standard rule is to train the chain multiple times using random sorting and averaging the final predictions over the different train runs.

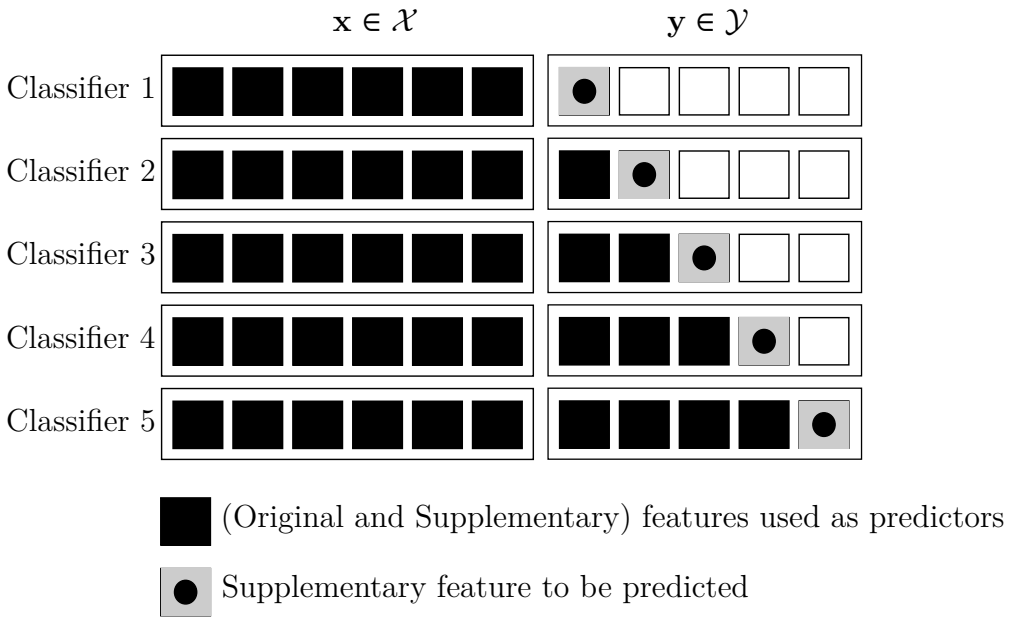


Figure D.1: Training of a classifier chain.

As mentioned before, the ordering of the outputs in the chain plays an important role. The first model in the chain (providing a parameter configuration to a cell) has no information about the other outputs (meaning that it can not model the impact of other cells). In contrast, the last one has features indicating the presence of all of the other outputs (thus considering the effect of other cells in the modeling). To cope with this unfairness, many randomly ordered chains are trained, and their predictions are averaged altogether [89, 90].

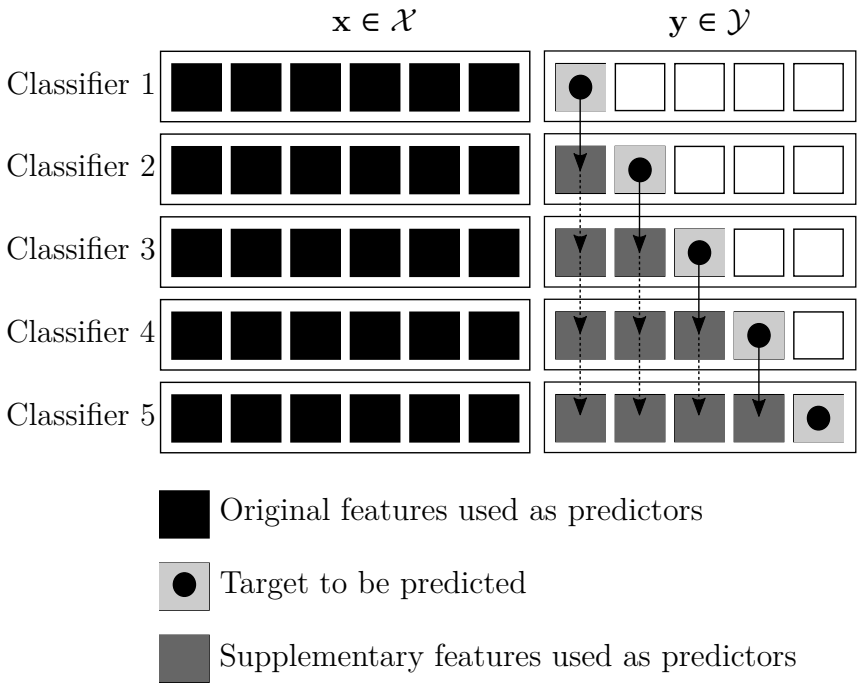


Figure D.2: Prediction using a classifier chain

E Simulated Annealing

Simulated Annealing (SA) is a heuristic introduced in the 80s to solve large combinatorial problems. The way how SA works in this kind of scenario is depicted in Fig. E.1, which is adapted from [117], where (for the sake of illustration) the minimization of a univariate function ($f(x)$) is sought. Let $N(\boldsymbol{x'})$ be the neighborhood of a feasible solution $\boldsymbol{x'}$, $\tau_{\max}$ the maximum (initial) temperature of the annealing process, which is reduced through a cooling down procedure until reaching the minimum temperature $\tau_{\min}$.

At the beginning of the procedure, due to the high temperature, strong perturbations are applied (to the candidate solutions $\boldsymbol{x'}$) allowing the method to jump over peaks (and escape from local minima), searching for the bottom of the box (i.e., the global minimum).

As the temperature monotonically decreases, it is less likely to jump over high peaks. Hopefully, once $\tau_{\min}$ is reached and if the temperature has not been decreased that quickly, the optimization space has been explored long enough, and $\boldsymbol{x'} \rightarrow \boldsymbol{x}^*$.

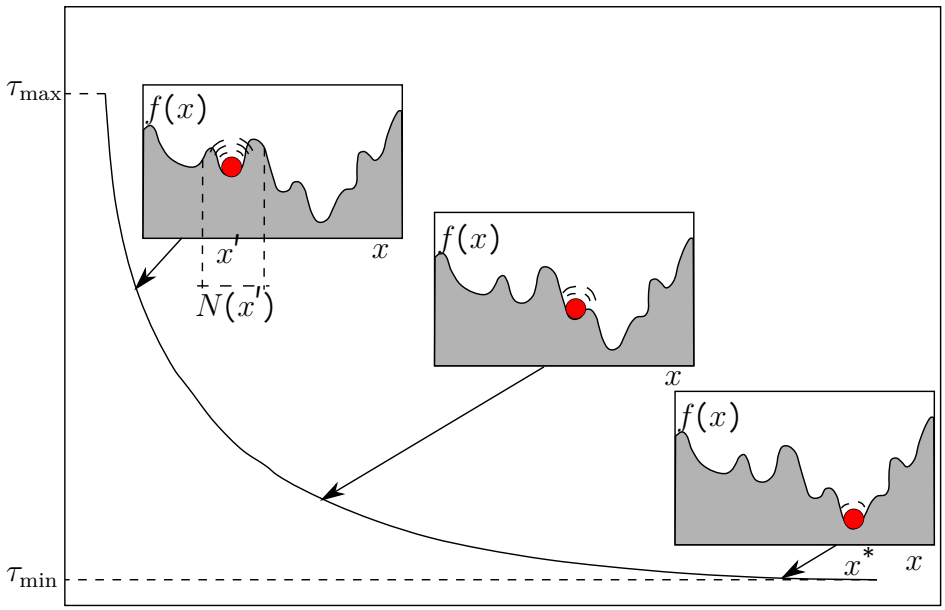


Figure E.1: Simulated Annealing for a minimization problem.

In a nutshell, the heuristic (which has been implemented as in Algorithm 12) has three main stages:

1. Solution perturbation: generating a random solution within a neighborhood (followed by a utility calculation under the new solution). It takes place in lines 6-7 in Algorithm 12. Please notice that, for every temperature τ, a maximum number of iterations ($i_{\max}$) (solution perturbations) is carried out. The number of temperatures visited, as well as $i_{\max}$, determine the amount of exploration over the optimization space.

2. Acceptance test: it uses the *Metropolis Criterion* to accept not only utility improvements but also deteriorations to a limited extent to escape from local minima. It is implemented from lines 9 to 19 in Algorithm 12.

3. Cooling down schedule: the notion of cooling is interpreted as decreasing the probability of accepting solutions with worse utility as the search space is explored. It is taking place in line 24 in Algorithm 12.

Algorithm 12: Simulated Annealing

INPUT: $\boldsymbol{x}_{\mathbf{init}}, i_{max}, \tau_{min}, \tau_{max}$
INITIALIZE: $\tau = \tau_{max}, \boldsymbol{x} = \boldsymbol{x}_{\mathbf{init}}$
Compute $f(\boldsymbol{x})$
while $\tau > \tau_{min}$ **do**
  for all $i = 1$ to i_{max} **do**
    Compute $f(\boldsymbol{x}')$ with $\boldsymbol{x}' \in N(\boldsymbol{x}) \setminus \boldsymbol{x}$
    $\Delta(f) = f(\boldsymbol{x}') - f(\boldsymbol{x})$
    flag=1
    if $\Delta(f) > 0$ **then**
      Accept $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
    else
      Calculate probability: $\Pr(\Delta(f)) = e^{\frac{\Delta(f)}{\tau}}$
      if $\Pr(\Delta(f)) > \mathrm{rnd}(0,1)$ **then**
        Accept $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
      else
        Reject $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
        flag=0
      end if
    end if
    if flag=1 **then**
      Update $\boldsymbol{x}^* = \boldsymbol{x}$
    end if
  end for
  $\tau = \frac{\tau}{log(i+1)}$
end while
OUTPUT:$\boldsymbol{x}^*$

Bibliography

[1] NGMN Alliance et al. NGMN recommendation on SON and O&M requirements. *Next Generation Mobile Networks, White paper, December*, 2008.

[2] 3GPP. Telecommunication management; Study on the Self-Organizing Networks (SON) for 5G networks. Technical Report (TR) 28.861, 3rd Generation Partnership Project (3GPP), 12 2019. Version 16.0.0.

[3] Tanmoy Bag, Sharva Garg, Diego Preciado, Zubair Shaik, Jens Mueckenheim, and Andreas Mitschele-Thiel. Self-organizing network functions for handover optimization in LTE cellular networks. In *Mobile Communication-Technologies and Applications; 24. ITG-Symposium*, pages 1–7. VDE, 2019.

[4] Diego Preciado and Andreas Mitschele-Thiel. Machine learning-based SON function conflict resolution. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2019.

[5] Diego Preciado and Andreas Mitschele-Thiel. A scalable SON coordination framework for 5G. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

[6] Diego Preciado, Faiaz Nazmetdinov, and Andreas Mitschele-Thiel. Zero-touch coordination framework for self-organizing functions in 5G. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2020.

[7] Gerald Budigiri, Diego Preciado, Andreas Mitschele-Thiel, and Stephen Mwanje. Optimal rules mining in SON for distributed intelligence in future cognitive cellular networks. In *2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, pages 1–6. IEEE, 2019.

[8] Tanmoy Bag, Sharva Garg, Diego Preciado, and Andreas Mitschele-Thiel. Machine learning-based recommender systems to achieve self-coordination between

SON functions. *IEEE Transactions on Network and Service Management*, 17(4):2131–2144, 2020.

[9] Diego Preciado, Martin Kasparick, Renato L. G. Cavalcante, and Slawomir Stanczak. SON function coordination in campus networks using machine learning. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2022.

[10] Diego Preciado and Andreas Mitschele-Thiel. A data driven coordination between load balancing and interference cancellation. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE, 2022.

[11] Christopher Cox. *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications*. John Wiley & Sons, 2012.

[12] Harri Holma, Antti Toskala, and Takehiro Nakamura. *5G technology: 3GPP new radio*. John Wiley & Sons, 2020.

[13] Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. prentice hall PTR New Jersey, 1996.

[14] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Further advancements for E-UTRA physical layer aspects. Technical Report (TR) 36.814, 3rd Generation Partnership Project (3GPP), 3 2017. Version 9.2.0.

[15] Ling Liu, Yiqing Zhou, Athanasios V Vasilakos, Lin Tian, and Jinglin Shi. Time-domain ICIC and optimized designs for 5G and beyond: a survey. *Science China Information Sciences*, 62(2):1–28, 2019.

[16] Abdelbaset S. Hamza, Shady S. Khalifa, Haitham S. Hamza, and Khaled Elsayed. A survey on inter-cell interference coordination techniques in OFDMA-based cellular networks. *IEEE Communications Surveys Tutorials*, 15(4):1642–1670, 2013.

[17] Mohamad Yassin, Mohamed A AboulHassan, Samer Lahoud, Marc Ibrahim, Dany Mezher, Bernard Cousin, and Essam A Sourour. Survey of ICIC techniques in LTE networks under various mobile environment parameters. *Wireless Networks*, 23(2):403–418, 2017.

[18] Tanmoy Bag, Sharva Garg, Diego Fernando Preciado Rojas, and Andreas Mitschele-Thiel. Machine learning-based recommender systems to achieve self-coordination between son functions. *IEEE Transactions on Network and Service Management*, 17(4):2131–2144, 2020.

[19] Harri Holma and Antti Toskala. *LTE for UMTS: Evolution to LTE-advanced*. John Wiley & Sons, 2011.

[20] Francesco Capozzi, Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Downlink packet scheduling in LTE cellular networks: Key design issues and a survey. *IEEE communications surveys & tutorials*, 15(2):678–700, 2012.

[21] Fayssal Bendaoud, Marwen Abdennebi, and Fedoua Didi. Survey on scheduling and radio resources allocation in LTE. *arXiv preprint arXiv:1404.2759*, 2014.

[22] Leonard Kleinrock. Computer applications. *Queueing systems*, 1976.

[23] Na Chen and Scott Jordan. Throughput in processor-sharing queues. *IEEE Transactions on Automatic Control*, 52(2):299–305, 2007.

[24] S Ben Fred, Thomas Bonald, Alexandre Proutiere, Gwénaël Régnié, and James W Roberts. Statistical bandwidth sharing: a study of congestion at flow level. *ACM SIGCOMM Computer Communication Review*, 31(4):111–122, 2001.

[25] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode. Technical Specification (TS) 36.304, 3rd Generation Partnership Project (3GPP), 12 2021. Version 16.6.0.

[26] 3GPP. General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. Technical Specification (TS) 23.401, 3rd Generation Partnership Project (3GPP), 12 2021. Version 17.3.0.

[27] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2. Technical Specification (TS) 36.300, 3rd Generation Partnership Project (3GPP), 7 2021. Version 16.6.0.

[28] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management. Technical Specification (TS) 36.133, 3rd Generation Partnership Project (3GPP), 1 2022. Version 17.4.0.

[29] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification. Technical Specification (TS) 36.331, 3rd Generation Partnership Project (3GPP), 12 2021. Version 16.7.0.

[30] 3GPP. Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements. Technical Specification (TS) 32.500, 3rd Generation Partnership Project (3GPP), 7 2020. Version 16.0.0.

[31] 3GPP. Telecommunication management; Self-configuration of network elements; Concepts and requirements. Technical Specification (TS) 32.501, 3rd Generation Partnership Project (3GPP), 7 2020. Version 16.0.0.

[32] 3GPP. Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases an d solutions. Technical Report (TR) 36.902, 3rd Generation Partnership Project (3GPP), 4 2011. Version 9.3.1.

[33] 3GPP. Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Requirements. Technical Specification (TS) 28.627, 3rd Generation Partnership Project (3GPP), 7 2020. Version 16.0.0.

[34] 3GPP. Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS). Technical Specification (TS) 28.628, 3rd Generation Partnership Project (3GPP), 9 2020. Version 16.1.0.

[35] 3GPP. Telecommunication management; Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS). Technical Specification (TS) 28.658, 3rd Generation Partnership Project (3GPP), 12 2020. Version 16.4.0.

[36] 3GPP. Telecommunication management; Energy Saving Management (ESM); Concepts and requirements. Technical Specification (TS) 32.551, 3rd Generation Partnership Project (3GPP), 7 2020. Version 16.0.0.

[37] 3GPP. Study on RAN-centric data collection and utilization for LTE and NR. Technical Report (TR) 37.816, 3rd Generation Partnership Project (3GPP), 7 2019. Version 16.0.0.

[38] 3GPP. Telecommunication management; Automatic Neighbour Relation (ANR) management; Concepts and requirements. Technical Specification (TS) 32.511, 3rd Generation Partnership Project (3GPP), 1 2020. Version 16.0.0.

[39] 3GPP. NR; NR and NG-RAN Overall description; Stage-2. Technical Specification (TS) 38.300, 3rd Generation Partnership Project (3GPP), 9 2021. Version 16.7.0.

[40] 3GPP. NR; Radio Resource Control (RRC); Protocol specification. Technical Specification (TS) 38.331, 3rd Generation Partnership Project (3GPP), 9 2021. Version 16.6.0.

[41] 3GPP. NG-RAN; Xn Application Protocol (XnAP). Technical Specification (TS) 38.423, 3rd Generation Partnership Project (3GPP), 10 2021. Version 16.7.0.

[42] 3GPP. Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements. Technical Specification (TS) 32.541, 3rd Generation Partnership Project (3GPP), 7 2020. Version 16.0.0.

[43] 3GPP. Universal Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collecti on for Minimization of Drive Tests (MDT); Overall description; Stage 2. Technical Specification (TS) 37.320, 3rd Generation Partnership Project (3GPP), 9 2021. Version 16.6.0.

[44] 3GPP. Telecommunication management; Subscriber and equipment trace; Trace concepts and requirements. Technical Specification (TS) 32.421, 3rd Generation Partnership Project (3GPP), 6 2021. Version 17.2.0.

[45] Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. *LTE self-organising networks (SON): network management automation for operational efficiency.* John Wiley & Sons, 2012.

[46] 3GPP. Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS). Technical Specification (TS) 32.522, 3rd Generation Partnership Project (3GPP), 9 2013. Version 11.7.0.

[47] Stephen S Mwanje and Christian Mannweiler. *Towards cognitive autonomous networks: Network management automation for 5G and beyond*. John Wiley & Sons, 2020.

[48] Markus Gruber, Sem Borst, and Edgar Kuehn. Stable interaction of self-optimization processes in wireless networks. In *2011 IEEE International Conference on Communications Workshops (ICC)*, pages 1–6. IEEE, 2011.

[49] Stephen S Mwanje, Henning Sanneck, and Andreas Mitschele-Thiel. Synchronized cooperative learning for coordinating cognitive network management functions. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):244–256, 2018.

[50] Stephen S Mwanje and Andreas Mitschele-Thiel. STS: space-time scheduling for coordinating self-organization network functions in LTE. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 357–362. IEEE, 2015.

[51] Hafiz Yasar Lateef, Ali Imran, Muhammad Ali Imran, Lorenza Giupponi, and Mischa Dohler. LTE-advanced self-organising network conflicts and coordination algorithms. *IEEE Wireless Communications Magazine*, 22(3):108–117, 2015.

[52] H. Klessig, A. Fehske, G. Fettweis, and J. Voigt. Improving coverage and load conditions through joint adaptation of antenna tilts and cell selection rules in mobile networks. In *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pages 21–25, August 2012.

[53] Albrecht J. Fehske, Henrik Klessig, Jens Voigt, and Gerhard P. Fettweis. Concurrent load-aware adjustment of user association and antenna tilts in self-organizing radio networks. *IEEE Transactions on Vehicular Technology*, 62(5):1974–1988, 2013.

[54] Henrik Klessig, Albrecht Fehske, Gerhard Fettweis, and Jens Voigt. Cell load-aware energy saving management in self-organizing networks. In *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, pages 1–6. IEEE, 2013.

[55] Jens Bartelt, Albrecht Fehske, Henrik Klessig, Gerhard Fettweis, and Jens Voigt. Joint bandwidth allocation and small cell switching in heterogeneous networks. In *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, pages 1–5, 2013.

[56] Abdoulaye Tall, Richard Combes, Zwi Altman, and Eitan Altman. Distributed coordination of self-organizing mechanisms in communication networks. *IEEE transactions on control of network systems*, 1(4):328–337, 2014.

[57] UNIVERSELF: The network of the future. http://www.univerself-project.eu/. Accessed: 2022-05-31.

[58] Ingo Karla. Resolving SON interactions via self-learning prediction in cellular wireless networks. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–6. IEEE, 2012.

[59] Jessica Moysen and Lorenza Giupponi. Self-coordination of parameter conflicts in D-SON architectures: a markov decision process framework. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):82, 2015.

[60] Pablo Mu, Raquel Barco, Sergio Fortes, et al. Conflict resolution between load balancing and handover optimization in LTE networks. *IEEE Communications Letters*, 18(10):1795–1798, 2014.

[61] Ovidiu Iacoboaiea, Berna Sayrac, Sana Ben Jemaa, and Pascal Bianchi. SON coordination for parameter conflict resolution: A reinforcement learning framework. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 196–201. IEEE, 2014.

[62] SEMAFOUR: Self-management for unified heterogeneous radio access networks. http://fp7-semafour.eu/. Accessed: 2022-05-31.

[63] Zwi Altman, Mehdi Amirijoo, Fredrik Gunnarsson, Hendrik Hoffmann, István Z Kovács, Daniela Laselva, Bart Sas, Kathleen Spaey, Abdoulaye Tall, Hans van den Berg, et al. On design principles for self-organizing network functions. In *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 454–459. IEEE, 2014.

[64] Hafiz Yasar Lateef, Ali Imran, and Adnan Abu-Dayya. A framework for classification of self-organising network conflicts and coordination algorithms. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2898–2903. IEEE, 2013.

[65] Christoph Frenzel, Bernhard Bauer, Christoph Schmelz, and Henning Sanneck. Objective-driven coordination in self-organizing networks. In *2015 IEEE 26th*

Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pages 1453–1458. IEEE, 2015.

[66] Harrison Mfula and Jukka K Nurminen. Business-aware SON coordinator for LTE-A networks. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 246–252. IEEE, 2016.

[67] Bart Sas, Kathleen Spaey, Irina Balan, Kristina Zetterberg, and Remco Litjens. Self-optimisation of admission control and handover parameters in LTE. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE, 2011.

[68] Jessica Moysen, Mario Garcia-Lozano, Lorenza Giupponi, and Silvia Ruiz. Conflict resolution in mobile networks: a self-coordination framework based on non-dominated solutions and machine learning for data analytics [application notes]. *IEEE Computational Intelligence Magazine*, 13(2):52–64, 2018.

[69] SELFNET: A framework for self-organized network management in virtualized and software defined networks. https://5g-ppp.eu/selfnet/. Accessed: 2022-04-01.

[70] Wei Jiang, Mathias Strufe, and Hans D Schotten. Intelligent network management for 5G systems: The SELFNET approach. In *2017 European conference on networks and communications (EuCNC)*, pages 1–5. IEEE, 2017.

[71] Wei Jiang, Mathias Strufe, and Hans D Schotten. A SON decision-making framework for intelligent management in 5G mobile networks. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1158–1162. IEEE, 2017.

[72] Alberto Huertas Celdrán, Manuel Gil Pérez, Félix J García Clemente, and Gregorio Martínez Pérez. Automatic monitoring management for 5G mobile networks. *Procedia Computer Science*, 110:328–335, 2017.

[73] Lorena Isabel Barona Lopez, Angel Leonardo Valdivieso Caraguay, Marco Antonio Sotelo Monge, and Luis Javier García Villalba. Key technologies in the context of future networks: Operational and management requirements. *Future Internet*, 9(1):1, 2016.

[74] Lorena Isabel Barona López, Jorge Maestre Vidal, and Luis Javier García Villalba. An approach to data analysis in 5G networks. *Entropy*, 19(2):74, 2017.

[75] Ángel Leonardo Valdivieso Caraguay and Luis Javier García Villalba. Monitoring and discovery for self-organized network management in virtualized and software defined networks. *Sensors*, 17(4):731, 2017.

[76] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2019.

[77] Ron Wehrens and Johannes Kruisselbrink. Flexible self-organizing maps in kohonen 3.0. *Journal of Statistical Software*, 87:1–18, 2018.

[78] Jörg Sommer and Joachim Scharf. IKR simulation library. In *Modeling and Tools for Network Simulation*, pages 61–68. Springer, 2010.

[79] ITU. Architectural framework for machine learning in future networks including IMT-2020. Recommendation Y.3172 (06/2019), ITU-T Study Group 13, jun 2019.

[80] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[81] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[82] Hongseok Kim, Gustavo De Veciana, Xiangying Yang, and Muthaiah Venkatachalam. Distributed-optimal user association and cell load balancing in wireless networks. *IEEE/ACM Transactions on Networking*, 20(1):177–190, 2012.

[83] Hua Qu, Gongye Ren, Jihong Zhao, Zhenjie Tan, and Shuyuan Zhao. Joint optimization of content placement and user association in cache-enabled heterogeneous cellular networks based on flow-level models. *Wireless Communications and Mobile Computing*, 2018.

[84] Apoorva M Sampat and Victor M Zavala. Fairness measures for decision-making and conflict resolution. *Optimization and Engineering*, 20(4):1249–1272, 2019.

[85] Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. The price of fairness. *Operations research*, 59(1):17–31, 2011.

[86] Iana Siomina and Di Yuan. Analysis of cell load coupling for LTE network planning and optimization. *IEEE Transactions on Wireless Communications*, 11(6):2287–2297, 2012.

[87] Chin Keong Ho, Di Yuan, and Sumei Sun. Data offloading in load coupled networks: A utility maximization framework. *IEEE Transactions on Wireless Communications*, 13(4):1921–1931, 2014.

[88] Peng Tian, Jian Ma, and Dong-Mo Zhang. Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of generation mechanism. *European Journal of Operational Research*, 118(1):81–94, 1999.

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[90] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[91] O. Maimon and L. Rokach. *Data mining and knowledge discovery handbook*, volume 14. Springer, 2010.

[92] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[93] Nur Oyku Tuncel and Mutlu Koca. Joint mobility load balancing and inter-cell interference coordination for self-organizing OFDMA networks. In *2015 IEEE 81st vehicular technology conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[94] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[95] Usama Sallakh, Stephen S Mwanje, and Andreas Mitschele-Thiel. Multi-parameter Q-learning for downlink inter-cell interference coordination in LTE

SON. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2014.

[96] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

[97] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[98] Renato LG Cavalcante, Yuxiang Shen, and Sławomir Stańczak. Elementary properties of positive concave mappings with applications to network planning and optimization. *IEEE Transactions on Signal Processing*, 64(7):1774–1783, 2015.

[99] Renato LG Cavalcante, Emmanuel Pollakis, and S Stańczak. Power estimation in LTE systems with the general framework of standard interference mappings. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 818–822. IEEE, 2014.

[100] Felix M Riese, Sina Keller, and Stefan Hinz. Supervised and semi-supervised self-organizing maps for regression and classification focusing on hyperspectral data. *Remote Sensing*, 12(1):7, 2020.

[101] Nurettin Yorek, Ilker Ugulu, and Halil Aydin. Using self-organizing neural network map combined with ward's clustering algorithm for visualization of students' cognitive structural models about aliveness concept. *Computational Intelligence and Neuroscience*, 2016, 2016.

[102] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[103] Zhiyuan Yao, Tomas Eklund, and Barbro Back. Using SOM-Ward clustering and predictive analytics for conducting customer segmentation. In *2010 IEEE International Conference on Data Mining Workshops*, pages 639–646. IEEE, 2010.

[104] Alboukadel Kassambara. *Practical guide to cluster analysis in R: Unsupervised machine learning*, volume 1. Sthda, 2017.

[105] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[106] E Myasnikov. Using UMAP for dimensionality reduction of hyperspectral data. In *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pages 1–5. IEEE, 2020.

[107] Clément Pealat, Guillaume Bouleux, and Vincent Cheutet. Improved time-series clustering with UMAP dimension reduction method. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5658–5665. IEEE, 2021.

[108] Muhammad Sidik Asyaky and Rila Mandala. Improving the performance of HDBSCAN on short text clustering by using word embedding and umap. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–6. IEEE, 2021.

[109] Ricardo JGB Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–51, 2015.

[110] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(3):231–240, 2011.

[111] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[112] Joelson Antônio dos Santos, Talat Iqbal Syed, Murilo C Naldi, Ricardo JGB Campello, and Joerg Sander. Hierarchical density-based clustering using MapReduce. *IEEE Transactions on Big Data*, 7(1):102–114, 2019.

[113] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[114] Howard M Schwartz. *Multi-agent machine learning: A reinforcement approach*. John Wiley & Sons, 2014.

[115] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[116] Dominik Heider, Robin Senge, Weiwei Cheng, and Eyke Hüllermeier. Multilabel classification for exploiting cross-resistance information in hiv-1 drug resistance prediction. *Bioinformatics*, 29(16):1946–1952, 2013.

[117] Sergio Ledesma, Gabriel Aviña, and Raul Sanchez. Practical considerations for simulated annealing implementation. *Simulated annealing*, 20:401–420, 2008.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungsbzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch bewertet wird und gemäß § 7 Abs. 10 der Promotionsordnung den Abbruch des Promotionsverfahrens zur Folge hat.

Ilmenau, November 23, 2022 Diego Fernando Preciado Rojas