Cognitive management of self organized radio networks of fifth generation

Tony Daher

NNT : 2018SACLT023

Gestion Cognitive des Réseaux Radio Auto-Organisant de Cinquième Génération

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

Ecole doctorale n°574 Ecole Doctorale de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques Appliquées

Thèse présentée et soutenue à Paris, le 11-12-2018, par

TONY DAHER

Composition du Jury :

M. Xavier LAGRANGE Professeur, IMT Atlantique, France	Président
Mme. Raquel BARCO Professeur, Université de Malaga, Espagne	Rapporteur
M. James LEDOUX Professeur, INSA Rennes, France	Rapporteur
M. Tijani CHAHED Professeur, Telecom SudParis, France	Examinateur
M. Laurent DECREUSEFOND Professeur, Telecom ParisTech, France	Directeur de thèse
Mme. Sana BEN JEMAA Ingénieur de Recherche Senior, Orange Labs, France	Encadrant de thèse

Thèse de doctorat

Abstract

The operation of mobile networks has always been a challenging task for network operators. In the recent years, the pressure on operators to improve the network management efficiency has been constantly growing for many reasons: the user traffic that is increasing very fast, higher end users expectations, emerging services with very specific and different requirements, inducing an increasing complexity in the operation and management of such networks. Many solutions have already been proposed to simplify and optimize the management of complex networks, notably the Autonomic Network Management (ANM) paradigm. A first step towards ANM was achieved with the 3rd Generation Partnership Project (3GPP) introduction of the Self-Organizing Networks (SON) concept, in its release 8. Many SON functions are already being deployed in today's networks, especially in the Radio Access Network (RAN), and will be the bedrock of 5G networks. Such networks can be seen as SON enabled networks, and they have already proved to be useful in reducing the complexity of network management, improving the Quality of Service (QoS) of end users and reducing operational costs. However, SON enabled networks are still far from realizing a network that is autonomous and self-managed as a whole, because the deployed SON functions need themselves to be properly managed. In fact, the behavior of the SON functions depends on the parameters of the algorithm running in the control loop, as well as on the network environment where it is deployed (dense or not, traffic profile, technology, required services, etc). Furthermore, SON objectives and actions might be conflicting with each other, leading to incompatible parameter tuning in the network, causing performance degradation. Each SON function hence still needs to be itself manually configured, depending on the network environment and the desired high level objectives of the operator.

It is clear then, that in order to achieve a step further towards truly self-organized networks, there is a need for an integrated SON management entity that efficiently manages the many deployed SON functions according to the operator's objectives. Furthermore, the SON functions are considered to be provided to network operators by SON vendors, and that the latter share very few information about the algorithms running inside the functions, which makes them black boxes for the operator. In this thesis, we propose an approach for an integrated SON management system through a Cognitive Policy Based SON Management (C-PBSM) approach, based on Reinforcement Learning (RL). The C-PBSM translates autonomously high level operator objectives, formulated as target Key Performance Indicators (KPIs), into configurations of the SON functions. Furthermore, through its cognitive capabilities, the C-PBSM is able to build its knowledge by interacting with the real network and learning from past decisions and experiences, without relying on pre-defined models that can misleading be far from reality. It is also capable of reasoning and adapting with the environment changes and evolutions. We first investigate an online learning approach using stochastic Multi-Armed Bandit (MAB) under stationary traffic hypothesis. The scalability of the proposed approaches is analyzed and tackled through two different approaches: linear stochastic MAB and distributed Q-learning. Practical aspects of deploying RL agents in real networks are also investigated under Software Defined Network (SDN) architecture. In addition,

we overcome the traffic stationarity hypothesis using a contextual MAB algorithm. We show that with this approach, it is possible to learn simultaneously and collaboratively over different network sections (e.g. a group of sector in a geographical area) having different contexts, and transfer the acquired knowledge to other different networks sections. Proof of concept and validation are performed on a 3GPP compliant system level LTE-A simulator. Several use cases are investigated, featuring SON functions such as Mobility Load Balancing (MLB), Cell Range Expansion (CRE), enhanced Inter Cell Interference Coordination (eICIC) and Sleep Mode modules.

Contents

List of Figures

List of Tables

Nomenclature

1G	First Generation
2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
5G-PPP	Fifth Generation Public Private Partnership
AAS	Active Antenna Systems
ABS	Almost Blank Sub-frame
AE	Action Elimination
AI	Automatic Inventory
ANB	Automatic Neighbor Relation
ANM	Autonomic Network Management
API	Application Programmable Interface
ASD	Automatic Software Download
BF	Bandit Forest
CapEx	Capital Expenditures
CDMA	Code Division Multiple Access
CIO	Cell Individual Offset
COC	Cell Outage detection and Compensation
C-PBSM	Cognitive Policy Based SON Management
CQI	Channel Quality Information
CRE	Cell Range Expansion
C-SON	Centralized Self-Organizing Networks
D2D	Device-to-Device
DFT	Discrete Fourier Transform
DQL	Distributed Q-Learning
D-SON	Distributed Self-Organizing Networks
DSS	Decision Support System
eICIC	enhanced Inter Cell Interference Coordination
eMBB	enhanced Mobile Broad Band
eNB	eNodeB
EPC	Evolved Packet Core
ES	Energy Saving
ETSI	European Telecommunications Standards Institute
E-UTRA	Evolved Universal Terrestrial Radio Access
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FFT	Fast Fourier Transform

FTP	File Transfer Protocol
GSM	Groupe Spéciale Mobile
HetNet	Heterogeneous Network
HO	HandOver
HSS	Home Subscriber Server
i.i.d.	Independent and Identically Distributed
ICIC	Inter Cell Interference Coordination
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
IoT	Internet of Things
KPI	Key Performance Indicators
LTE-A	Long Term Evolution – Advanced
MAB	Multi-Armed Bandit
MD	Monitoring and Diagnosis
MDP	Markov Decision Process
MDT	Minimization Drive Tests
ML	Machine Learning
MLB	Mobility Load Balancing
MME	Mobility Management Entity
M-MIMO	Massive-Multiple Input Multiple Output
MRO	Mobility Robustness Optimization
MTC	Machine Type Communication
OAM	Operation Administration and Maintenance
OFDMA	Orthogonal Frequency-Division Multiple Access
OFU	Optimism in Face of Uncertainty
OMC	Operations and Maintenance Center
OpEx	Operational Expenditures
PAC	Probably Approximately Correct
PAPR	Peak-to-Average Power Ratio
PBSM	Policy Based SON Management
PCI	Physical Cell Identity
PCRF	Policy and Charging Rules Function
P-GW	Packet Data Gateway
PRB	Physical Resource Block
QoE	Quality of Experience
QoS	Quality of Service
RACH	Random Access Channel
RAN	Radio Access Network
RAT	Radio Access Technologies
RE	Resource Element
RL	Reinforcement Learning
RLF	Radio Link Failure
SC-FDMA	Single Carrier Frequency Division Multiple Access
SCV	SON Configuration Value
SDN	Software Defined Network
SE	Successive Elimination
SFM	SON Function Model
S-GW	Serving Gateway

SINR	Signal to Interference plus Noise Ratio
SMS	Short Message Service
SON	Self-Organizing Networks
SONCO	Self-Organizing Networks Coordination
SVD	Singular Value Decomposition
TD	Temporal Difference
TDMA	Time Division Multiple Access
TTT	Time to Trigger
UCB	Upper Confidence Bound
UE	User Equipment
uRLLC	ultra Reliable and Low Latency Communications
UTRA	Universal Terrestrial Radio Access
V2X	Vehicular-to-Anything
VE	Variable Elimination
VNI	Visual Networking Index

Chapter 1

Introduction

Mobile usage, services and data traffic are growing at a rapid rate. This increase is a consequence of a number of factors. First, the number of wireless devices accessing the network is remarkably increasing. Second, few years ago, the majority of devices using wireless networks were mobile phones and computers, today a variety of other devices are using the network such as connected cars, smart houses and buildings, health monitoring devices, industrial and agricultural sensors and monitors, etc. Furthermore, these devices have become smarter with increased processing capacities, leading hence to the emergence of new services with specific and various requirements such as ultra Reliable and Low Latency Communications (uRLLC), enhanced Mobile Broad Band (eMBB) services, Massive Machine type Communications (MMC). The Cisco Visual Networking Index (VNI) forecasts the traffic evolution in mobile networks [4]. All the indicators confirm the massive mobile usage and data traffic increase, for instance the mobile data traffic has grown eighteen fold over the past five years. This increase is expected to continue for the next years: mobile data traffic will increase sevenfold by 2021, reaching 49 exabytes a month.

To deal with such requirements and demands, networks have to be much more flexible, with optimized capacity and more resources. Such characteristics can be brought to the network through virtualization, slicing, dense and heterogeneous deployment, and enhanced Radio Access Technologies (RAT). Furthermore, new technologies will most likely have to be deployed in parallel with legacy networks such as 2G,3G and 4G. This induces an increase in the network parameters to be monitored and optimized by operators, as well as an increase in the interdependencies between the configurations. The complexity of the operation and management of future networks is hence expected to grow. On the other hand, the operation and management of mobile networks has always been an important and strategic task for network operators, especially with the need to continuously reduce the Operational Expenditures (OpEx), as well as the Capital Expenditures (CapEx), while maintaining a good and competitive Quality of Service (QoS) and Quality of Experience (QoE) for the users.

With the increasing network complexity, optimizing the management and operation of networks is becoming a challenging task for the operators. Automating the management of the networks is thus a must. Autonomic Network Management (ANM) approaches have already been proposed and studied in a number research works and projects since IBM launched the autonomic computing initiative in 2001 [5]. The ANM paradigm aims to develop self-managing systems, that are able to manage themselves and adapt their behaviors to provide the best service, given high-level objectives from administrators and with minimal human intervention.

A first step towards ANM was achieved with the Third Generation Partnership (3GPP) first introduction and standardization of the Self-Organizing Networks (SON) functions in its release 8 [6]. A SON function receives measurements feedback from the network, and changes certain

network parameters according to this feedback and to its objective. It is hence a control loop, performing autonomously a well defined operational task in the network as shown in figure 1.1 [7].

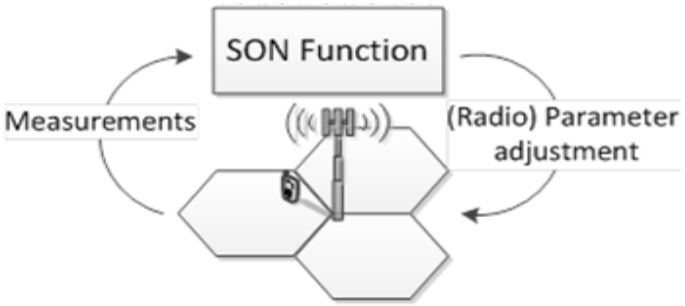


Figure 1.1: SON Control Loop

SON functions are already being deployed in today's radio networks. They have proved to be able to bring profit in the operation of networks by improving the CapEx and OpEx of operators and improving QoS of users as well. They will hence be the bedrock of 5G networks. However, a network with SON functions, that can be seen as a SON enabled network, is still far from realizing a network that is autonomous and self-managed as a whole. In fact, an operator would ideally want its high-level objectives and requirements, expressed as global KPIs, to be automatically translated into proper network parameters. On the other hand, the deployed SON functions still need themselves to be properly managed. First because their objectives might be conflicting with each other, leading to incompatible parameter tuning in the network, which may lead to performance degradation. Second, the behavior of the SON functions depend on the parameters of the algorithm running in the control loop, as well as on the network environment where it is deployed. Therefore, each SON function still needs to be itself manually configured, depending on the network environment and the operator's high level objectives. Autonomic SON management consists hence in autonomously translating the high-level objectives of the operator into efficient configurations of the SON functions. Once configured properly, the SON functions will themselves autonomously manage and configure network parameters, so that the specified operator objectives are satisfied as shown in figure 1.2.

The SON management process has to take into consideration a number of constraints and alleviate several challenges. In fact, the network environments, specifically radio networks, are very complex and many elements can affect the behavior and the impact of the SON functions in the networks such as the network topology, geographical location, technology, required services, transmission frequency bands, etc. Besides, it has to be able to manage the dynamics of the network, in particular the different traffic profiles and variations. It should also manage the interaction between the SON functions, especially if they have conflicting objectives. Finally, the SON functions are usually provided by SON vendors to the network operators, and the former share very few information about the algorithms running inside the functions, which makes them black boxes for the operator, adding an additional constraint to the SON management process.

Consequently, in order to bring self-organization to a higher level and realize an important step towards a radio network that is self organized as a whole, we need to efficiently manage the deployed SON functions, by introducing cognition into the SON management process. A cognitive process can be seen as an analogy with the human reasoning: it observes and gathers information

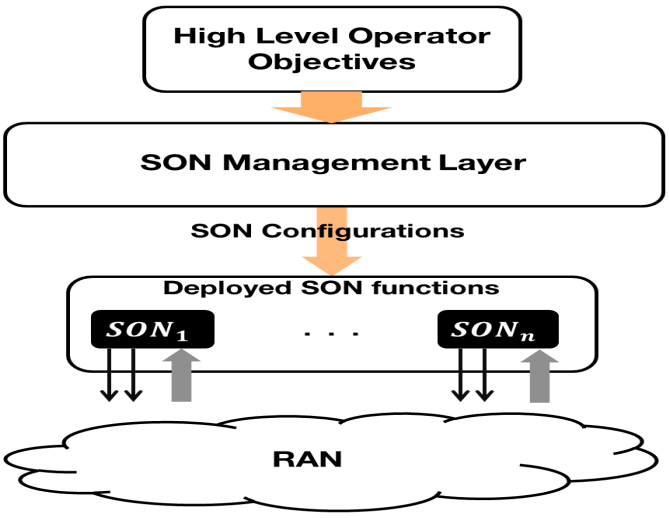


Figure 1.2: SON Management Scheme

from the environment, it then takes a decision based on its knowledge and takes an action based on this decision. It then observes the reaction of the environment, gathers new information, and so on. It builds its knowledge based one the observations and decisions from past experiences (figure 1.3). A cognitive system is a system which is autonomic and intelligent. The cognitive loop can be implemented through Reinforcement Learning (RL). A RL algorithm is a process that learns how to achieve a goal by interacting with the environment.

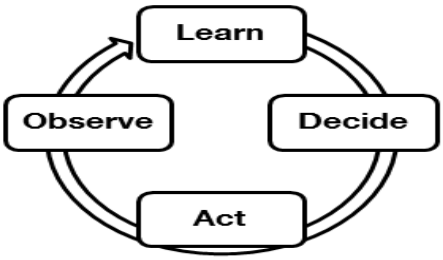


Figure 1.3: Cognitive Loop

Even though RL has been theoretically studied for many years, and has been implemented and tested in a number of cases and scenarios (especially in robotics), implementing these techniques in complex systems such as the SON management in mobile radio networks is a challenging task. Many considerations have to be taken into account. In the following we cite the main challenges that we attempt to tackle in this work:

- Convergence rate: All learning algorithm need a learning phase where they build their knowledge in order to find optimal strategies. However, during learning phases, the network performance is sub-optimal, which is problematic when learning online in the real network.
- Scalability: A well known issue in RL is scalability. In fact, when the set of possible actions or the set of possible system states increases, the time needed to converge towards

the optimal policy increases too.

- Various Network Contexts: The learning process should be able to efficiently learn and adapt over the many various contexts that can be present in a large RAN.
- Dynamic Environment: The radio environment is known to be a dynamic environment, mainly because of different traffic profiles and variations, and because of long term evolutions and changes in the network such as traffic demand increase, the emergence of new services, changes in the network topology, etc.
- Objectives Changes: Operator high level objectives may change according to the operator's strategy, but also depending on the network context and the social activity of the users (mass events, vacation period, holidays, etc). The learning algorithm should hence be able to adapt quickly with eventual objective changes.

In this thesis, we propose a Cognitive Policy Based SON Management (C-PBSM) framework based on RL. The C-PBSM should be able to efficiently manage the deployed SON functions in a network, by learning from the real network the optimal SON configurations that satisfy to the best the objectives specified by the operator. We propose different approaches, in an attempt to tackle the previously mentioned challenges. Each proposed approach is analyzed thoroughly. Use cases are simulated on a system level Long Term Evolution - Advanced (LTE-A) simulator.

The thesis is structured as follows.

- Chapter 2 is an extensive state of the art about SON management, more precisely the PBSM. We develop about the need to go beyond the mere automation of SON management, and to enhance the process with cognitive capabilities through a cognitive control loop that can be implemented through RL algorithms. We discuss the advantages and improvements rendered possible by introducing cognition to the process. We then provide an overview of the theoretical background for RL and present a functional architecture for the C-PBSM based on RL. In this chapter we argue in details the challenges facing a RL based C-PBSM.
- In chapter 3 we give an overview on the LTE-A architecture. We focus on the physical layer, mobility management and interference management in LTE-A. We also present the system level LTE-A simulator used for simulations and validations, along with the assumptions, system model and interference model. We finally detail the SON functions we will be using in our use case studies as well as their iterative algorithms.
- Chapter 4 studies a RL technique base on a stochastic Multi-Armed Bandit (MAB) formulation under traffic stationarity hypothesis. We propose to use the UCB1 algorithm. The optimality, convergence speed and scalability of the approach are addressed. A case study is conducted on an LTE-A Heterogeneous Network (HetNet) scenario with different SON functions simultaneously deployed. An extension of this approach with improved convergence speed using a linear model assumption is then proposed and discussed. The LinUCB MAB algorithm is used for this scenario. Performances are studied and compared with the UCB1 approach.
- In chapter 5 we propose a distributed learning approach through a Distributed Q-Learning (DQL) framework. By distributing the learning processes, the learning becomes scalable and can hence be applied to bigger sections of the network. Furthermore, we propose a Software Defined Network (SDN) based architecture for piratical and flexible deployment of learning agents in the network. The DQL performances are compared with the previous centralized framework.

- In chapter 6 we tackle the Radio Access Network (RAN) environment dynamics, in particular the traffic variations in the network. We propose a solution based on contextual MAB. The learning process is done by a centralized learning agent, that learns simultaneously over different network clusters, located in different geographical locations in the network. The learning agent stores and updates its knowledge in a centralized knowledge data base. We show that this approach is able to efficiently learn and adapt over environment context variations, and that the acquired knowledge can be transferred and applied to different sectors of the network through an open loop control process. The speed of convergence is improved by increasing the number of clusters trained simultaneously.
- Chapter 7 is the last chapter of the manuscript. It summarizes and concludes the thesis while providing directions for potential future works and further improvements.

Chapter 2

Cognitive Policy Based SON Management

2.1 Introduction to Self-Organizing Networks

A SON function is a control loop which objective is to automate operational tasks in the network and optimize the network parameters and configurations. This automation and optimization is expected to reduce the CapPex and OpEx, that are likely to increase with future networks. SON functions can be typically divided into three main categories:

- Self-Configuration (Plug-and-Play): These SON functions automate tasks related to the deployment of new sites. Through plug-and-play functionalities, self-configuration SON functions are expected to reduce the manual effort related to hardware and software installation and configuration as well as network authentication (e.g. in LTE, an eNodeB (eNB) needs to discover neighboring nodes). Well known self-configuration SON functions are the Automatic Neighbor Relation (ANR), automatic Physical Cell Identity (PCI), Automatic Software Download (ASD) and Automatic Inventory (AI), etc.

- Self-Optimization: Self-optimization SON functions are in charge of continuously monitoring and optimizing the network radio and transport parameters, in order to preserve certain levels of Key Performance Indicators (KPI) in the networks and QoS for the users. Such functions include interference mitigation techniques through Inter Cell Interference Coordination (ICIC), HandOver (HO) optimization through Mobility Robustness Optimization (MRO), Mobility Load Balancing (MLB), Cell Range Expansion (CRE), Energy Saving (ES) through sleep mode optimization, etc.

- Self-Healing: In charge of detecting faults and anomalies in the network, and compensating performance and service degradation due to equipment malfunction. Self-healing functions include self-recovery of network equipment software, cell degradation diagnosis and predictions, Cell Outage detection and Compensations (COC), etc.

Moreover, SON functions are classified according to their location in the mobile network architecture [8]:

- Centralized SON (C-SON): These functions are located in the Operations and Maintenance Center (OMC). The parameter output of C-SON functions are passed to the eNB either on a periodic basis or on demand. Centralized solutions permit to manage big parts of

the network, with multi-layer and multi-RAT, and benefit from large datasets and statistics collected from different parts of the network. However, such solutions present certain drawbacks such as slow reactivity (minutes to hours), because KPIs and measurements are forwarded to a centralized server for processing before SON decisions and output are sent back to the eNBs. Furthermore, to preserve the scalability of the solution, the informations sent from the eNBs to the C-SON server are condensed and filtered. Hence less information would be available for the C-SON functions, compared to the information available at the eNB.

- Distributed SON (D-SON): In a distributed approach, the SON functions are located at a lower level, typically at the eNBs. They benefit from fast reactivity (seconds to minutes). D-SON functions usually monitor a single cell or a single site (sites can contain more than one cell, e.g. tri-sectorized sites), having hence a local knowledge.

- Hybrid SON: As its name indicates, this approach mixes the two previous solutions, by deploying simple optimization functions that require fast reactivity on the eNBs and keeping the complex tasks in the OMC.

SON functions have already been standardized and some of them are already deployed in today's networks. In fact, 3GPP started introducing and standardizing SON functions since it's 8th Release [6]. While Release 8 dealt more with self-planning functions such as AI, ASD, ANR and PCI, Release 9 [9] extended the SON standardization to self-optimization and included functions such as MRO, Random Access CHannel (RACH) optimization, MLB and ICIC. In Releases 10 and 11, 3GPP kept developing the SON concept and addressed more various use cases including HetNets scenarios such as enhanced Inter Cell Interference Coordination (eICIC), it also included green network use cases with energy saving modules as well as Minimization of Drive Tests (MDT) [10, 11]. In later releases several practical cases and enhancements were addressed such as troubleshooting and fault recovery for multi RAT and multi-layer HetNets in release 12, Active Antenna Systems (AAS) in release 13 as well as enhancements of Operations, Administration, and Maintenance (OAM) aspects of distributed MLB SON function [12, 13].

First experimental deployments of SON have shown great potential to automatically optimize the configuration of networks, reduce the complexity of network management, improve the QoS of end users and reduce CaPex and OpEx. However, a network with SON functions, that can be seen as a SON enabled network, is still far from realizing a network that is autonomous and self-managed as a whole. In fact, deploying several SON functions with many instances (a SON function instance is a realization of a SON function deployed and running on a section of the network) may generate conflicts in the network, resulting in bad network configurations and bad KPIs [7]. Also, on the one hand, different SON functions might have objectives that are not in line with each other, or with the operator high level objectives, e.g. KPI targets set by the operator for a certain context (time, network location, etc) and according to a certain operator strategy . On the other hand, the behavior of the SON functions depend on the parameters of the algorithm running in the control loop, as well as on the network environment where it is deployed (network topology, geographical location, traffic profile, technology, required services, etc) [1]. Therefore, each SON function still needs to be itself manually configured, depending on the network environment and the operator's high level objectives. Furthermore, the SON functions are generally provided by SON vendors to the network operators. The former share very few information about the algorithms running inside the functions because of proprietary issues, which makes them black boxes. Even though the operator is still able to steer the behavior of the black box SON functions by modifying certain configurations of the SON algorithms (typically thresholds, parameters range,

iteration steps, etc), managing and configuring black boxes remains a tricky task to achieve.

Today's SON enabled networks still require a significant amount of manual configuration and intervention to properly manage and orchestrate the SON functions, in order to avoid SON conflicts and be in line with the operator high level objectives (according to the 2018 Gartner report, 75 percent of current networks are still managed manually [14]). It is clear then that there is a need for an integrated SON management entity, that permits to autonomously and efficiently manage the many deployed SON functions and instances, hence providing higher level of automation in the network. This entity will be the interface between the operator and the network. Ideally, the operator will only have to define high level objectives, that will be given as input to the integrated SON management entity. The latter will autonomously translate these objectives into proper configurations of the deployed SON functions and instances, avoid conflicts between the SON, while monitoring KPI measurements and providing operation and optimization support to the network operator. The integrated SON management is hence a major step in the direction of networks that are self-organized as a whole.

2.2 Integrated SON Management

2.2.1 A Global View

The European project Semafour [1] proposed and studied an integrated SON management framework. The proposed general scheme is depicted in figure 2.1. The framework is constituted of different entities that interact with each other, in order to automate and optimize network operations.

Operator Interface The Operator Interface is situated on top of the framework. It takes the KPI target as input and forwards them to the Policy Based SON Management entity. The operational teams' task is therefor restricted to translating the operator's strategy and high level objectives into proper KPI targets.

Policy Based SON Management (PBSM) The PBSM is the central entity of the framework. Its objective is to autonomously translate the operator's KPI targets into configurations of the deployed SON functions, that we henceforward refer to as SON function Configuration Value sets (SCV sets). SCV sets include threshold configurations, parameter ranges, step sizes, turning off or on a SON function instance, etc. In other words, the PBSM finds the best configuration policy that satisfies the operator objectives, and enforces it in the SON enabled network that it is managing.

SON Coordination (SONCO) The SON coordination entity deals with SON conflicts. According to [15], the tasks of SONCO include:

- Avoiding as much as possible having conflicts.
- Detecting and diagnosing the conflicts.
- Providing conflict resolution mechanisms when needed.

SON conflicts can be classified into two types:

- parameter conflicts: when two SON instances target the same parameter and request opposite changes.

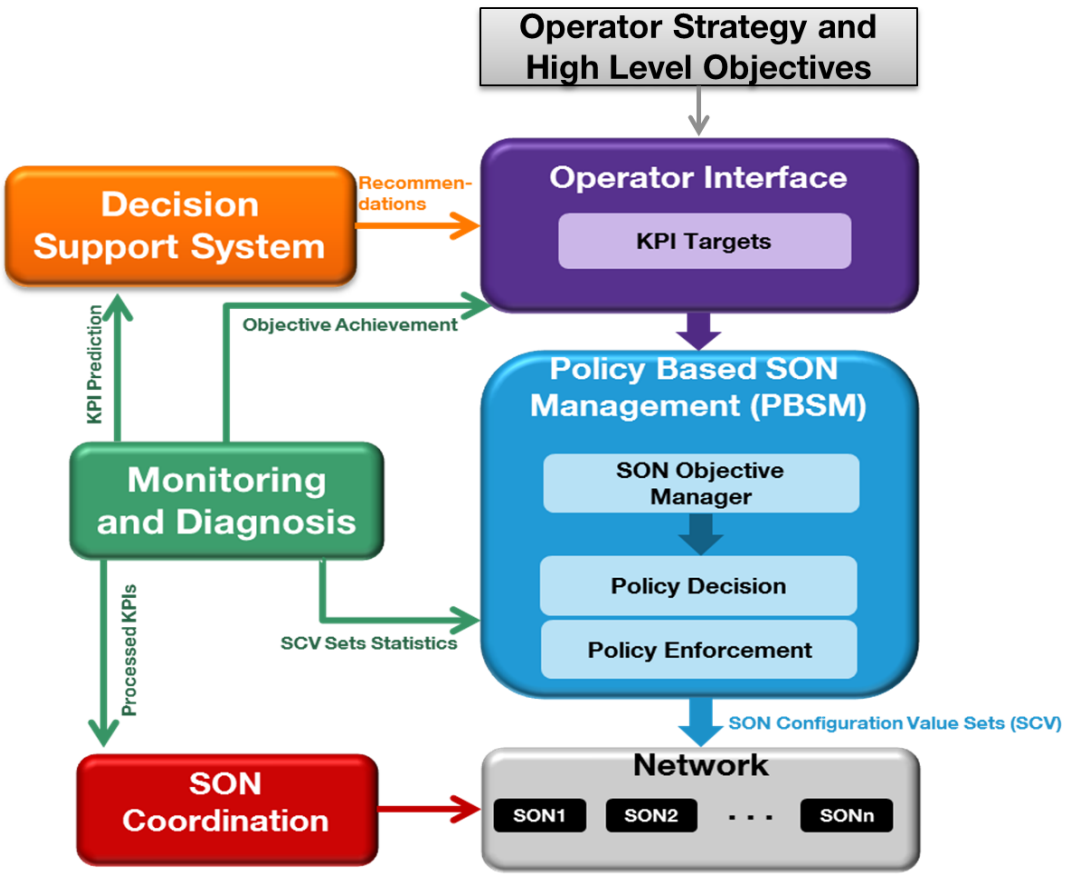


Figure 2.1: Integrated SON Management Framework [1]

- measurement conflicts: the parameter targeted by one SON instance affects the measurement of a different SON instance

SONCO is a crucial entity in the integrated SON management framework. Many studies can be found in the literature on the subject [16–21].

Decision Support System (DSS) The main task of DSS is to automatically provide recommendations towards the operator about network enhancements and extensions, when it is expected that the desired service level cannot longer be met by the SON management system in the near future. The recommendations include deployment of new sites, migration of existing sites to support new radio access technologies, installation of new hardware to support new frequency bands, etc.

Monitoring and Diagnosis (MD) MD insures the pre-processing of raw network KPIs so that they can be efficiently exploited and used by the different entities of the integrated management framework. KPI processing in the MD entity entails averaging, normalizing and standardizing the data, detecting and dropping outliers and missing data, clustering, classification, etc

2.2.2 Policy Based SON Management

The integrated SON management is, as explained earlier, constituted by a set of functionalities that collaborate in order to automate and optimize the network operation. In this thesis, we focus on the PBSM entity. As mentioned previously, the PBSM is the central entity of the Integrated SON Management framework. Its task it to autonomously translate the operator's KPI targets, that reflect the high level operator objectives, into proper SCV sets. In our work, we consider that the SON functions are provided by SON vendors as black boxes, i.e. the operator does not know the exact algorithm running in each SON function, due to patents and proprietary aspects of the algorithm developed by the SON vendors. The operator can however steer the behavior of these functions by changing their SCV sets, according to its objectives (figure2.2).

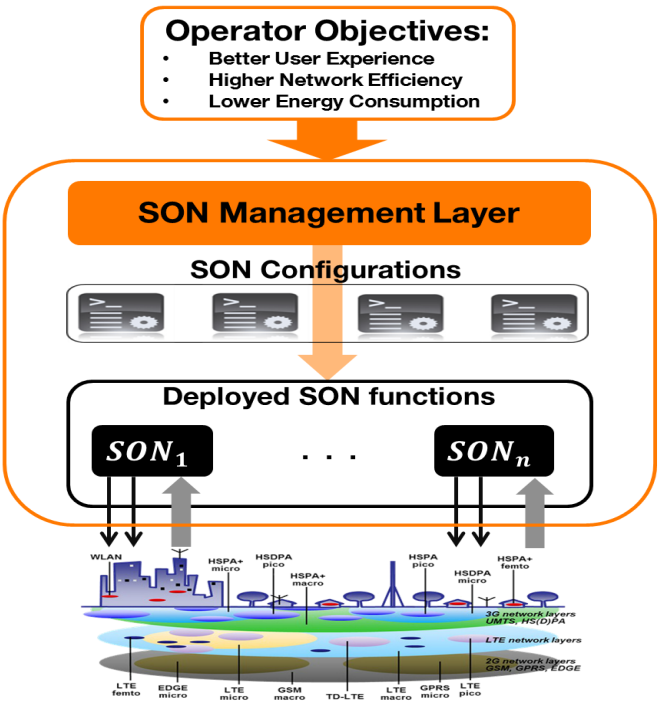


Figure 2.2: PBSM Scheme

Several approaches have already been studied to automate the PBSM process [22–25]. These approaches consider the SON functions to be black boxes as well. They rely on input models, provided by the SON vendor or generated by the operator, to derive the SCV sets policy. The considered input models are the following:

- Objective Model: It describes the objectives of the operator. It can be formulated in different forms. In [22] for instance, it takes the form of "IF *condition* THEN *KPI target* WITH *priority*". The condition can be the time of the day and the cell location for example. The KPI target defines the KPI that should be optimized, for example minimizing drop call rate, or maximizing handover success rate. The priority represents the importance of the corresponding KPI target for the operator. In [24] and [25], KPI targets are not only max and min requirements, but can also be intervals e.g handover success rate $> 95\%$. And priorities are replaced with weights, allowing a weighted satisfaction of KPI targets.

- Context Model: Taking the cell context into consideration is very important because it im-

pacts the behavior of SON functions instances. The context model provides a description of the system, the cell's properties, and eventually derives a cell class definition. These properties can be compared with the conditions stated in the objective model in order to derive the policy. Cell context can include time (since it is strongly related to traffic), cell location, network topology, etc.

- SON Function Model (SFM): It is a mapping, for each SON function, from KPI targets to SCV sets. SFMs are considered to be provided by the SON vendors with the SON functions. They are typically generated in a simulation environment or over a network test cluster where the SCV sets for each SON function are tested and output KPIs are determined. In [23], the authors describe a generation and testing process for SFMs, where the operator simulates the network with different SCV sets for each SON function. Several approaches are possible. The first approach would be to run all the SON instances of each SON function deployed in a certain network section. A second approach would be to define cell clusters with similar network contexts and test the SCV sets only on the SON instances deployed in these clusters. A third approach would be to evaluate a SFM for each and every SON instance.

The PBSM operates by comparing the objective model with the SFMs mappings and the context model, and outputs optimal SCV sets to be enforced in the corresponding SON functions. This approach was extended to include network KPI measurements [25]. The idea is to gradually replace the static SFMs provided by the vendors by real network measurements once the network measurements pool collected from the network becomes big and reliable enough. However, the process is still strongly relying on static input models, which often do not represent exactly the actual network where the PBSM is operating, nor the dynamics of the environment. Besides, SFMs are considered to be provided by the SON vendors. This consideration is rather problematic for the operator in the case of trust issues with the vendor, and because the SON vendor is most probably willing to provide these SFMs with extra charges, especially that these models will require probably constant updates and improvements. Alternatively, if the operator chooses to generate itself the SFMs, then it would require a lot of simulations and time effort to simulate and test the SCV sets. Furthermore, pre-simluated models are generated for each SON function individually. They hence do not depict well enough the interactions between the different SON functions when deployed simultaneously in the network. In the following section, we motivate the need to improve the state of the art PBSM to meet with the requirements of future networks and we discuss how the PBSM can be enhanced with cognitive capabilities.

2.3 From Autonomic Management to Cognitive Management

2.3.1 The Motivation Behind Introducing Cognition

It is clear today that future networks, notably 5G RAN, will be highly flexible and adaptive. They are expected to meet the increasing user expectations, absorb much greater amounts of traffic, and provide services with heterogeneous and high quality requirements. Extensive research is currently ongoing in several Fifth Generation Public Private Partnership (5G-PPP) projects, for example, 5G-NORMA [26], METIS II [27], and FANTASTIC-5G [28], to define the basics of the RAN architecture, as well as new data link and physical layer concepts for the air interface. In addition, the corresponding standardization activities have started in 3GPP [29]. 5G RAN will be enriched with a plethora of new features and technologies such as Device-to-Device (D2D), Vehicular-to-Anything (V2X), Massive-Multiple Input Multiple Output (M-MIMO) antenna systems, Internet

of Things (IoT), Machine Type Communication (MTC) or new spectrum. 5G will also be vertical driven [30] with a multitude of different requirements corresponding to the specific needs of the vertical industries (e.g. automotive, e-Health, multimedia and entertainment,...) in addition to the classic operator objectives (e.g. network capacity and operational efficiency). This high flexibility and adaptability translates into a tremendous number of possible network configurations. This will certainly induce more complexity and ambiguity in the deployment, configuration, optimization and operation of future networks.

It is true that automation through SON functions has considerably facilitated the operational task, and helped reduce CapEx and OpEx. Also, existing studies on integrated SON management concepts are promising. However, they will not be sufficient to efficiently optimize and configure future networks, especially with the shortcomings suffered by the state of the art PBSM as mentioned previously. In fact, in order to meet with the high level of flexibility and dynamics, the required degrees of automation of future networks has to be higher. Automated processes must become more aware of their environment dynamics and changes, be able to learn and reason from past experience, and continuously adapt and improve their decisions and policies. In other words, automation has to be enhanced by cognitive capabilities.

Cognition in wireless networks was first introduced by Mitola when he presented the concept of cognitive radio [31]: "The term cognitive radio identifies the point at which wireless personal digital assistants and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications: (a) to detect user communications needs as a function of use context, and (b) to provide radio resources and wireless services most appropriate to those needs." Cognition is inspired by the reasoning, adapting, and learning capabilities of the human brain. A cognitive system is hence an autonomic and intelligent system.

Regarding the SON framework, cognition can be introduced in two different levels. First at the level of each SON function. Introducing learning capabilities in the SON algorithm itself will permit the SON function to intelligently adapt its decisions with the network context. The literature is rich with works and studies to enhance the SON functions with learning capabilities, through Machine Learning (ML) e.g. for MLB [32, 33], for CRE [34, 35], for eICIC [36, 37], energy saving [38, 39],etc. Second, cognition can be introduced at the SON management level, more precisely in the PBSM process. The Cognitive PBSM (C-PBSM) learns the effect of SON functions on the network and predicts changes in terms of performances when changing the SCV sets of the SON functions. The C-PBSM learns by interaction with the real network, and enhances its policies and decisions by learning from the impact and feedback of past decision in the network. The C-PBSM is hence a powerful SON management framework, that is able, through its cognitive loop, to insure the required levels of automation and intelligence for future 5G mobile networks.

In this work, we are interested in introducing cognition in the SON management process, i.e. designing and evaluating a C-PBSM. ML algorithms are a key tool and bedrock to render the PBSM cognitive [40]. In fact, ML algorithms can be useful to enhance different parts of the process:

- Improving Input Models: ML techniques can improve the input models that the PBSM relies on. For example the context model can be enhanced through unsupervised learning, in order to provide a reliable cell clustering. That is, using unsupervised learning techniques such as associations rules, clustering, outliers detection, etc [41], similar cells can be grouped in clusters. SON instances deployed in cells belonging to the same cluster are expected to behave similarly. Consequently, the PBSM policies shall be derived per cell clusters instead of per SON function instance, hence reducing considerably the dimensionality of the problem.

- Translation of High Level Operator Goals: Another level where ML can be introduced to the PBSM, is to enhance the transformation procedure that translates semantic goals (the operator high level objectives) into quantitative objectives (KPI targets and their respective priorities). Learning algorithms will permit to understand and interpret the high level goals expressed in human language, and then define the technical objectives that correspond to these high level goal.

- Redefinition of the PBSM Management Process: the state of the art management process of the PBSM configures the SON functions based on input models such as the SFM, the context model, the objective model [1]. Based on these modes, the PBSM outputs a set of rules, according to which the SON functions are configured. The process is performed in open loop, with no feedback from the real network to validate the decisions (knowing the the input models often introduce bias in the decision process with respect to the real network). Even though in [25] an approach is proposed to replace the SFM by real network measurements over time, the approach still relies strongly on the pre-simulated SFM, and does not rely on any learning framework with theoretical guarantees. Learning algorithms, more specifically RL, can help overcome this drawback by introducing a cognitive closed control loop, that will permit the PBSM to observe, learn, reason and adapt with its environment.

- Knowledge and Data Management: Learning algorithms often need a big amount of data. These data sets are essential to create the environment awareness needed to build the cognitive loop. These data sets are not limited to the radio environment. They originate from various sources e.g. subscriber metrics, network KPIs, hardware configurations, etc. In order to be efficiently exploited by the learning processes, the data sets need to be stored in unified dashboards, providing smart correlation across the various data sources. Data mining and big data algorithms are crucial to build such dashboards.

Self-organization needs to be enhanced with cognition, in order to be able to cope with the prime requirements and high flexibility of future networks. Nevertheless, cognitive automation is not achievable by a single big algorithm. It is rather a multitude of automated, intelligent tasks, on different levels of the process. As described above, there are many axes where cognition can be implemented through ML, and the advantages and enhancements it is able to bring to the system are evident. On the other hand, even though ML techniques have gained a lot of interest in many domains, and have already proven to be very useful in automating tasks and optimizing revenues and time, their applications in complex systems such as mobile network management, is still a challenging task. For instance, applying ML in telecommunications requires more work and expertise than simpler domains of applications, because of the many constraints and requirements that have to be taken into consideration such as the scalability and robustness of the algorithms, complex and dynamic radio and core network environments, standardization constraints, the operator's infrastructure, data processing and storage challenges, user privacy, etc. Implementing ML in all the described parts of the PBSM process and succeeding in making them work together would be a big step towards an intent based network management of mobile radio networks [14]. The operator would then tell the network what he wants, i.e. its intent, rather than what to do. The operator's intent is then automatically interpreted across different devices of the network. According to many industry analysts, intent-based networking systems will be necessary to manage the networks of the future, connecting data centers, public clouds and IoT. However, intent based networking is still in an early stage of definition, conception and development.

In this thesis, we particularly focus on a major part of the SON management enhancement, that is the redefining the PBSM process by means of RL, so that the new process, i.e. C-PBSM,

can learn and derive optimal policies only by interacting with the real network. In the framework proposed in this work, the C-PBSM does not rely on any input model. It only needs the operator target KPIs to derive, from scratch, the appropriate SCV sets to be enforced in the network. The approach is thoroughly explained in the following sections. Note that henceforward, by operator objectives we mean KPI targets.

2.3.2 Reinforcement Learning

The cognitive loop of the C-PBSM can be implemented through RL. RL as defined in [2], is learning how to achieve a goal by interacting with the environment. The process is illustrated in figure 2.3. The learning agent observes the state of the environment and based on this state, it takes an

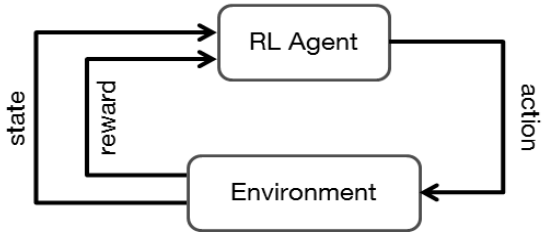


Figure 2.3: RL Process [2]

action according to some policy. The environment reacts by triggering a state change according to a probability distribution that depends on the enforced action, and generates a numerical reward. The agent observes the new state and the reward and takes a new action following the new observation. The aim of the agent is to maximize the long term perceived reward. In other words, the objective is to find the optimal policy π^*, that is the mapping between states and actions that maximizes the long term reward. Note that throughput this work, optimal policy or action refers to the best policy or action in the defined set of actions, and not the optimal solution of the problem, which can lie outside the considered set of actions. The RL model consists of:

- A discrete set of environment states S.
- A discrete set of agent actions A.
- A reward function $r : S \times A \rightarrow \mathbb{R}$.

Let r_t be the immediate perceived reward at iteration t. Then the long term perceived reward has the following expression:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.1}$$

Where $0 \leq \gamma \leq 1$ is a discount factor. R_t is the sum of the discounted rewards that the agent receives over the future. In other words, R_t represents the agent's insight of the rewards in future states starting from iteration t. As γ approaches to 1, future rewards are taken into account more strongly, the agent becomes hence more farsighted.

Markov Decision Process (MDP) The dynamics of RL problems are often modeled with MDPs. MDPs are stochastic processes that satisfy the Markov property. This property is satisfied if and only if:

$$P(s_{t+1} = s', r_{t+1} = r|s_t, a_t, r_t, a_{t-1}, ..., r_1, s_0, a_0) = P(s_{t+1} = s', r_{t+1} = r|s_t, a_t) \quad (2.2)$$

Equation 2.2 means that a process is an MDP if the probability of being in a state s' and perceiving a reward r at iteration $t+1$ depends only on the state and action of the previous iteration t. In other words, the environment's response at a certain iteration depends only on the state and action of the previous iteration. Having the Markov property, one can express the transitions probabilities (the probability of each possible next state s', given action a and state s) and the expected value of the next reward (given action a and state s), which describe the dynamics of an MDP as follows:

$$\mathtt{P}^a_{ss'} = P(s_{t+1} = s'|s_t = s, a_t = a) \quad (2.3)$$

$$\mathtt{R}^a_{ss'} = \mathbb{E}[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'] \quad (2.4)$$

Value Function and Q Function There is a variety of RL algorithms in the literature, however, almost all of them are based on the estimation of the value function and the action-value function. The value functions is defined as follows:

$$V^\pi(s) = \mathbb{E}_\pi\{R_t|s_t = s\} = \mathbb{E}_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\} \quad (2.5)$$

It reflects how good it is for the agent, in terms of the expected perceived rewards in the future states, to be in a certain state s when following a policy π. The optimal value function is therefore:

$$\begin{aligned} V^*(s) &= \underset{\pi}{max} V^\pi(s) \\ &= \underset{a \in A}{max} \sum_{s'} \mathtt{P}^a_{ss'}[\mathtt{R}^a_{ss'} + \gamma V^*(s')] \end{aligned} \quad (2.6)$$

Respectively, the action-value function, also known as the Q-function, reflects how good it is for the agent, in terms of the expected perceived rewards in the future states, to be in a certain state s and choosing action a, when following a policy π:

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_\pi\{R_t|s_t = s, a_t = a\} \\ &= \mathbb{E}_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\} \end{aligned} \quad (2.7)$$

And the optimal Q function:

$$\begin{aligned} Q^*(s, a) &= \underset{\pi}{max} Q^\pi(s, a) \\ &= \sum_{s'} \mathtt{P}^a_{ss'}[\mathtt{R}^a_{ss'} + \gamma \underset{a' \in A}{max} Q^*(s', a')] \end{aligned} \quad (2.8)$$

Equations 2.6 and 2.8 are known as the Bellman's optimality equations for V^* and Q^* respectively. They are derived using Bellman's equations [2]. RL algorithms' objective is to find the best strategy to estimate V^* or Q^* and derive the optimal policy based on these estimates.

One interesting category of RL algorithms is the Temporal Difference (TD) family of algorithms. Unlike other kinds of approaches (Dynamic Programming and Monte Carlo Methods),

TD algorithms do not need to know the environment transition model, which is often unknown and complicated to evaluate in complex real life problems such as ours. Furthermore, TD algorithms are bootstrapping algorithms, i.e. they update their estimates of the values of states based on estimates of the values of future states. They do not wait for the final outcome of the experience to update their strategy, which practically accelerates their convergence. Q-learning is a well known RL-TD algorithm that was shown to converge to an optimal policy in [42]. Q-learning and its variants were applied to multiple real life problems and have shown to be very useful in learning optimal polices. The Q-learning algorithm is described in Algorithm 1.

Algorithm 1 Q-Learning

Initialize $Q(s,a)$ arbitrarily
Initialize s
for t=1,...,T
- pick action a for state s according to ϵ-greedy policy:

$$a = \begin{cases} \underset{a \in A}{argmax}\{Q(s,a)\} & \text{with probability } 1-\epsilon \\ rand(A) & \text{with probability } \epsilon \end{cases}$$

- observe new state s'
- observe perceived reward $r(s,a)$
- update Q function as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \underset{a' \in A}{max} Q(s',a') - Q(s,a)]$$

end for

$rand(A)$ refers to picking a random action from the set of actions A. Note that the ϵ-greedy policy is used in order to ensure exploration in the learning algorithm, and avoid being stuck in a local minima. α is the learning step.

However, a well known issue in the RL framework is the scalability problem. In fact, when the set of actions or set of states grows, the time required by the RL algorithms to explore the set of possible policies increases as well, leading hence to a slow convergence towards the optimal policy. In fact, classic RL algorithm fail to efficiently deal with such environments, with large sets of actions and states. Unfortunately, most of complex real life problems have huge action and state space. Several methods can however be used to alleviate the scalability problem such as the use of approximation functions, neural and fuzzy networks [2,43,44]. Another efficient way to deal with the scalability issue is through distributed learning. The latter consists in decentralizing the learning process by deploying several RL agents, where each agent learns based on the observations of a fraction of the environment (local environment) where it is deployed. The main advantages that can be achieved by distributing the learning are 1) scalability since the number and capabilities of RL agents can easily be adapted, 2) computational efficiency due to parallel processing, 3) maintainability because of the modularity of the distributed approach, the system is easier to maintain and 4) robustness in case the agents have a degree of redundancy [45, 46]. Nevertheless, one should pay attention to the environment dynamics in multi-agent settings, where each agent learns independently from the other. In fact, the actions taken by the independent agents will affect their local environments but also the global environment, that includes the local environments of other agents as well. Consequently, the agents' environments are no longer strictly MDPs. Instead, for each agent, the other agents are approximated as part of the environment. The learning model is

hence an approximated MDP where the Bellman's equations and optimality criterion can still be applied (equations 2.6 and 2.8). Distributed Q-learning approaches have shown to be efficient in multiple applications for example in [47, 48].

2.3.3 Reinforcement Learning via Multi-Armed Bandits

A particular case of the RL problem is the single state case, also known as the Multi-Armed Bandit (MAB) problem. In this formulation, the agent's actions do not change the state of the environment. The agent is faced with a single state where its objective is to find the optimal action using the most efficient strategy to do so. This problem was extensively studied in the literature. Recent theoretical advances lead to multiple MAB strategies and algorithms [49–52], making the MAB a strong candidate for many real life learning problems.

The MAB problem was first introduced in [53]. In its simple form, the MAB problem is formulated as follows: An agent is confronted with a set of actions known in the literature as arms. Let A and K be the set of arms and the number of possible arms respectively. Each one when pulled, generates a reward. In a common setting of the problem, the agent is only aware of the reward of the arm after pulling it. The problem is hence: how should the agent pick the arms in order to maximize its cumulated reward? The MAB process can be described in the following:

Algorithm 2 MAB Process

for t=1,...,T
- Agent selects an arm $a_t \in A$
- Environment outputs a vector of rewards $\mathbf{r}^t = (r_{t,1}, r_{t,2}, ..., r_{t,K}) \in [0,1]^K$
- Agent observes only r_{t,a_t}

end for

It is a sequential decision problem where the learning agent is faced with an exploration-exploitation trade-off. In fact, if the agent only exploits the arms (always pulling the arms it thinks is giving it the higher reward, based on the information it has about the system at a certain iteration t), it will probably be stuck in a sub-optimal decision because its knowledge about the arms was not sufficient. However, if it only explores all the time, it will certainly learn more about the arms' rewards, but it will fail to maximize its cumulative reward because it is never exploiting the acquired information through exploration. Therefore, the notion of regret is introduced in order to evaluate the algorithm's efficiency: it is the difference between the reward that the agent actually received by pulling some arm and the one that it might have received if it had pulled the best arm. Hence the objective is to minimize the average regret by maximizing the chance of pulling the best arm at each iteration.

There exists different frameworks for the MAB, depending on the reward feedback model, the dependencies that may exist between the arms, the available side information, etc. There are two main reward feedback models: stochastic feedbacks where the reward of an arm is sampled from some unknown distribution, and adversarial feedbacks where the reward of an arm is chosen by an adversary and is not necessarily sampled from any distribution. Radio networks are usually modeled as stochastic environments, whence our interest in stochastic MAB. In the following we describe the MAB formulations of interest.

Stochastic MAB In the stochastic MAB the rewards of each arm are supposed to be an independent and identically distributed (i.i.d) sequence following some unknown distribution, specific to each arm. The main entities of the stochastic MAB are the following:

- A is the set of arms.
- ν_a is the reward's unknown probability distribution of arm a.
- μ_a is the unknown average reward of arm a.

An efficient way to insure a trade-off between exploration and exploitation is by bounding the regret of a MAB algorithm. As explained previously, the regret expresses the difference between the reward that the agent actually received by pulling some arm and the one that it might have received if it had pulled the best arm, at a certain iteration. In its general form, the cumulated regret after n plays can be formulated as [50]:

$$R_n = \underset{a \in A}{max} \sum_{t=0}^{n} r_{t,a} - \sum_{t=0}^{n} r_{t,a_t} \tag{2.9}$$

In the stochastic framework, it makes sense to bound the regret of an algorithm in expectation. That is:

$$\mathbb{E}[R_n] = \mathbb{E}[\underset{a \in A}{max} \sum_{t=0}^{n} r_{t,a} - \sum_{t=0}^{n} r_{t,a_t}] \tag{2.10}$$

Because of the max term inside the expectation, it is difficult to deal with the expression above. Instead, the pseudo-regret is considered in the analysis of the stochastic MAB, defined as follows:

$$\overline{R}_n = \underset{a \in A}{max} \mathbb{E}[\sum_{t=0}^{n} r_{t,a} - \sum_{t=0}^{n} r_{t,a_t}] \tag{2.11}$$

The expectation is taken with respect to the draw of arms and rewards. Let $\mu^* = max(\mu_a)$ for all $a \in A$. Then the pseudo-regret can be written as:

$$\overline{R}_n = n\mu^* - \sum_{t=0}^{n} \mathbb{E}[\mu_{a_t}] \tag{2.12}$$

In the stochastic framework, the MAB problem is how to find the optimal arm (the arm with the highest mean reward μ^*) while minimizing the pseudo-regret in equation 2.12. In the MAB framework, it is important to find a tight upper bound to the regret of the MAB algorithms. The more this upper bound matches with the lower regret bound that can be possibly achieved, the more the exploration strategy of the algorithm is optimal and efficient. There exists several strategies that deal with the stochastic MAB problem such as strategies Upper Confidence Bound (UCB) [49,54], Thompson Sampling [52,55], the Probably Approximately Correct (PAC) setting [56,57], ϵ-greedy strategies [2, 49], etc

Linear MAB An interesting extension of the stochastic MAB framework is the linear MAB. In fact, in the stochastic MAB formulation described previously, the arms are considered to be independent. However, in many cases, a structure can be found or reasonably assumed between the arms or between the arms and the rewards. When such structure exists or is assumed, more efficient strategies can be derived, outperforming the standard stochastic MAB algorithms. A natural assumption for the structure is to consider the expected reward function to be linear with respect to actions. Linearity is a standard assumption (think, for instance, of linear regression) and naturally occurs in many optimization and learning applications. Many works in the literature study Bandit problems with linear assumptions. Such models where first considered and studied in

[51] where each arm is viewed as a vector in $\mathbb{R}^n$ and the rewards are linear functions of this vector. Several strategies and algorithms were proposed and analyzed for this framework ([58, 59]). In [60] the authors assume that the expected reward of each arm is a linear function of an unknown scalar with a certain distribution. They study this framework and derive optimal strategies to find the optimal policy. This study is extended in [61] where the model assumes that the expected reward is a linear function of a multivariate random vector. In [62], a practical algorithm, the LinUCB, based on the upper confidence bound is proposed and its regret bound are theoretically analyzed and shown to be optimal.

Contextual MAB Contextual bandits are MAB problems with side contextual information, besides the reward feedback of the arms. In most real life problems, the perceived reward depends on the taken action but also on the given context of the environment. In this case, the algorithm's objective is to find the optimal mapping between contexts and actions rather than the best action. For example in web-add recommendation problems, the click-through rate (perceived reward) depends on the add the recommender shows (action) on the web pages, but also on the profile of the web user information e.g. age, gender, region, etc (context information). A straightforward approach to deal with changing contexts would be to consider a stochastic MAB process per context. Hence finding an optimal policy per context. However, such approach would require a lot of time to converge because each MAB process would be updated only when its corresponding context is observed. Also when faced with a large number of contexts, then this approach would simply be infeasible. Instead, contextual MAB deals more efficiently with this problem by taking certain assumptions about context observations. We can find many examples in the literature. For instance in [62], a linear dependence between the contexts and the reward is assumed. In [63], a similarity information between the context-arms pairs is supposed to be known and exploited by the algorithm. Other algorithms assume a Lipschitz continuity between arms and contexts coming from a metric space [64]. In this work we implement the bandit forest contextual MAB algorithm, which belongs to the decision trees family. This algorithm does not need similarity information, nor linearity assumption which makes it more suitable for application where similarity information is not available or hard to evaluate, and where the dependence between contexts and rewards is not linear. Instead, it considers that there is a subset of context variables that are more relevant than the rest. It hence reduces its exploration space by considering this subset of variables, and discarding the others. Furthermore, it scales well with the number of context variables, as will be explained in chapter 6, which is a very useful characteristic when dealing with complex environments such as RAN. The bandit forest algorithm was proposed and analyzed in details in [65]. Note that there is a major difference between the context definition of the contextual MAB and the state definition in the more general RL framework: in the contextual MAB, the agent's actions do not change the state of the environment (i.e. the contexts), whereas in the general RL framework (e.g. Q Learning), the state of the system at iteration t depends on the action taken at the previous iteration (equation 2.3).

2.3.4 The Cognitive PBSM

In this thesis we propose an approach to enhance the PBSM with cognitive capabilities i.e. C-PBSM, through RL. We consider hence a RL agent, that explores the set of possible SCV sets and learns the optimal SON configurations, on the defined set of SCV sets, that maximizes the operator's objectives (given as input to the agent). The learning is done from scratch by interacting with the real network (no input models). The approach we propose is depicted in figure 2.4.

We consider a SON enabled network, with distributed SON functions i.e. each SON function

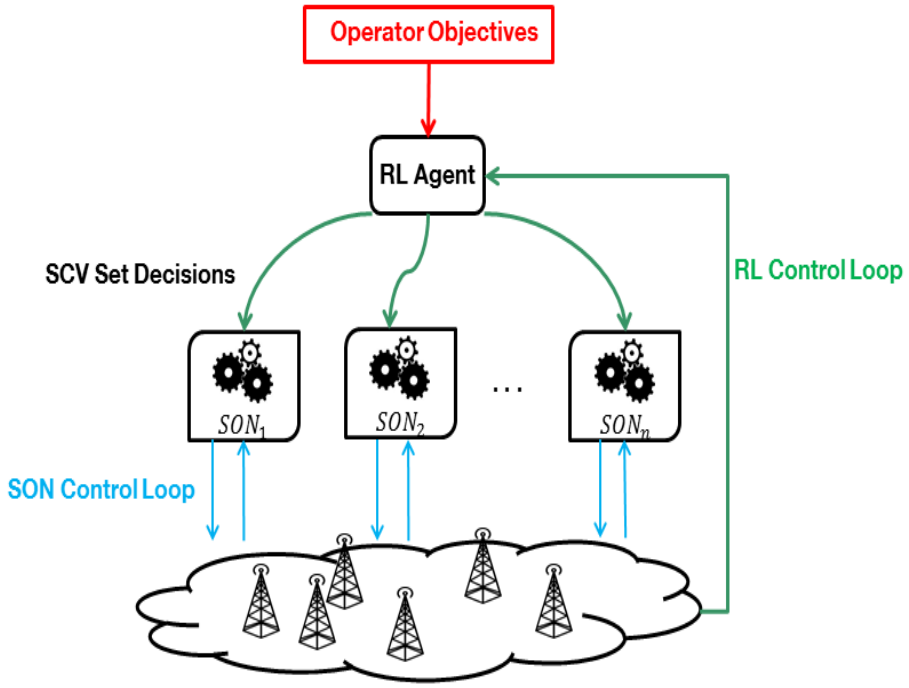


Figure 2.4: Learning Functional Scheme

is distributed through several instances on different cells. Each instance has its own control loop: it collects local KPI measurements and changes certain parameters depending on its objective, algorithm, and configuration (SCV set). The operator can steer the behavior of the instances of a SON function in a direction or another by configuring them through the SCV sets. On top of the SON layer, we consider a learning agent. The agent learns and enforces autonomously the optimal SON configuration policy, according to the objectives specified by the operator. The optimal policy in this case is the SCV set combination of the deployed SON instances that satisfies to the best the operator's objectives. The learning process is done by RL: KPI measurements are reported and SCV sets are enforced through the RL control loop.

The RL loop enhances the SON management process with cognitive capabilities. Indeed, the C-PBSM is aware of the effects of the SON configurations in the network, through the network KPIs feedback to the RL agent. This latter uses these KPIs to build the knowledge about the different SCV sets and their impact on the network. The optimal policy is derived based on the acquired knowledge and the objectives of the operator. This approach does not rely on any input model. It learns and derives its policies from scratch. Note that the C-PBSM is able, besides configuring the SON functions through SCV sets, to change directly certain network configurations. For instance, it can choose to turn off certain cells, if the operator's objective includes enhancing energy efficiency.

This said, running learning algorithms online in real life complex systems such as the future RAN, presents several challenges. In its learning phase, a RL algorithm explores the set of actions in order to build its knowledge and takes often sub-optimal actions during the process. This means that in the learning phase, network KPIs, and consequently the QoS in the network, might degrade or not be compatible with the operator objectives. The convergence speed is therefore of the highest importance. Especially that in our case, the RL is an external control loop (RL Control Loop), whereas each SON instance has its own internal control loop (SON Control Loop),

as shown in figure 2.4. This means that, at each RL iteration, the agent configures the SON functions instances with SCV sets. The agent has then to wait for a certain number of SON (internal) iterations, so that the SON functions instances converge to a stable behavior. Then the agent can start collecting measurements that will asses the impact of the applied SCV sets in the network. Otherwise, if the measurements are collected while the SON functions instances are still adapting with the new SCV sets, then these measurements would be distorted and would not properly represent the SCV sets impact in the network. To do so, the C-PBSM makes use of a guard window, where it does not collect any measurements from the network. Note that the SON functions use a similar guard window on a smaller time scale, when changing network parameters. The mechanism is explained in the next chapter (figure 3.6). Consequently, each RL iteration includes several SON iterations. It is hence a slow control loop and its speed depends on the dynamics of the SON functions. The convergence speed of the C-PBSM is hence critical and needs to be enhanced through the design of the RL algorithms.

Moreover, the scalability of the RL algorithms has to be taken into consideration, notably the scalability regarding the action space. In fact, the set of actions of the C-PBSM is basically the combination of the SCV sets of the SON instances of all the SON functions deployed in a certain network section. Consequently, assigning a large network section to the C-PBSM to manage, with multiple SON functions and multiple possible SCV sets will lead to a huge action space, slowing down considerably the convergence of the learning process.

Also, the interdependencies between neighboring cells should not be neglected e.g. when changing certain configurations of a single cell, for example antenna tilt or transmit power, neighboring cells might be affected in term of interference, coverage, traffic load, etc. RAN environments are also very dynamic, especially when it comes to traffic dynamics and their impact on network KPIs and users QoS. The C-PBSM should ideally be able to efficiently deal with these dynamics, and adapt simultaneously with the operator objectives changes (operator objectives will probably change depending on the network context, operator strategy, social events, telecom regulator requirements, etc).

Finally, it is unfeasible for the C-PBSM to manage an entire network, because of computational, scalable, optimal and practical constraints. These constraints should be throughly taken into consideration. There is hence a necessity to efficiently distribute the learning and transfer the acquired knowledge between different network sections.

2.3.5 Contribution of the Thesis

In this thesis, we develop and study a cognitive and integrated SON management framework, that efficiently manages the many deployed SON functions according to the operator's objectives and requirements. The contribution of the thesis can be summed up in the following:

- Modeling the Cognitive SON Management Problem: The C-PBSM learns the optimal SON configurations through interaction with the real network, whereas in the state of the art PBSM, the process relies on pre-simulated input models that are generated through simulations or tests over some typical network sections. In this thesis we model the cognitive loop of the C-PBSM as a RL problem where the C-PBSM learns the optimal policies by interacting with the real network. The proposed approach does not rely on any input model, the policies and the knowledge are built from scratch. A generic functional scheme is proposed for the process.

- Analysis and Evaluation of a Centralized Stochastic MAB Approach: We model the C-PBSM as a sequential learning problem. We propose an efficient RL solution based on

stochastic MAB. We analyze the optimality and convergence. We propose to enhance the convergence speed of this approach using a linear model. The optimality and convergence rates of both approaches are thoroughly studies.

- Analysis and Evaluation of a Distributed RL Approach: We propose a distributed RL solution for the cognitive SON management that is able to find optimal policies satisfying the operator's objectives. We study the scalability of this approach and compare it with the stochastic MAB. We also propose a SDN based architecture that can be used to implement the distributed RL agents.
- Analysis and Evaluation of a Context Aware C-PBSM Based on Contextual MAB: The proposed approach is able to learn over varying network contexts, including traffic variations. The built knowledge is transferable to other sections of the network. The proposed C-PBSM is able to learn simultaneously and collaboratively over different network clusters. The optimality and convergence rate of the algorithm is thoroughly studied.
- All the approaches were tested on practical use cases with different SON functions, multi layer HetNets. The results were derived using an LTE-A, 3GPP compliant system level simulator developed in C++ and adapted to the studies of SON management.

Chapter 3

System Model

3.1 Introduction

SON functions for 5G scenarios are still today in the research phase and are not well defined yet. Therefore in this thesis, we test and evaluate our approach on LTE-A scenarios. However, the approach as proposed and described in the previous chapter is generic, and can be applied and adapted to SON functions using 5G features. Consequently, in this chapter we give an overview on LTE(-A) system and architecture. We focus on the RAN part, that is the Evolved Universal Terrestrial Radio Access Network (E-UTRAN). We give an overview of the physical layer transmission schemes, mobility and handover procedures, as well as the interference model. We finally give details about the system level simulator used for validations, and describe the SON functions we use in the use cases throughout this work.

3.2 LTE-A Systems

3.2.1 Introduction

Mobile communication technologies are divided into generations. First Generation (1G) was the first analog mobile network system. It was released in the 1980s. The Goupe Spécial Mobile (GSM) project was initiated in the mid-1980s to develop a European mobile-telephony system. The GSM standard was based on Time Division Multiple Access (TDMA), followed by a later development of Code Division Multiple Access (CDMA) in the USA. 2G was the first digital mobile system in which the Short Message Service (SMS) and circuit switched data services were introduced. 3G was the first mobile system handling broadband data with the use of higher bandwidth radio interface of Universal Terrestrial Radio Access (UTRA). 3G was followed by the next generation, the 4G LTE system with its Evolved Universal Terrestrial Radio Access (E-UTRA), providing better services and higher bandwidth. LTE-A is an evolution of LTE that was standardized by the European Telecommunications Standards Institute (ETSI) and 3GPP in 2011 (3GPP release 10), while keeping a complete compatibility with LTE. LTE-A is capable of providing peak throughputs higher than 1Gb/s. The task of developing future generation mobile technologies, notably 5G systems, is undertaken within the 3GPP framework, and is currently under development. Many key features of 5G are currently under standardization, and will drastically change and improve the performances of mobile networks, such as massive MIMO (multiple-input multiple-output), millimeter waves (24-86GHs), slicing, D2D communications, etc.

In the following we present the LTE-A architecture. We then develop about the physical layer, mobility and interference management. We then present the LTE-A system level simulator we

have developed and used during the thesis. Finally we discuss the management framework and self-organization in LTE-A systems.

3.2.2 System Architecture

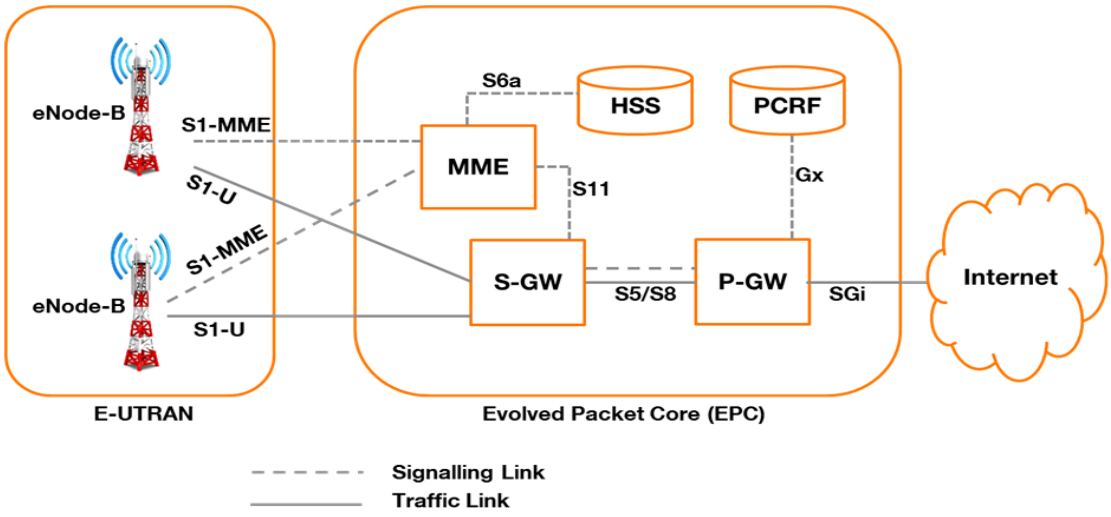


Figure 3.1: LTE-A System Architecture

The LTE-A core network, also known as Evolved Packet Core (EPC) and the LTE-A RAN, known as E-UTRAN, are depicted in figure 3.1. The EPC supports the user mobility, wireless data connections and authentication in LTE-A. A single EPC can serve several radio access technologies. The main entities of an EPC are:

- Home Subscriber Server (HSS): is in charge of storing and updating all the user subscription information such as user identification and addressing user profile information.
- Serving Gateway (S-GW): is the user-plane node connecting the EPC to the RAN. It acts as a mobility anchor as the terminals move between eNBs and as a mobility anchor for other 3GPP technologies.
- Packet Data Gateway (P-GW): connects the EPC to the internet via the SGi interface. It is responsible for the allocation of IP address for a specific terminal and serves as a mobility anchor for non-3GPP RAT.
- Mobility Management Entity (MME): manages the signaling and control plane between the User Equipments (UE) and the EPC. It is also in charge of sending profiles information and data of the UEs to the HSS. MME selects the appropriate S-GW and P-GW that are used for data transfer of UEs established connections.
- Policy and Charging Rules Function (PCRF): provides a dynamic control of policy and charging on a per subscriber and per IP flow basis.

The E-UTRAN is the RAN of LTE. It contains the radio functions of LTE-A systems such as radio-resource management, scheduling, transmission and retransmission protocols, coding, S1

link between the eNBs and MME/S-GW, the X2 link between the eNBs. The X2 link is used to support active mode mobility, interference mitigation and radio resource management between neighboring cells.

3.3 E-UTRAN

3.3.1 Physical Layer

In our studies, we consider only downlink transmissions and the scheme used in LTE downlink is the Orthogonal Frequency-Division Multiple Access (OFDMA). In OFDMA, the original bandwidth is subdivided into multiple subcarriers. Each of these subcarriers can then be individually modulated. Typically in OFDMA systems we can have hundreds of subcarriers with a spacing between them (15KHz on the LTE case). Different numbers of subcarriers can be assigned to different users, thereby providing the flexibility to support differentiated QoS. Besides, by transmitting over multiple subcarriers in parallel, OFDMA has the ability to cope with severe channel conditions, such as interference and frequency selective fading due to multipath. Also, parallel transmission reduces the symbol rate over each subcarrier, permitting the use of cyclic-prefix, which provides a guard interval to eliminate intersymbol interference from the previous symbol and allows channel estimation and equalization. OFDMA brings hence robustness and flexibility for the E-UTRAN. The chain to generate an OFDM signal starts by parallelizing the symbols that need to be transmitted, after they are modulated (in LTE the modulation can be QPSK, 16AQM, 64QAM). Then they are used as input bands for an Inverse Fast Fourier Transform (IFFT) operation. This operation produces OFDM symbols, which will be transmitted. Notice that a conversion from the frequency to the time domain was made when the IFFT was used. On the receiving side of the OFDMA system we should expect a Fast Fourier Transform (FFT) operation that will then convert the symbol to the frequency domain again.

Nevertheless, it is not possible to use OFDMA on the uplink since it presents a high Peak-to-Average Power Ratio (PAPR). Single Carrier FDMA (SC-FDMA) is a single carrier multiplexing, and has the benefits of having a lower PAPR. On SC-FDMA before applying the IFFT the symbols are pre-coded by an extended Discrete Fourier Transform (DFT). This way, each subcarrier after the IFFT will contain part of each symbol, spreading hence the symbol over all the subcarriers, resulting in smaller variations in the instantaneous power of the transmitted signal. Similarly to transmission, Inverse DFT (IDFT) is used at the reception after the FFT operation.

In LTE, a subframe is transmitted over 1ms. Each subframe is divided into two slots known as Physical Resource Blocks (PRB). A PRB aggregates several Resource Elements (RE), each of them is one 15KHz subcarrier by one OFDM symbol. A PRB has in its turn dimensions of twelve sub carriers in the frequency domain and of six or seven symbols in the time domain. Figure 3.2 represents the structure of a PRB in the time and frequency domains.

The scheduler is a controller that is used by the eNBs to decide which UEs should be given resources to send or received data at each iteration. In LTE-A, scheduling is done per subframe, that is per 1ms. Scheduling can be as simple as round robbin, or it can be more complicated and takes into consideration certain factors, for example the radio conditions of the UEs, the amount of data in the buffers, QoS requirements, etc. In this work we use a proportional fair scheduler, whose objective is to balance between two competing interests: maximizing the network throughput while at the same time allowing all users at least a minimal level of service. One way of translating this fairness is by serving the UE that has the highest ratio between the instantaneous throughput (estimated through Channel Quality Information (CQI)) and the total amount of data transmitted to the UE during the current established connection.

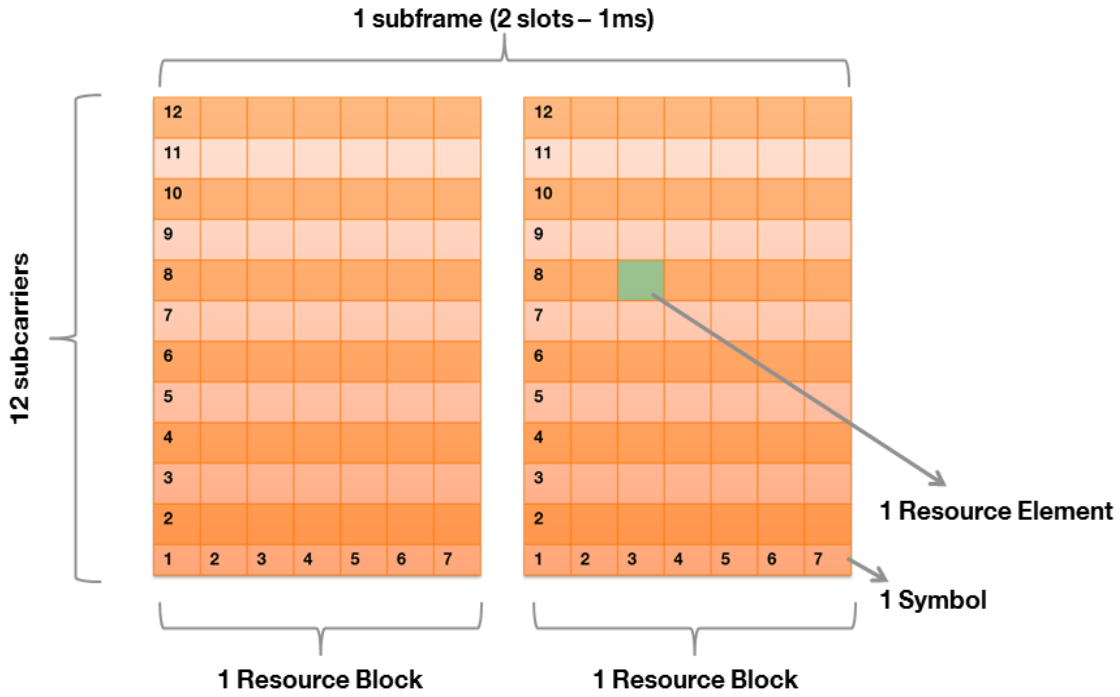


Figure 3.2: LTE-A Resource Blocks

3.3.2 Mobility

In LTE-A, the UE is connected to one cell at a time. During the handover process, it breaks the connection with the source cell before connecting to the target cell. LTE-A HOs are hence hard HOs, unlike soft HOs in 3G where the UE can be connected simultaneously to more than one cell. On the other hand, LTE-A HOs can be classified as intra-site HOs, which refer to the case where the source and target cells reside in the same site or inter-site HOs where the source and target cells are located in two different sites. Furthermore, HOs may be classified according to target RAT (2G,3G,LTE, etc) or target frequency carrier of the same RAT. In this thesis we consider intra-LTE-A, intra-frequency HOs only, and both intra-site and inter-site HOs. In our simulator, we consider that UEs arrive to the network according to a Space Poisson Point Process [66], with different speeds and movement directions. Their location is updated every second. We consider a File Transfer Protocol (FTP)-like traffic, i.e. each UE which arrives in the network aims to download a file (of fixed size). When a UE arrives in the network and wants to download its file, it first has to attach to a cell. Let RxP_n^k be the received signal power of UE k from cell n. The UE k connects to the cell i according to the following equation:

$$i = argmax_n(RxP_n^k + CIO_n) \tag{3.1}$$

where CIO_n is the Cell Individual Offset (CIO) of cell n [67]. This means that, if we do not consider CIO, the UEs connect to the cell which had the best channel conditions. If the *Connection Establishment* is successful then the UE goes into the *Receive Data* procedure, otherwise it is dropped (figure 3.3).

If the serving cell is no longer offering the best radio channel condition then the UE searches for a new target cell which offers the best channel conditions. This procedure is called Connection ReConfiguration or HO. More precisely, a UE k attached to cell s starts a Time To Trigger (TTT)

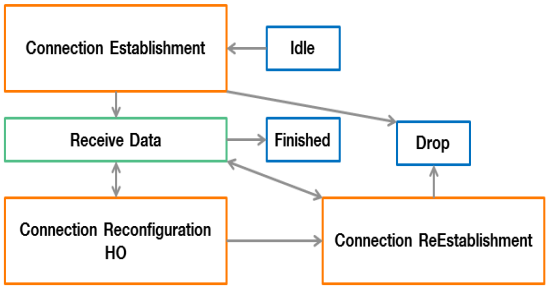


Figure 3.3: UE procedures

countdown for a HO to a target cell $t \neq s$ if:

$$t = argmax_n(RxP_n^k + CIO_n + HYS_s.\mathbb{I}_{n=s}) \tag{3.2}$$

where HYS_s is the HO Hysterisis parameter of cell s.

During its data download and even during a HO procedure, a UE may loose connection with its serving cell. We call this a Radio Link Failure (RLF). If such an event occurs then the UE goes into the *Connection ReEstablishment* procedure where the UE attempts to connect to a cell using similar steps as in the *Connection Establishment*. If the RLF occurs while in the *Connection ReEstablishment* procedure then the UE is dropped.

3.3.3 Interference and Almost Blank Subframes

Let RxP_i^k be the power received per PRB by UE k from cell i. H is the interference matrix between the cells. $H(i,j) = 1$ if cells i and j are transmitting on the same frequency band, and is zero otherwise. Since we are using the LTE-A OFDMA downlink transmission scheme, a UE k attached to cell j is hence interfered by the downlink transmission of cells $i \neq j$. A UE k connected to cell j and using one PRB, receives an interference signal equals to:

$$I_j^k = \sum_{i \neq j} H(i,j).L_i.RxP_i^k \tag{3.3}$$

where L_i is the probability that cell i is transmitting on the same PRB as cell j. L_i can be evaluated by the following expression:

$$L_i = \frac{C_{o,i}}{C_{T,i}} \tag{3.4}$$

Where $C_{o,i}$ and $C_{T,i}$ are the occupied and the total number of PRBs in a scheduling time interval. In this thesis, the load of a cell i is calculated by averaging the value L_i over several intervals. The downlink Signal to Interference plus Noise Ratio (SINR) for UE k attached to cell j over a single PRB has the following expression:

$$SINR_j^k = \frac{RxP_j^k}{(I_j^k + N_{th})} \tag{3.5}$$

where N_{th} is the thermal noise per PRB. At each scheduling time interval, the scheduler of cell j assign a number N_k of PRB for each UE k attached to j. The number of bits transmitted per resource block is determined through the mapping of the SINR per PRB and the modulation coding

scheme, using link level curves. The throughput of a UE is hence the sum of the throughput over each of the N_k PRB used for transmission.

In order to reduce the interference towards the neighboring cells some cells might be forbidden to transmit at certain time instances. To better understand this we present in figure 3.4 the structure of an LTE-A downlink frame. A frame is composed of ten sub-frames and has thus a length of 10 ms. In our simulator, the resources of each sub-frame are allocated to only one user. We assume that all the cells are synchronized on a frame basis. A sub-frame where no user data is transmitted is called an Almost Blank Sub-frame (ABS). Although certain control signals are still transmitted in an ABS at reduced power, we neglect their effect on the interference in our work. The ABS helps reduce the interference towards neighboring cells. The technique is typically used in HetNets, where sometimes the range of the small cells is extended. So, to protect the UEs at the small cell edge from high level of interference from the macro cell, we use ABSs transmissions on the macro cells. For instance, the eICIC SON function automatically tunes the ABSs transmissions of macro cells (see section 3.4 for details).

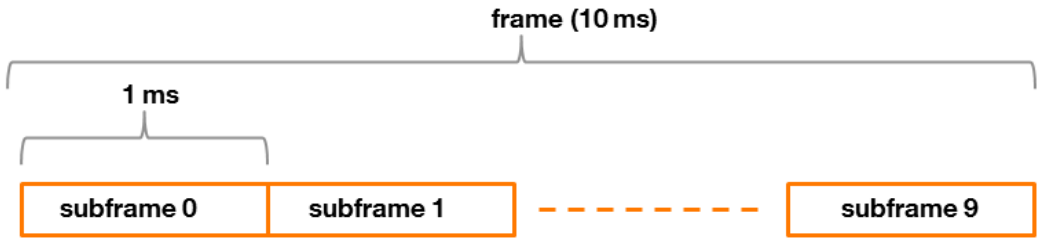


Figure 3.4: LTE-A Frame Structure

3.4 SON Functions Description and Algorithms

As stated in Chapter 2, SON functions are categorized into self-configuration, self-optimization and self-healing functions. In this thesis we focus on self-optimization SON functions. Although 3GPP specifies the input, output and targets of the SON functions, the algorithm inside the SON are not standardized, and they are specific to each vendor. For the use cases we consider in this work, we are particularity interested in the following SON functions:

Mobility Load Balancing The objective of this function is to balance the load between the cells, in order to avoid over loaded cells, hence reducing the blocking rate and increasing the robustness of the network to traffic variations. Load balancing can be done in different manners. Traffic can be offloaded to a different frequency band (inter-frequency), or to a different RAT (inter-RAT) or to neighboring cells via antenna tilt adaptation, power adaptation or CIO, which is an artificial offset used when the UE selects its serving cell. A significant amount of studies on MLB exists in the literature [32, 33, 68–72]. We consider one independent MLB instance per macro cell. At each

time iteration, the MLB algorithm updates the CIO as follows:

$$\Phi_i = \begin{cases} min(\Phi_{max,i}, \Phi_i + \Delta_i) & \text{if } l_i < l_{L,i} \\ max(\Phi_{min,i}, \Phi_j - \Delta_i) & \text{if } l_i > l_{H,i} \\ \Phi_i & \text{otherwise} \end{cases}$$

where l_i, Φ_i, $\Phi_{max,i}$, $\Phi_{min,i}$, Δ_i, $l_{H,i}$ and $l_{L,i}$ are respectively the load, CIO value, max and min values the CIO can take, tuning step, upper and lower load threshold of marco cell i. Note that the UE connects to the cell from which it receives the highest power and CIO.

Cell Range Expansion This function is designed to balance the load in HetNets. More specifically between a macro cell and the small cells in its coverage area. The offloading from the macro to the small cell is done by expanding the range of the small cell through its CIO. A significant amount of studies on CRE exists in the literature [34, 35, 73–77]. In our work, we consider that each small cell is deployed to serve a traffic hotspot inside a coverage region of a macro cell, and that a CRE runs on each small cell. At each iteration, the algorithm updates the CIO of the small cell where it is deployed in order to off-load (force UEs to HO to neighboring cells) the macro cells onto small cells when they are over-loaded, i.e. have a high resource utilization. The mechanism also deals with the case where the small cells are (or we anticipate they will become) over-loaded. At each iteration, the CRE algorithm tunes the CIO according to the following:

$$\Phi_j = \begin{cases} min(\Phi_{max,j}, \Phi_j + \Delta_j) & \text{if } l_i > l_{H,j} \text{ and } l_j < l_{L,j} \\ max(\Phi_{min,j}, \Phi_j - \Delta_j) & \text{if } l_j > l_{H,j} \\ \Phi_j & \text{otherwise} \end{cases}$$

Where j is the small cell and i is the its associated macro cell.

Enhanced Inter Cell Interference Coordination This function is also HetNet specific. It manages the ABS (sub-frames with only control signals transmitted at low power as explained in 3.2.3) transmission of the macro cell in order to protect small cell edge users from macro downlink interference. A significant amount of studies on eICIC exists in the literature [36, 37, 78–84]. In our work, we consider that we may have at most 36 ABS sub-frames out of 40 sub-frames, i.e. sub-frames 0 and 5 of every frame will never be ABS. The ABS pattern is random and we reset it every frame (40 sub-frames). The protected UEs, i.e. the ones which were forced to HO from the macro cell to the small cells, will take advantage of the ABS sub-frames due to the proportional fair scheduler that we employ, as the instantaneous channel conditions should be significantly better during the ABS. Only the Macro cell will change the number of ABS sub-frames. Given some time instance, the eICIC algorithm updates the number of ABS of the macro cell according to the following:

$$\kappa_i = \begin{cases} min(\kappa_{max,i}, \kappa_i + \Delta_i) & \text{if } R_i > R_{H,i} \\ max(0, \kappa_i - 1) & \text{if } R_i < R_{L,i} \\ \kappa_i & \text{otherwise} \end{cases}$$

where R_i, κ_i, $\kappa_{max,i}$, Δ_i, $R_{H,i}$ and $R_{L,i}$ are respectively the ratio of the throughput of the UEs that are attached to macro cell i and the throughput of the protected UEs, the number of transmitted ABS, max number of transmitted ABS, tuning step, upper and lower ratio threshold of marco cell i.

Energy Saving This function focuses on sleep mode optimization to improve the energy efficiency of radio networks. The general concept aims to dynamically match the capacity offered by operators to the actual users demand at given times and locations. In other words, the objective of energy saving functions is to minimize the energy consumption in the network while maintaining the required levels of QoS, by turning off or putting certain network components in sleep mode, such as base band units, radio heads, amplifiers, etc. For further insight, please refer to [38, 39, 85–92]. In this work, we focus on sleep mode procedures for small cells. There exists in the literature different strategies for small cells sleep mode. Small cells can be augmented with a low-power sniffer capability, that enables the detection of active UEs in the coverage area of the small cell. The small cell can hence enter sleep mode (turn off pilot transmission and signal processing) when no close UEs are detected. Another approach would be to control the sleep mode of the small cells through the UEs, which broadcast wake up signals to wake up the small cells within its range. A third approach consists in a core network controlled sleep mode. The state of the small cell (being in sleep or awake mode) is controlled via control messages through the backhaul. In our work, we do not consider a SON function for sleep mode. Instead, we consider that the wake up decision is taken by the C-PBSM, depending on the network context observed by learning agent and on the objective of the operator. The decision of putting the small cell in sleep mode or not is then transmitted to the small cell at each RL iteration

3.5 LTA-A System Level Simulator

This paragraph gives an overview on the LTE-A system level simulator [93]. All the previously mentioned elements are put together with the SON management framework described in the previous chapter, in an LTE-A simulator built using C++. In the framework of this thesis, we have developed and implemented the previously mentioned SON functions, as well as the C-PBSM related modules and we have integrated them in the simulator. In the following we focus on the channel model of the simulator and the C-PBSM and SON functions processes.

3.5.1 Channel Model

The channel models used in the simulator are based and benchmarked with the 3GPP TS 36.814 [94] and the ITU-R M2134 [95] with extensions to HetNets (specifically Pico Cells). Modeling the radio channel comes down to estimating power received by the UEs from the network cells. Let $\overline{RxP}_n^k(t)$ be the power received by UE k from cell n at time t expressed in dBs. We model it as follows:

$$\overline{RxP}_n^k(t) = TxP_n(t) - FL_n + AG_n + AP_n^k(t) - PL_n^k(t) + Sh_n^k(t) + Ff_n^k(t) - PtL_n \quad (3.6)$$

where $TxP_n(t)$ is the transmission power, FL_n is the feeder loss, AG_n is the antenna gain, $AP_n^k(t)$ is the antenna pattern, $PG_n^k(t)$ is the path loss, $Sh_n^k(t)$ is the shadowing effect, $Ff_n^k(t)$ is the fast fading effect, and PtL_n is the penetration loss. We define the filtered signal through layer 3 filtering as follows:

$$RxP_n^k(t) = FtC.\overline{RxP}_n^k(t) + (1 - FtC).\overline{RxP}_n^k(t-1) \quad (3.7)$$

where FtC is the filter coefficient [96].

The antenna gain includes the gain and the horizontal and vertical directivity of the antenna. Macro cells use directive antennas and the small cells use omni-directional antennas.

The shadowing is established per UE with respect to every site. If the UEs are moving, we update the shadowing (with respect to all cells) at every time instance. Each update is correlated with the previous. The correlation is implemented for the shadowing of UE k with respect to site n as follows:

First let the shadowing correlation coefficient be:

$$corrSh_n^k(t) = exp(-\frac{|d_{u_k(t),u_k(t-1)}|}{corrDistSh_n^k(t)}) \tag{3.8}$$

where $u_k(t) \in \mathbb{R}^2$ is the location of UE k at time t, d_{u_1,u_2} is the distance between u_1 and u_2 ($u_1, u_2 \in \mathbb{R}^2$) and $corrDistSh_n^k(t)$ is the correlation distance established for site n and UE k. The shadowing correlations are then performed as follows:

$$Sh_n^k(t) = \sqrt{(corrSh_n^k(t))^2}Sh_n^k(t-1) + \sqrt{1-(corrSh_n^k(t))^2}Y_n^k(t) \tag{3.9}$$

where $Sh_n^k(t)$ is the shadowing at time t experienced by UE k with respect to site n in dB, $Y_n^k(t)$ is a random variable (normal distribution) with zero mean and standard deviation equals to the shadowing standardized standard deviation.

Note that the detailed numerical values of the shadowing correlation distance ($corrDistSh_n^k(t)$), standard deviation as well as values and calculations of feeder loss, antenna gain and pattern, path loss, fast fading (based on Rayleigh distribution) and penetration loss can be found in [94, 95].

Note that this channel model is used in this work when simulating hexagonal heterogeneous networks. In the case where we use real network topology (chapter 4), we use real-like network parameters and accurate ray tracing based propagation, extracted from Orange's network. We then add to these measurements the theoretical and standardized shadowing variations (described previously) to dynamically simulate the network. More details about the simulator parameters and topology can be found in [93].

3.5.2 SON Management Framework

We implement the SON functions and the SON management layer i.e. the (C-)PBSM and we integrate them in the LTE-A system level simulator. The SON management simulation process is detailed through a block diagram in figure 3.5.

The SON mechanisms make use of a guard window where the SON functions do not collect measurements, for instance immediately after a network parameter has been changed as shown in figure 3.6. The C-PBSM as well uses a guard window on a bigger scale. It does not collect data immediately after changing the SCV sets of the deployed SON functions. In fact, the C-PBSM waits for the SON functions to adapt with the new SCV sets and converge to a stationary state. The C-PBSM guard window hence consists of several SON iterations (figure 3.6). Note that, as stated in the previous chapter, throughout the thesis we consider that all SON functions in the network converge to a stationary state after few SON iterations.

In all our simulations we have considered the SON functions and the C-PBSM to be synchronous. They run their algorithm at the end of a time window of size T (a multiple of the time granularity). All the SON functions as well as the C-PBSM can have access to measurements and network parameters.

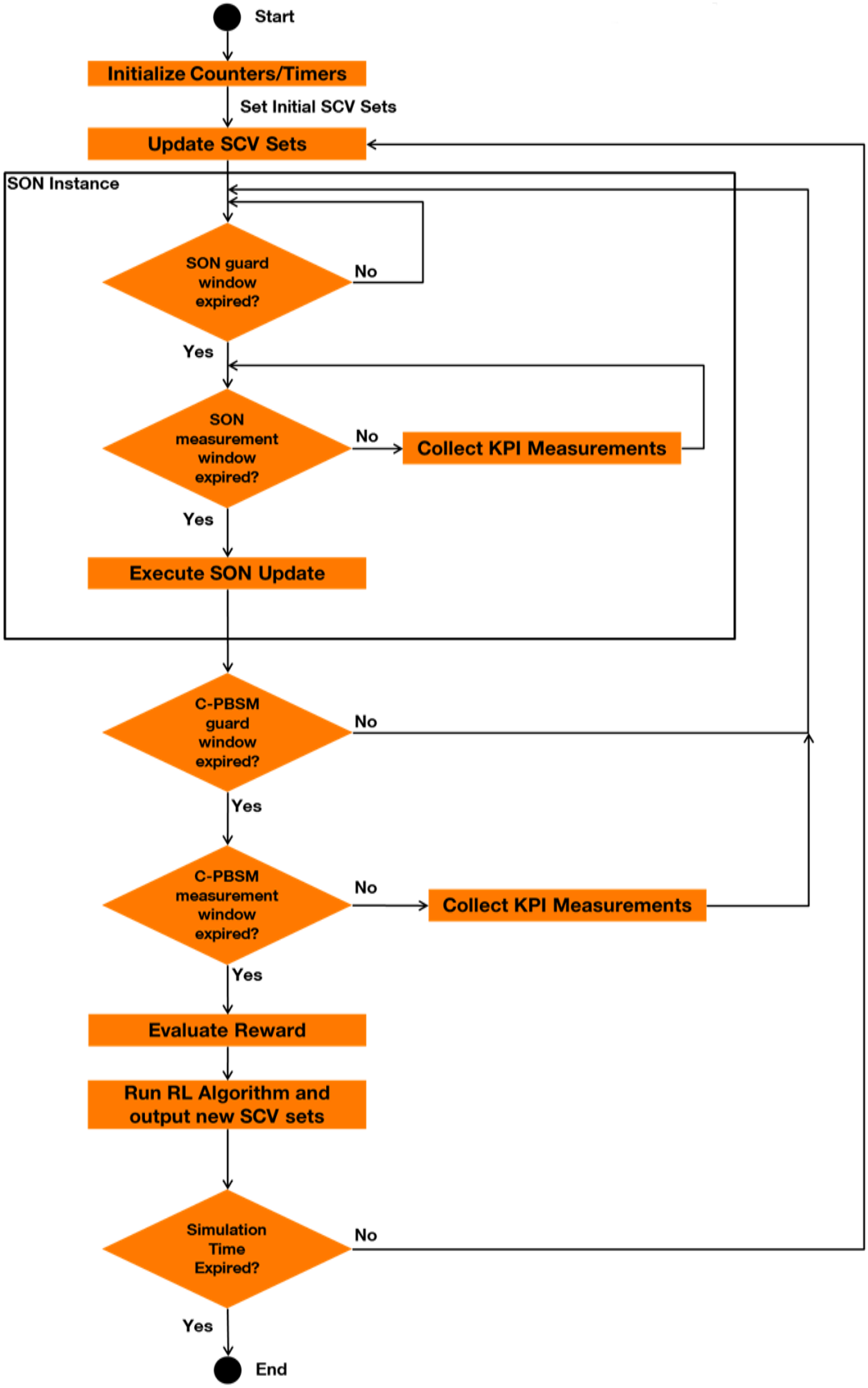


Figure 3.5: C-PBSM and SON Functions Simulations Block Diagram

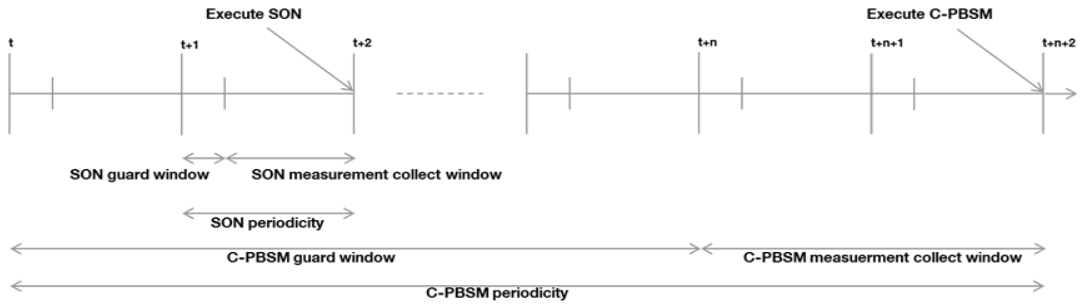


Figure 3.6: C-PBSM and SON Functions Timers

3.5.3 Simulator Block Diagram

The simulator's general block diagram is depicted in figure 3.7. At each time step: i) the UEs move, ii) their radio conditions are updated, iii) the communication procedure gets updated, iv) the UEs (in the Receive Data procedure) get scheduled and receive data accordingly, v) users that finished the download (or cannot reconnect to a cell) are deleted, vi) new users are generated in a random manner and vii) the SON and the C-PBSM instances collect measurements, check their timers and take action according to their respective timers, that is, the C-PBSM configures the SON functions instances and the SON functions instances configure their corresponding network parameters. Note that in this work, the simulator's granularity is of the order of 1 second.

The measurements are for example: the HO event counter, the RLFs counter, the load of the cells, the counter for number of UEs in the network, the cell/UEs throughout, etc. We reset them at the beginning of each time step. They are updated throughout each time step. The measurements serve as inputs to the network optimization functions, i.e. the SON functions or the C-PBSM as explained in figure 3.5.

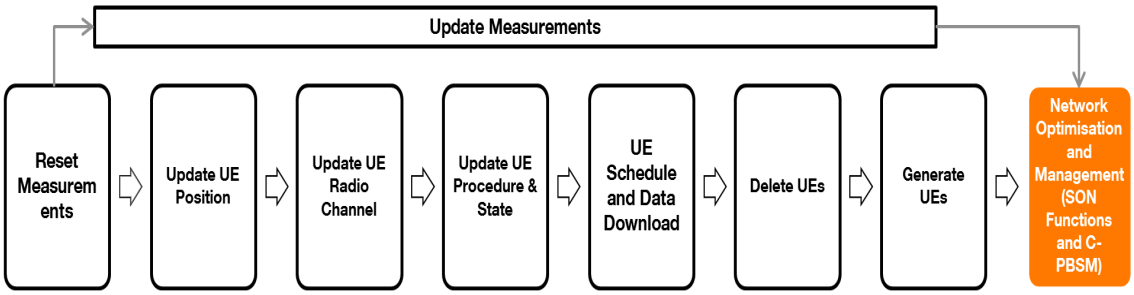


Figure 3.7: LTE-A Simulator Block Diagram

Chapter 4

Multi Armed Bandit for Cognitive Policy Based SON Management

4.1 Introduction

In this chapter we present a MAB framework for C-PBSM. We consider a distributed SON approach, where several instances of each SON function are deployed in the network. Recall that the SON functions are considered as black boxes for the operator, i.e. the algorithm inside is unknown to the operator. We consider a network section where several SON functions are deployed in multiple instances. A learning agent is placed on top of the SON functions, as described in chapter 2 figure 2.4, which objective is to learn the best SCV set combination, that satisfies to the best the operator objectives.

In section 4.2 we introduce a C-PBSM based on a centralized stochastic MAB algorithm. The C-PBSM sequentially explores the possible SCV sets combinations and learns the optimal possible combination that satisfies to the best the operator objectives. The proposed C-PBSM is able to efficiently adapt its strategies with objective changes of the operator. We assess the performances of the proposed approach on an LTE-A system level simulator. Section 4.3 introduces a bandit based approach with improved convergence rate. This approach exploits better the structure of the actions by considering a linear model of the rewards with respect to the actions features.

4.2 C-PBSM Based on Stochastic MAB

As stated previously, MAB is a RL problem, formulated as a sequential decision problem, where at each iteration an agent is confronted with a set of actions called arms, each when pulled, generates a reward. The agent is only aware of the reward of the arm after pulling it. The objective is to find a strategy that permits to identify the optimal action, while maximizing the cumulated rewards obtained during the process. The learning agent is hence faced with the exploration/exploitation dilemma.

In this chapter, we model the cognitive SON management as a sequential learning problem. The optimal policy is the SCV set combination of the deployed SON instances of different SON functions, that insures that certain KPI targets specified by the operator, are optimized.

4.2.1 Problem Statement

We consider a network section where the traffic is stationary and unequally distributed. The cognitive capability of the C-PBSM is realized through a learning agent that is placed on top of the SON functions as shown in figure 2.4 of chapter 2. This agent takes actions by configuring all the instances of the different SON functions in the network. The set C of possible actions is defined as follows: Let S be the set of SON functions deployed in the network. Let N_s be the number of deployed instances of a SON function s and V_s be the set of possible SCV sets for SON s, $\forall s \in S$ (i.e. assuming that all the instances of the same SON function have the same set of possible SCV sets). Then we define the action set as:

$$C = (V_{s_1})^{N_{s_1}} \times (V_{s_2})^{N_{s_2}} \times ... \times (V_{s_{|S|}})^{N_{s_{|S|}}} \tag{4.1}$$

where $s_1, s_2, ..., s_{|S|} \in S$.

For each action $c \in C$, the C-PBSM receives a reward r, evaluated based on a combination of network KPIs, once all the SON functions instances have converged. In this work we consider that all the SON functions instances converge after a sufficient time. The reward function is hence the following:

$$r_{t,c_t} = \sum_i \omega_i . K_{i,t,c_t} \tag{4.2}$$

where K_{i,t,c_t} is the value of the i^{th} KPI when action c_t is taken, at iteration t, and ω_i are weights set by the operator, they are positive and add up to one. They reflect the operator's priority to optimize the corresponding KPI. Whenever $\omega_i > 0$, then the KPI K_i is a target KPI for the operator, with optimization priority ω_i. Contrariwise, $\omega_i = 0$ means that K_i is not considered as target KPI. The agent is aware of the reward of an action only after applying it to the network for a sufficient time: at each iteration t, the agent picks $c_t \in C$ and then evaluates the corresponding reward r_{t,c_t}. For each applied action $c \in C$, under the conditions of traffic stationarity and SON convergence, the observed KPIs converge towards the same distribution, independently of the previous actions. It is then reasonable to consider that, for the same configuration combination c, the observations of r are i.i.d. random variables following an unknown probability distribution. The best action (the one that satisfies best the operator's objective targets) is defined as the action that has the highest expected reward:

$$c^* = \underset{c \in C}{\operatorname{argmax}}(\mathbb{E}(r_c)) \tag{4.3}$$

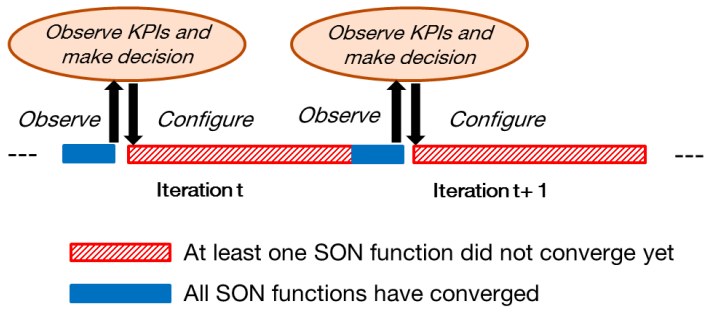


Figure 4.1: C-PBSM Sequential Learning Process

The agent's task is to sequentially explore the set of actions C in the real network in order to find c^*. The question is: what should be the agent's exploration strategy so that it reaches as fast as possible its objective (finding c^*), while minimizing suboptimal decisions (i.e. minimizing the iterations where the agents choses $c \neq c^*$). This is also known as the exploration/exploitation dilemma. The C-PBSM's sequential learning process is represented in figure 4.1.

4.2.2 Stochastic Multi-Armed Bandit

We propose to use a stochastic MAB, which is a RL framework where the rewards of each arm are supposed to be an i.i.d sequence following an unknown distribution, specific to each arm. Hereafter we consider C (as described in equation 4.2.1) to be the set of arms and $|C|$ the number of arms. ν_c and μ_c are respectively the reward's unknown probability distribution and the unknown average reward of arm c. The MAB learning process is the following:
For $t = 0, 1, 2, ...$:

- The agent selects an arm $c_t \in C$ according to some policy
- The environment outputs a vector of rewards $\mathbf{r}^t = (r_{t,1}, r_{t,2}, ..., r_{t,|C|}) \in [0,1]^{|C|}$
- The agent observes only r_{t,c_t}

The MAB's objective is to define a strategy that finds the optimal arm, while minimizing the pseudo-regret defined as follows:

$$\overline{R}_n = \max_{c \in C} \mathbb{E}[\sum_{t=0}^{n} r_{t,c} - \sum_{t=0}^{n} r_{t,c_t}] \tag{4.4}$$

The expectation is taken with respect to the draw of arms and rewards. Let $\mu^* = \max(\mu_c)$ for all $c \in C$. Then the pseudo-regret can be written as:

$$\overline{R}_n = n.\mu^* - \sum_{t=0}^{n} \mathbb{E}[\mu_{c_t}] \tag{4.5}$$

The UCB1 algorithm that we propose to use in this work is based on the Upper Confidence Bound (UCB) strategy, and was introduced first in [49]. The algorithm does not need any prior knowledge on the rewards distributions. Each arm is associated with an index composed of two terms: the first is the empirical average of the perceived rewards and the second is related to the upper confidence bound derived from the Chernoff-Hoeffding bounds. The UCB1 algorithm is based on the principle of Optimism in Face of Uncertainty (OFU), which is applied in many decision making problems in uncertain environment. Basically the idea of OFU consists in picking the most optimal action according to the acquired data (knowledge) at a certain iteration. That is, despite the agent's lack of knowledge in what actions are best, it will construct an optimistic guess as to how good the expected payoff of each action is, and pick the action with the highest guess. If the guess is wrong, then the optimistic guess will quickly decrease and the agent will be compelled to switch to a different action. But if the action's choice was well picked, then the agent would be exploiting that action and incur little regret. This is how OFU balances exploration and exploitation. In the UCB1 algorithm, the optimistic guess is constructed through the two terms index (the empirical average and the upper confidence bound) associated with each arm. The algorithm's pseudo-code is presented in the following:

Algorithm 3 UCB1 Algorithm

$\alpha > 0$ input constant parameter
$\widehat{\mu}_{0,c} = 0$ for all $c \in C$
$N_{0,c} = 0$ for all $c \in C$
for each arm $c \in C$
- successively select arm c
- observe reward r_c
- evaluate $\widehat{\mu}_{0,c} = r_c$
- $N_{0,c} = 1$
end for
for each iteration $t = 1, 2, 3...$
- select arm c_t that maximizes $\widehat{\mu}_{t-1,c_t} + \sqrt{\frac{\alpha.log(t)}{2.N_{t-1,c_t}}}$
- observe reward r_{t,c_t}
- evaluate $\widehat{\mu}_{t,c_t} = \frac{(t-1).(\widehat{\mu}_{t-1,c_t})+r_{t,c_t}}{t}$
- $N_{t,c_t} = N_{t-1,c_t} + 1$
end for
where $\widehat{\mu}_{t,c_t}$ is the empirical average evaluated at iteration t
and N_{t,c_t} the number of times arm c_t was pulled at iteration t

The UCB1 has theoretical guarantees on the pseudo-regret: in [50] the authors prove the following bound on the pseudo-regret for the UCB1 after n iterations, for $\alpha > 2$:

$$\overline{R}_n \leq \sum_{c:\Delta_c>0} (\frac{2\alpha}{\Delta_c} log(n) + \frac{\alpha}{\alpha - 2}) \tag{4.6}$$

$\Delta_c = \mu^* - \mu_c$ is the sub optimality parameter of arm c, and α is an input parameter. The authors in [50] also show a matching lower bound in case the rewards of the arms follow a Bernoulli distribution. Furthermore, a more general result in [49, 50] shows that the upper bound of the pseudo-regret of UCB1 matches, up to a logarithmic factor, with the lower bound achievable by any other algorithm. In fact, it was shown that a distribution independent lower bound on the pseudo-regret for K arms bandit problem is of order $\Omega(\sqrt{Kn})$, whereas the upper bound of the UCB1 pseudo-regret is of order $O(\sqrt{Knlog(n)})$. These results make the UCB1 a reasonable solution for the stochastic MAB.

4.2.3 Scenario Description

We consider a 2 layers heterogeneous LTE-A network. A set S of decentralized SON functions is deployed:

- **MLB:** Deployed on each macro cell. Recall that the objective of MLB is to balance the load between macro cells by iteratively comparing the observed cell load to predefined upper and lower load thresholds and tuning the CIO parameter accordingly.
- **CRE:** Deployed on each small cell. Recall that the CRE optimizes the load difference between the small cell and the macro cell where it is deployed, by tuning CIOs using an iterative process similar to MLB.
- **eICIC:** Deployed on each macro cell containing small cells in it coverage region. Recall that its objective is to protect small cell edge users (attached to a small cell because of

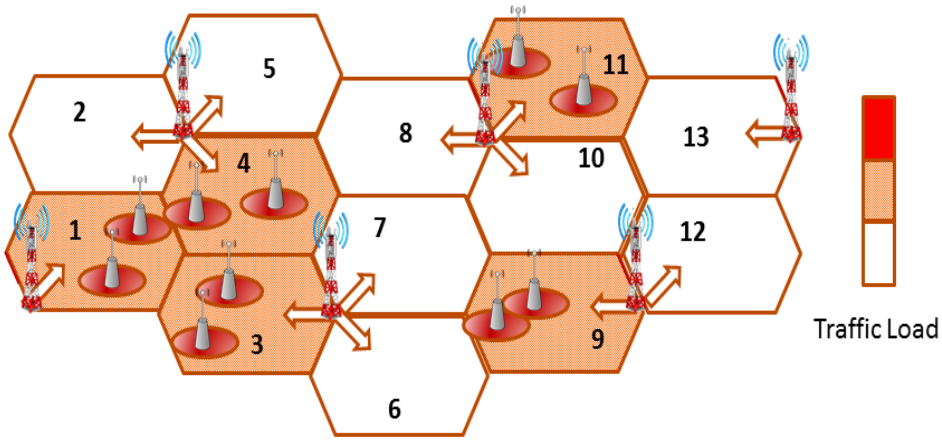


Figure 4.2: Network Model

CRE's offset), from the high levels of interference caused by the closest macro cell (due to the power difference between the small and the macro cells) by tunning the number of ABS in the macro cell frames.

These functions affect directly two KPIs: the load variance and the user throughput. A well balanced network is in fact more stable towards traffic variations and has lower call block rates that are caused mostly by overloaded cells. It is consequently more robust to serve an eventual increasing traffic demand. However, under the effect of the CIO, traffic will offload from one cell to another, meaning that users will not attach necessarily to the best serving cell in term of received power. Hence, in order to maximize the network's average throughput, there is a trade-off to be found for these users, between having good radio conditions but low resources (if they are attached to the best serving cell and that this cell is overloaded) and having more resources but with degraded radio channel conditions (when they are offloaded to a less loaded cell). On the other hand, the eICIC's objective is to improve the throughput of small cell edge users by requesting ABS from the corresponding macro cell. While this procedure improves the throughput of cell edge users, it affects the resources of the macro cell which may cause a load increase, thus increasing the load variance in the network. It is clear that these functions influence each another when deployed simultaneously in a network.

To test the UCB1, we consider a section of a two layer heterogeneous LTE-A network as shown in figure 4.2. The traffic distribution is unbalanced and stationary (high traffic in cells $\{1, 3, 4, 9, 11\}$). Additional traffic hot-spots are deployed in the network, they are each served by a small cell. The main simulation parameters are summarized in table 4.1. We consider the following SON deployment: an MLB on each macro cell, an eICIC on each macro cell where small cells are deployed and a CRE on each small cell. Each of these functions can be configured with SCV sets presented in table 4.2 (SON functions are detailed in chapter 3). Soft configuration means a configuration that reduces the load difference to some extent without reaching high levels of CIO and by aggressive we designate a configuration that seeks to equilibrate the load as much as possible by configuring higher levels of CIO. Recall that in the case of MLB and CRE, $\Phi_{max,i}$, $\Phi_{min,i}$, Δ_i, $l_{H,i}$ and $l_{L,i}$ are respectively the max and min values the CIO can take, tuning step, upper and lower load threshold of marco cell or small cell i. In the case of eICIC, $\kappa_{max,i}$, Δ_i, $R_{H,i}$ and $R_{L,i}$ are respectively the max number of transmitted ABSF, tuning step, upper and lower throughput ratio threshold between the UEs attached to marco cell i and the protected UEs in the

cell edge of the small cells deployed in the coverage area of macro cell i.

The total number of possible actions is: $2^{N_{s_1}} \times 2^{N_{s_2}} \times 3^{N_{s_3}}$, where N_{s_1}, N_{s_2} and N_{s_3} are respectively the number of eICIC, CRE and MLB instances in the network. In our case, $N_{s_1} = 5$, $N_{s_2} = 10$ and $N_{s_3} = 13$. There are approximatively 5×10^{10} possible actions. It is impossible to consider such a big set of actions, especially that the regret of UCB1 scales linearly with the number of actions as stated in the previous section. In order to reduce the complexity of the algorithm, we configure SON instances per groups of similar cells instead of per instance, hence reducing considerably the number of possible actions of the C-PBSM in the network [97]. The motivation behind this reasoning is that, when deployed in similar network contexts, the SON instances of a SON function will behave similarly to each other if configured with the same SCV set. Cell classes are hence defined according to their network context. In our scenario, we consider 2 cell classes: A class of macro cells with high traffic and where a network layer of small cells is deployed ($C1 = \{1, 3, 4, 9, 11\}$) and a class of single layer macro cells with low traffic ($C2 = \{2, 5, 6, 7, 8, 10, 12, 13\}$). This leaves us with 12 actions for $C1$ and 3 for $C2$, hence a total of 36 possible SCV sets combinations, which is a reasonable number of actions to apply UCB1.

Parameters	**Settings**
Bandwidth	10 MHz
PRBs per eNB	50
Carrier Frequency	2 GHz
Macro ISD	1732 m
Macro Path Loss to UE	$128.1 + 37.6 \times log_{10}(d[Km])$
Pico Path Loss to UE	$140.1 + 37.6 \times log_{10}(d[Km])$
Antenna Gain	macro: 14 dBi; pico: 5 dBi
Transmit Power	macro: 46 dBm; pico: 30 dBm
Shadowing Standard Deviation	macro: 8 dBm; pico 10 dBm
Traffic model	Dowlink File Transfer Protocol (FTP), file size: 14 Mbits
RL Simulation Time Iteration	20 min (in simulation time)

Table 4.1: Simulation parameters

We define the following KPIs, that are considered to be the most relevant for this scenario: For each iteration t and action c_t:

- l_{i,t,c_t} is the load of cell i
- $\bar{l}_{t,c_t}$ is the average load in the considered section
- $\sigma_{t,c_t} = \frac{\sum_{i=0}^{|N|}(l_{i,t,c_t} - \bar{l}_{t,c_t})^2}{|N|}$ the load variance in the considered section. $|N|$ being the number of cells in this section.
- $\overline{T}_{t,c_t}$ is the average user throughput in the section
- $\overline{T'}_{t,c_t}$ is the average small cell edge user throughput in the considered section (average throughput of the protected UEs).

The variables were normalized between 0 and 1. The reward function reflecting the operator's objective is:

$$r_{t,c_t} = \omega_1(1 - \sigma_{t,c_t}) + \omega_2\overline{T}_{t,c_t} + \omega_3\overline{T'}_{t,c_t} \tag{4.7}$$

ω_1, ω_2 and ω_3 are weights set by the operator depending on its priorities.

SON	SCV sets
MLB	Off: Function is turned off SCV1: Soft configuration ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 6dB; l_{L,i} = 0.55; l_{H,i} = 0.85; \Delta_i = 1dB$) SCV2: Aggressive configuration ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.725; l_{H,i} = 0.775; \Delta_i = 2dB$)
CRE	SCV1: Soft configuration ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 8dB; l_{L,i} = 0.6; l_{H,i} = 0.85; \Delta_i = 2dB$) SCV2: Aggressive configuration ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.8; l_{H,i} = 0.8; \Delta_i = 2dB$)
eICIC	Off: Function is turned off SCV1: Function is turned on ($\kappa_{\max,i} = 8ABS; R_{L,i} = 1; R_{H,i} = 2.5; \Delta_i = 1ABS$)

Table 4.2: SCV Sets Description

Furthermore, since r is a linear combination of weights and KPIs, we consider that the agent preserves, besides the empirical average of the perceived reward, an empirical average of each of the considered KPIs, for each action c. Hence, we assume that these KPIs are always reported and stored, even if not involved in the reward function. This knowledge of these KPIs allows the algorithm to adapt faster to the operator's priority changes. In fact, on the one hand the upper confidence bound of each arm depends only on the number of trials and the number of total iterations of the algorithm. On the other hand, the estimated reward of each arm is a linear combination of the KPI estimates. Therefore, having an estimate of the average value of each KPI for each action permits the algorithm to adapt quickly with the objective changes of the operator. The knowledge acquired by the algorithm for a certain operator objective can hence be exploited for different objectives. We call this modified version of UCB1 "Adaptive UCB1" (Algorithm 4).

4.2.4 Simulation Results

The C-PBSM is tested using an LTE-A system level simulator based on the 3GPP specifications [94] as described in chapter 3. As a baseline for comparison, we consider the following default configuration of the SON functions: all deployed functions are on and with parameters that are manually set after observing the values of the KPIs involved in the reward function (table 4.3).

SON	SCV sets
MLB	$\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.6; l_{H,i} = 0.8; \Delta_i = 2dB$
CRE	$\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.6; l_{H,i} = 0.8; \Delta_i = 2dB$
eICIC	$\kappa_{\max,i} = 8ABS; R_{L,i} = 1; R_{H,i} = 2.5; \Delta_i = 1ABS$

Table 4.3: Default SCV Sets Description

Algorithm 4 Adaptive UCB1 Algorithm

$\alpha > 0$ input constant parameter
$\widehat{\mu}_{0,c} = 0$ for all $c \in C$
$\widehat{\mu}_{1,0,c} = 0$ for all $c \in C$
$\widehat{\mu}_{2,0,c} = 0$ for all $c \in C$
$\widehat{\mu}_{3,0,c} = 0$ for all $c \in C$
$N_{0,c} = 0$ for all $c \in C$
for each arm $c \in C$
- successively select arm c
- observe r_c, σ_c, $\overline{T}_c$, $\overline{T'}_c$
- evaluate $\widehat{\mu}_{1,0,c} = 1 - \sigma_c$
- evaluate $\widehat{\mu}_{2,0,c} = \overline{T}_c$
- evaluate $\widehat{\mu}_{3,0,c} = \overline{T'}_c$
- $N_{0,c} = 1$
end for
for each iteration $t = 1, 2, 3...$
- Get operator priority weight ω_1, ω_2, ω_3
- select arm c_t that maximizes:
$\omega_1\widehat{\mu}_{1,t-1,c_t} + \omega_1\widehat{\mu}_{2,t-1,c_t} + \omega_1\widehat{\mu}_{3,t-1,c_t} + \sqrt{\frac{\alpha log(t)}{2N_{t-1,c_t}}}$
- observe r_{t,c_t}, σ_{t,c_t}, $\overline{T}_{t,c_t}$, $\overline{T'}_{t,c_t}$
- evaluate $\widehat{\mu}_{1,t,c_t} = \frac{(t-1)(\widehat{\mu}_{1,t-1,c_t})+(1-\sigma_{t,c_t})}{t}$
- evaluate $\widehat{\mu}_{2,t,c_t} = \frac{(t-1)(\widehat{\mu}_{2,t-1,c_t})+\overline{T}_{t,c_t}}{t}$
- evaluate $\widehat{\mu}_{3,t,c_t} = \frac{(t-1)(\widehat{\mu}_{3,t-1,c_t})+\overline{T'}_{t,c_t}}{t}$
- $N_{t,c_t} = N_{t-1,c_t} + 1$
end for
where $\widehat{\mu}_{t,c_t}$ is the empirical average evaluated at iteration t
and N_{t,c_t} the number of times arm c_t was pulled at iteration t

In this scenario, we consider three consecutive phases where the operator sets a first set of priorities, then changes twice these priorities in the second and third phase as shown in figure 4.3. In the first iterations the algorithm is exploring the possible configurations, since it is learning from scratch with no prior information, hence generating low rewards. After a number of iterations, the algorithm converges towards a configuration, generating stationary rewards. Furthermore, the algorithm adapts rapidly with objective changes: in the second phase the algorithm passes through a brief phase of exploration, before converging back towards a new configuration. Same for the last phase, the algorithm explores briefly then converges to another configuration. This quick adaptation is possible by keeping an estimate of the mean value of each KPI for each of the actions as explained previously (Adaptive UCB1 Algorithm). Note that the different values of the rewards vary with the operator objectives. These value changes are the consequences of the normalization of different KPIs with different distributions, ranges and behaviors, and do not reflect the optimality of the policy.

In figure 4.4 we analyze the optimality of the C-PBSM. We compare the average reward generated by the configuration identified as best by the UCB1 with the default configuration on the one hand and the optimal configuration on the defined action space, identified through an exhaustive search on the other hand. We notice that in the three cases the UCB1 succeeds in identifying the

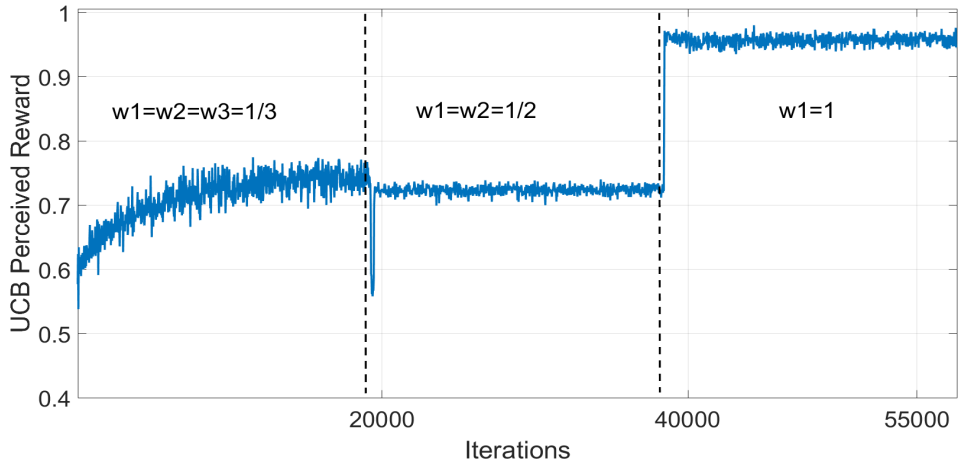


Figure 4.3: Evolution of UCB1 Perceived Rewards for Different Operator Objectives

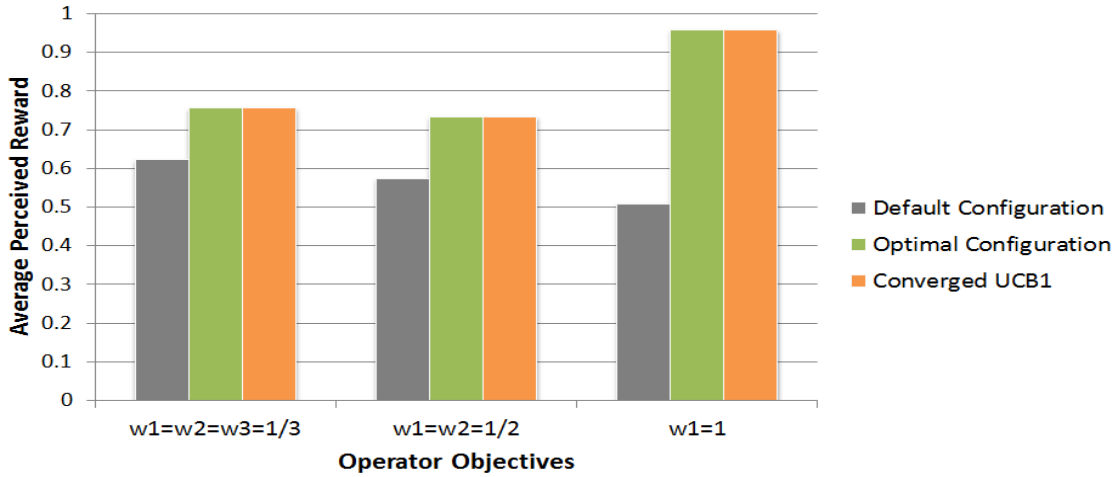
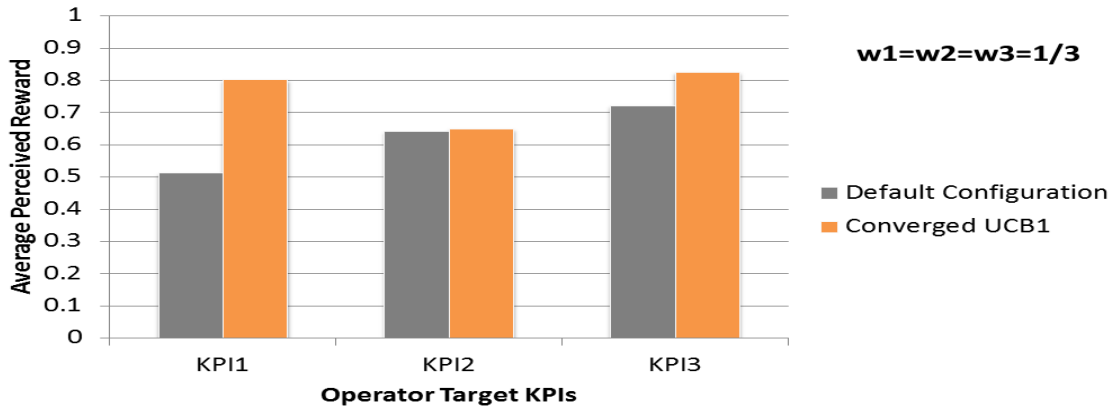


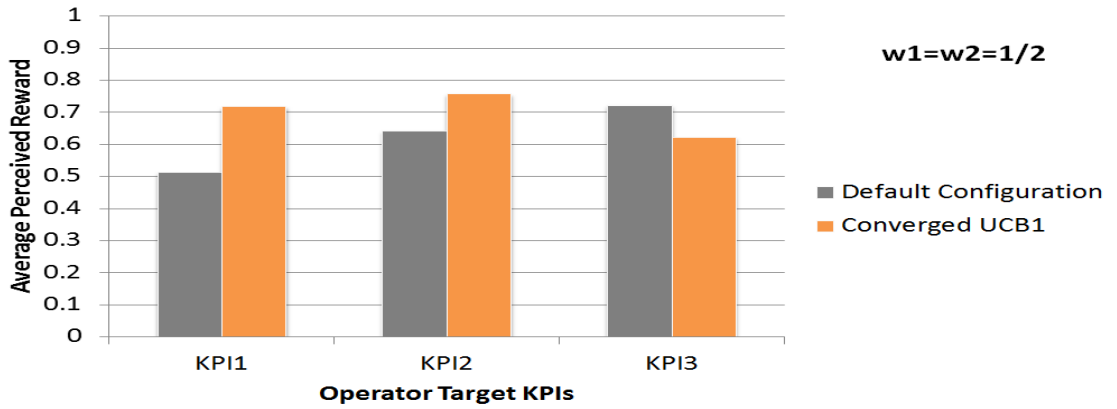
Figure 4.4: Average Final Rewards for Different Operator Objectives

best configuration, whereas the default configuration is always generating sub-optimal rewards. We can say that after a learning phase at the start, the C-PBSM converges towards SON configurations that maximize the perceived reward, performing better than a static default configuration. Note that exhaustive search is a rather naive and straightforward strategy that permits indeed to find an optimal action. It consists in successively testing each action for a sufficient period of time, in order to have a reliable estimate of its average reward. Therefore, it cannot be practically used in real networks because it does not consider regret minimization. It will hence generate high values of regret, meaning that the network would spend more time in suboptimal configurations.

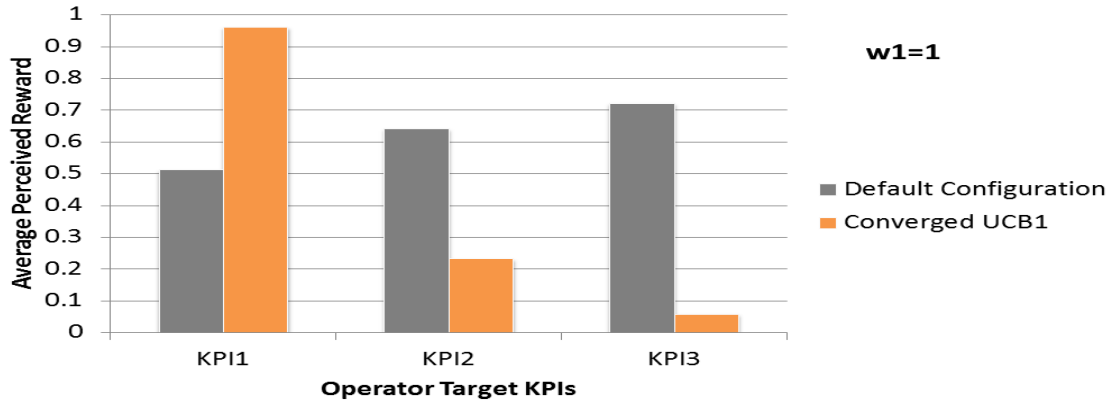
Figure 4.5 compares the performance of the optimal configuration with the default configuration in terms of normalized KPIs where KPI1, KPI2 and KPI3 are respectively the load variance, average user throughput and average small cell edge user throughput. A KPI is considered in the operator's objective by setting its corresponding priority weight to non zero. The results show that the C-PBSM successfully translates and adapts the network to the operator's objective. Indeed, it optimizes the KPIs when considered in the objective function, at the expense of degrading the



(a) w1=w2=w3=1/3



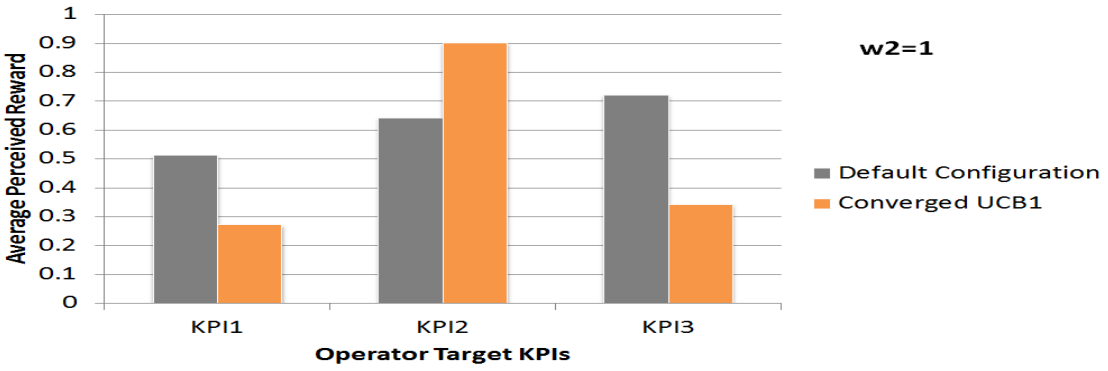
(b) w1=w2=1/2



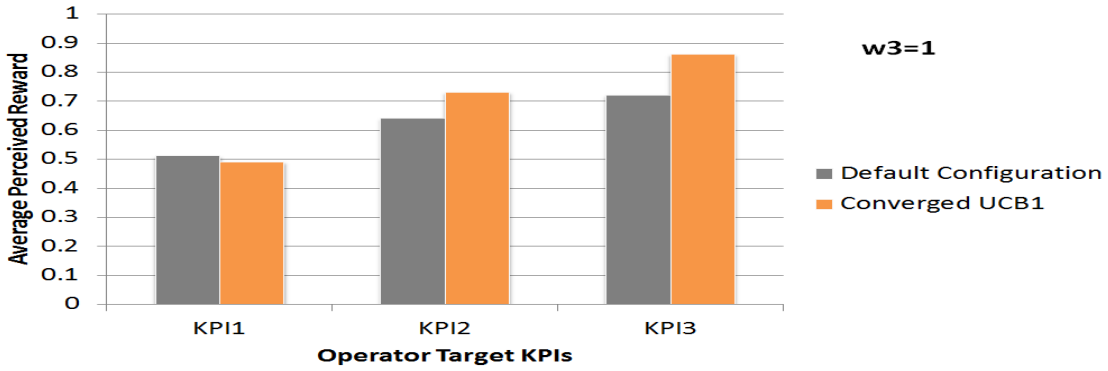
(c) w1=1

Figure 4.5: Average Perceived KPIs (KPI1: load variance, KPI2: average user throughput, KPI3: average cell edge user throughput)

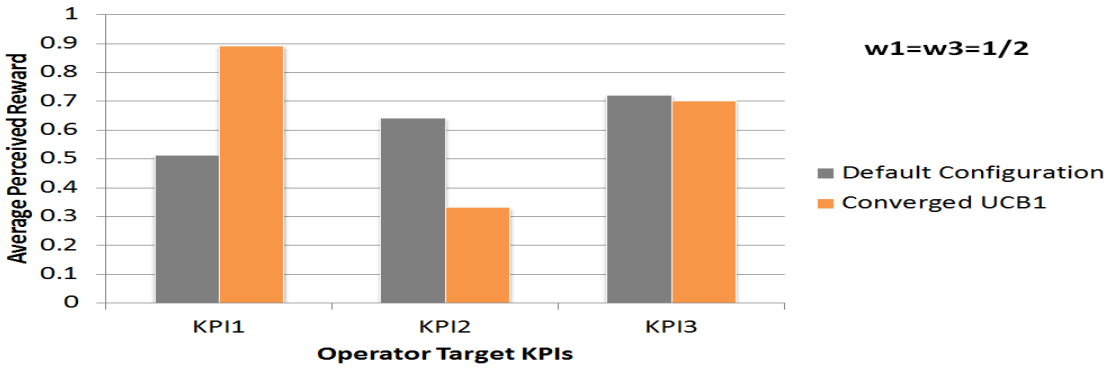
other KPIs (the ones not considered as objective). For instance when all three KPIs are considered as target KPIs, i.e. $\omega_1 = \omega_2 = \omega_3 = 1/3$ in figure 4.5-a, the C-PBSM finds the SCV sets combination that insures the best trade-off between the three KPIs, and hence all three are optimized. In the second phase, the C-PBSM is asked to optimize only the load variance and the average user throughput. We can see in figure 4.5-b that now the trade-off is between KPI1 and KPI2. KPI3, that is not considered as target, is not optimized. In the last phase, the operator wants to optimize only the load variance, that is KPI1. In this case a considerable gain is achieved for KPI1, at the expense of KPI2 and KPI3 degradation (figure 4.5-c). The KPIs analysis is similar for other operator objectives. A sample is depicted in figure 4.6.



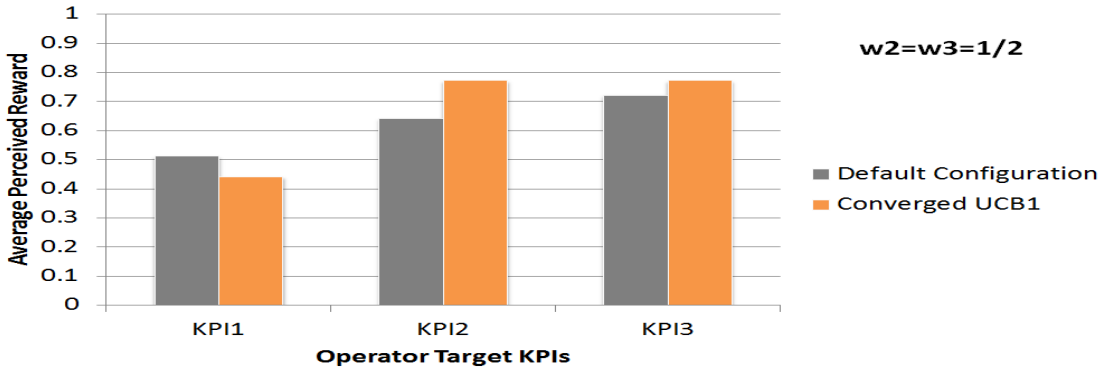
(a) w2=1



(b) w3=1



(c) w1=w3=1/2



(d) w2=w3=1/2

Figure 4.6: Average Perceived KPIs (KPI1: load variance, KPI2: average user throughput, KPI3: average cell edge user throughput)

4.3 C-PBSM Based on Linear MAB

In the previous section, we have shown that the MAB, in particular the UCB1, can be efficiently used to learn the optimal SON configuration policy from the network through a sequential exploration/exploitation of the SCV sets combinations of the deployed SON functions in a certain network section [98]. Nevertheless, in practice, an operator would like to spend the least time possible learning in the network. Because in the learning phase, the algorithm is still exploring, and therefore taking suboptimal decisions. The convergence rate of such an approach is consequently of the highest importance and should be optimized as much as possible. This said, enhancing the convergence speed should not be at the expense of the reliability and robustness of the algorithm's estimations. Poor estimations lead to suboptimal decisions and a convergence to a suboptimal final policy.

UCB1 is a good solution when the environment is stochastic and the arms are independent. However, the regret bound of UCB1 scales linearly with the number of arms (from equation 4.6, the pseudo regret is of order $O((|C|log(t))/\Delta)$ where $|C|$ is the number of arms and Δ is the difference between the average reward of the best arm and the second best arms). This linear dependency slows the convergence rate when the number of arms grows, and is hence a limitation for the UCB1 when the set of arms is large. Faster strategies can be used, if a particular structure or relation between the arms can be identified or reasonably assumed. In the literature, several techniques are used to exploit the dependency between the arms. For example in [99] there is a set of super arms, each containing a subset of arms. At each iteration when a super arm is played, the environment reveals the outcome of all the arms belonging to the corresponding subset of arms. In [100] the authors consider dependencies between clusters of arms in the form of a generative model.

In our case, each action can be described by a vector of categorical features, each representing the applied SCV set on each of the SON instances of the different SON functions deployed in a certain section. We consider the expected reward function to be linear with respect to the action features. There are many works in the literature that studied linear models for MAB, the most important of them were presented in chapter 2. For this work, we use the LinUCB algorithm that was first proposed and studied in [62]. LinUCB assumes that the expected reward is a linear function with respect to arms features. Besides having guarantees on the regret (as will be presented in the next section), LinUCB has some practical advantages. In fact, it is rather straightforward to implement. It also uses a ridge regression, which enjoys more stability and computational efficiency, whereas other algorithms of the same linear framework, such as LinRel [51] require more computational complexity such as solving SVD (Singular Value Decompositions) or eigen decompositions. Also, LinUCB has already been experimentally tested in practical cases and showed its efficiency in real life and practical problems such as in [101, 102]. In the following we introduce the algorithm and analyze its performance on a C-PBSM case study.

4.3.1 Linear UCB for C-PBSM

We consider the same problem formulation as before. The set of actions is

$$C = (V_{s_1})^{N_{s_1}} \times (V_{s_2})^{N_{s_2}} \times ... \times (V_{s_{|S|}})^{N_{s_{|S|}}}$$

where $s_1, s_2, ..., s_{|S|} \in S$ and we consider a reward functions as defined in equation 4.2. The same hypotheses as before still hold that is:

- The traffic in the network is stationary and unequally distributed

- $\forall c \in C$, all the SON functions instances converge after a sufficient time

At each iteration t, the agent picks an action c_t, described by a feature vector $\mathbf{x}_{c_t} \in \mathbb{R}^d$. In our case, the actions are described by categorical variables representing the SCV sets. We encode these variables as binary vectors, that we normalize to the unit [103]. The observed rewards r_{t,c_t} are independent random variables, for which the expectation is a linear combination of features:

$$\mathbb{E}[r_{t,c_t}|\mathbf{x}_{c_t}] = \mathbf{x}_{c_t}^T \theta^* \tag{4.8}$$

Where θ^* is an unknown parameter vector to be estimated. The definition of the best action and the regret are still the same as in equation 4.3 and 4.4, since we are still in the stochastic framework. Consider K to be the number of arms (in our case $K = |C|$), α an input parameter that tunes the upper confidence bound and d the dimension of the action's feature vector. The LinUCB pseudo-code is presented in table the following:

Algorithm 5 LinUCB Algorithm

1:Inputs $\alpha > 0$, $K, d \in \mathbb{N}$
2: $\mathbf{A} \leftarrow \mathbf{I}_d$
3: $\mathbf{b} \leftarrow \mathbf{0}_d$
4: for $t = 0, ..., T$
5: - $\theta_t \leftarrow \mathbf{A}^{-1}\mathbf{b}$
6: - Observe K features $\mathbf{x}_{c^{(1)}}, \mathbf{x}_{c^{(2)}}, ..., \mathbf{x}_{c^{(K)}} \in \mathbb{R}^d$
7: - $\forall c \in C$ do
8: $\quad p_{t,c} \leftarrow \theta_t^T \mathbf{x}_c + \alpha\sqrt{\mathbf{x}_c^T \mathbf{A}^{-1}\mathbf{x}_c}$
10: - Choose action $c_t = \underset{c \in C}{\operatorname{argmax}}(p_{t,c})$
11: - Observe reward $r_{t,c_t} \in [0, 1]$
12: - $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{x}_{c_t}\mathbf{x}_{c_t}^T$
13: - $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{x}_{c_t} r_{t,c_t}$
14: end for

The main idea behind the algorithm is to estimate the unknown parameter vector θ^* by applying a regression on training data at iteration t. The training data is in this case the previously tested feature vectors (arms) until iteration t. I_d is a d by d identity matrix and $\mathbf{0}_d$ is a column vector with 0 values. The ridge regression is performed on line 5 of the algorithm to estimate the parameter vector θ. In line 8, $p_{t,a}$ is composed of 2 expressions, the first is an estimate of the expected reward and the second is an upper confidence bound. The detailed analysis of LinUCB can be found in [62] where the authors show that the algorithm achieves a regret bound of $O(\sqrt{Td\ln^3(KT\ln(T)/\delta)})$ with probability $1 - \delta$, where T is the number of iterations. They also prove a matching lower bound for this framework $\Omega(\sqrt{Td})$ (Ω denotes an asymptotic lower bound). This analysis shows that the regret bound scales logarithmically with the number of arms K. We should hence expect that the LinUCB based C-PBSM will have faster convergence rates than the UCB1 based C-PBSM.

4.3.2 Scenario Description

We consider a heterogeneous RAN as represented in figure 4.7. The macro cellular layer corresponds to a real Orange network deployment in central Paris, with real-like network parameters

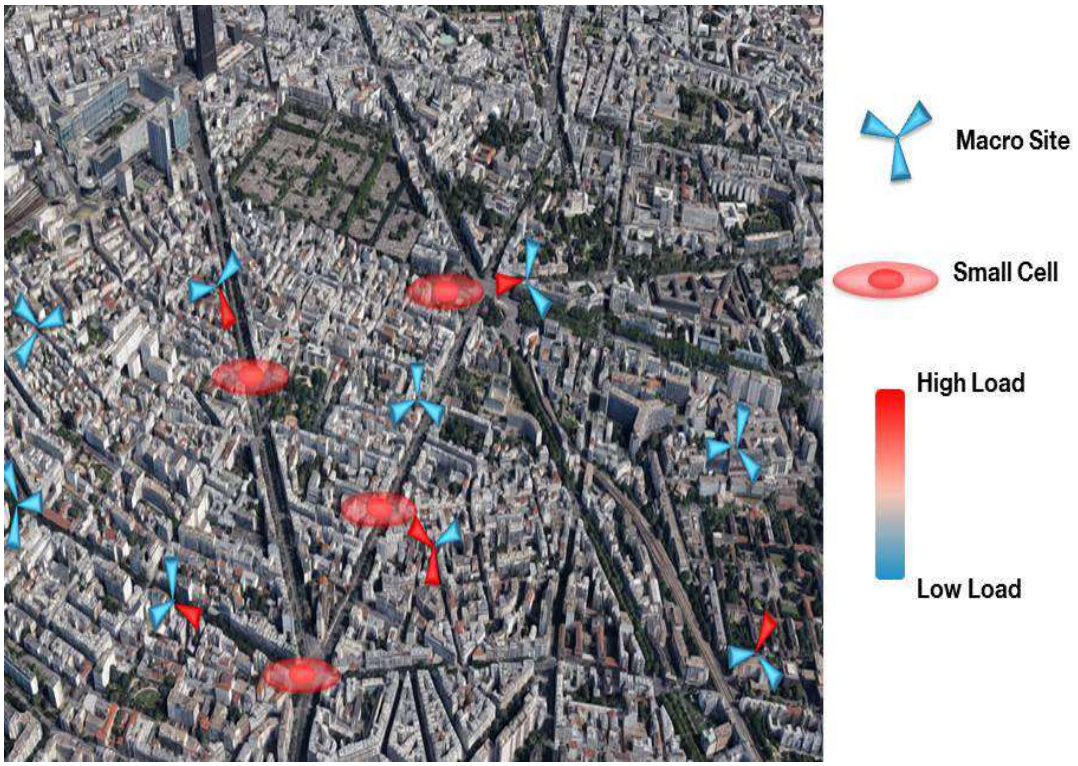


Figure 4.7: Network Section

and accurate ray tracing based propagation. The small cell layer is added using standard 3GPP propagation specifications [94]. We consider an unbalanced and stationary traffic distribution. This results in some macro cells being highly loaded and others with low load. Additional traffic hotspots are served by the small cells. We only consider downlink traffic.

The reason we simulate a real network topology is to test the linear model assumption in a realistic RAN environment, rather than the commonly used heterogeneous hexagonal model. For the sake of clarity, let's first assume that all the instances of the same SON function are always configured with the same SCV set. Consequently, assuming a linear model between the action features and the expected perceived reward means practically that the expected KPIs of an SCV set of a SON function is the same regardless of the SCV sets applied to the other SON functions. In other words, the considered linear model neglects the interaction and variations that could occur in the behavior of a SON function (configured with a certain SCV set), when changing the configuration of other SON functions. This can be justified by the fact that this interaction can be compensated with the SON function's own dynamics (through its internal control loop). That is, the SON function is able to adapt with the SCV sets changes of other neighboring SON functions, through its internal dynamics, hence keeping the same behavior for the same SCV set configuration. However, this linear model still needs to be handled with attention. Whence our choice to challenge its limits and test its validity over a realistic and irregular network such as the one in figure 4.7.

As in the previous section, we consider a scenario with three distributed SON functions, that are MLB, CRE and eICIC, each having several instances deployed in the network. As a reward function we use the definition in 4.7. We also keep the same SCV sets as before (table 4.2), the same action space of 36 arms and 2 classes of cells: that is a class $C1$ of macro cells with a layer of small cells, where MLB, CRE and eICIC function instances are deployed and configured

similarly, and a class of single layer macro cells $C2$ where only MLB instances are deployed and configured similarly. Consequently, the feature vector will have a dimension $d = 10$ (figure 4.8). Each feature is binary, indicating which SCV set is activated for each SON function of each cell

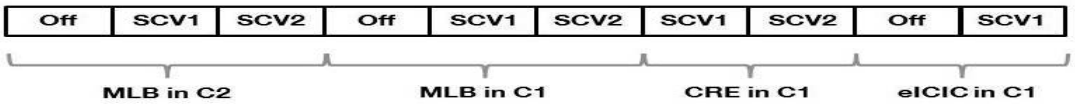
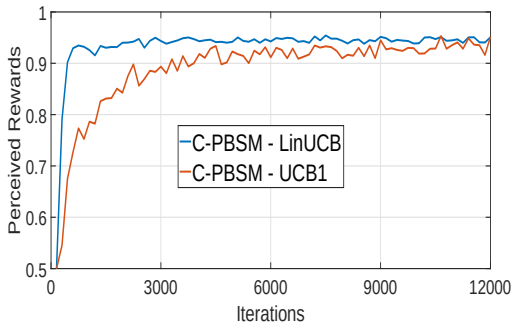


Figure 4.8: Action Features Vector

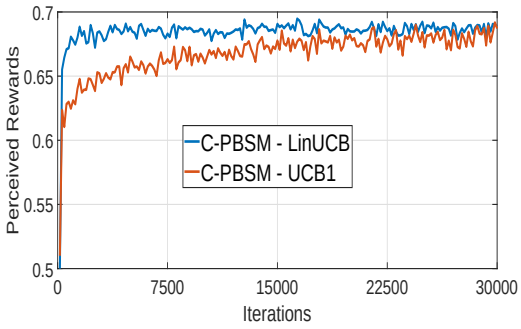
class. For example if we consider the action c where MLB instances in $C2$ are Off, MLB instances in $C1$ are configured with SCV1, CRE instances in $C1$ are configured with SCV1 and eICIC in $C1$ are off, then the feature vector would be: $\mathbf{x}_c = (1, 0, 0, 0, 1, 0, 1, 0, 1, 0)$.

4.3.3 Simulation Results

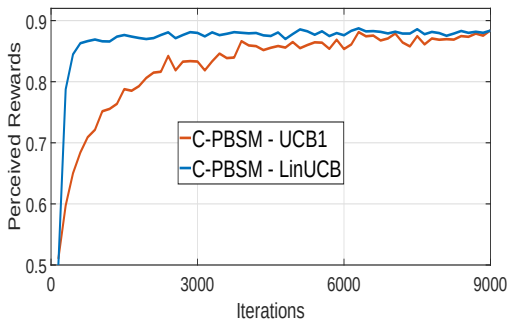
In the previously described scenario, we test the baseline C-PBSM based on UCB1 algorithm and the C-PBSM proposed in this section, based on LinUCB. We run each C-PBSM for different operator objectives. The results are presented in figure 4.9.



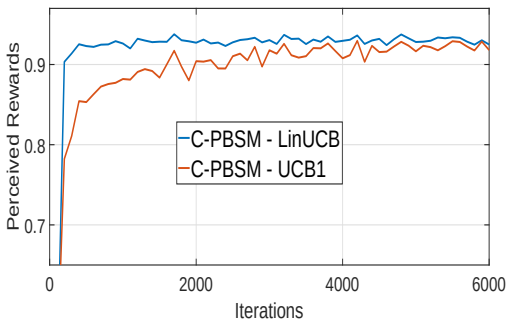
(a) w1=1



(b) w1=w2=w3=1/3



(c) w1=w3=1/2



(d) w2=1

Figure 4.9: Perceived Rewards Comparison

They show that both algorithms converge and lead to the same performances in terms of generated rewards and hence KPI performances. However, the LinUCB shows a much faster convergence than the UCB1 based approach. This fast convergence is due to the linearity assumption in

the LinUCB. As stated previously, unlike the UCB1 that assumes independent arms, the LinUCB assumes a linear relation between the arms as shown in equation 4.8. In other words, this means that by testing a certain SCV combination in the network, the C-PBSM based on LinUCB is able to estimate the outcome of other SCV combinations without testing them, i.e. at each iteration, the LinUCB tries a single action but deduces and updates the estimate of several actions according to the linear model. Whence the faster convergence.

A modification of the LinUCB permits fast adaptability with the operator objectives as shown in Algorithm 6. Instead of keeping a single response vector for the reward function $\mathbf{b}$, the algorithm's variation keeps a response vector for each KPI, which are three in our case, hence $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$. The response vector for the reward, $\mathbf{b}$, is then computed according to the priority weights of the operator (equation 4.7). Figure 4.10 shows the rapid adaptation of the algorithm with the objective changes of the operator.

Algorithm 6 Adaptive LinUCB Algorithm

1:Inputs $\alpha > 0$, $K, d \in \mathbb{N}$
2: $\mathbf{A} \leftarrow \mathbf{I}_d$
3: $\mathbf{b} \leftarrow \mathbf{0}_d$
4: $\mathbf{b_1} \leftarrow \mathbf{0}_d$
5: $\mathbf{b_2} \leftarrow \mathbf{0}_d$
6: $\mathbf{b_3} \leftarrow \mathbf{0}_d$
7: for t=0,...,T
8: - Get operator priority weights $\omega_1, \omega_2, \omega_3$
9: - $\theta_t \leftarrow \mathbf{A}^{-1}\mathbf{b}$
10: - Observe K features $\mathbf{x}_{c^{(1)}}, \mathbf{x}_{c^{(2)}}, ..., \mathbf{x}_{c^{(K)}} \in \mathbb{R}^d$
11: - $\forall c \in C$ do
12: $\quad p_{t,c} \leftarrow \theta_t^T \mathbf{x}_c + \alpha\sqrt{\mathbf{x}_c^T \mathbf{A}^{-1}\mathbf{x}_c}$
13: - Choose action $c_t = \underset{c \in C}{\operatorname{argmax}}(p_{t,c})$
14: - Observe reward $r_{t,c_t} \in [0, 1]$
15: - $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{x}_{c_t}\mathbf{x}_{c_t}^T$
16: - $\mathbf{b_1} \leftarrow \mathbf{b_1} + \mathbf{x}_{c_t}(1 - \sigma_{t,c_t})$
17: - $\mathbf{b_2} \leftarrow \mathbf{b_2} + \mathbf{x}_{c_t}\overline{T}_{t,c_t}$
18: - $\mathbf{b_3} \leftarrow \mathbf{b_3} + \mathbf{x}_{c_t}\overline{T'}_{t,c_t}$
19: - $\mathbf{b} = \omega_1\mathbf{b_1} + \omega_2\mathbf{b_2} + \omega_3\mathbf{b_3}$
20: end for

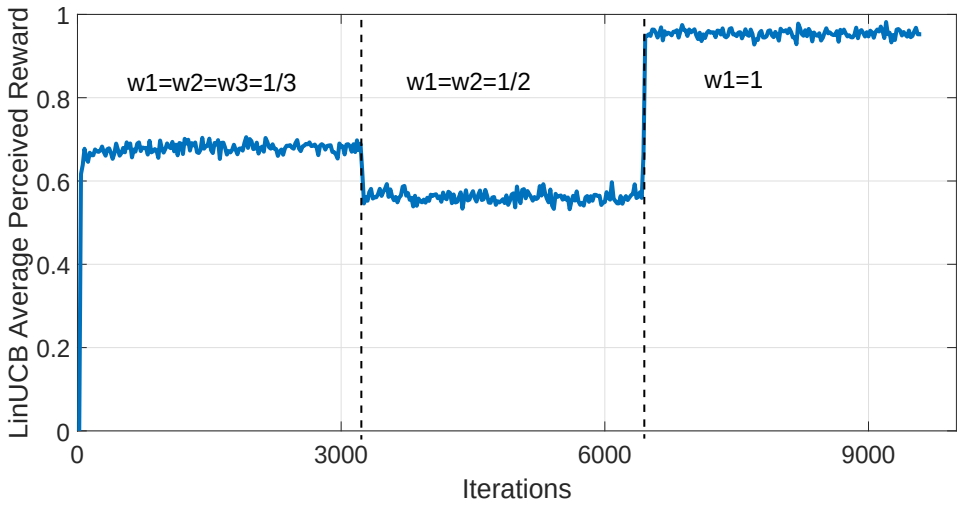


Figure 4.10: Evolution of LinUCB Perceived Rewards for Different Operator Objectives

Even though the LinUCB improves drastically the convergence of the C-PBSM (simulations show that few iterations suffice for convergence) as stated previously, the linearity hypothesis should be handled with caution when generalizing the approach to other scenarios. Mainly because in certain cases, the deployed SON functions and instances may by strongly correlated with each other in the sense that changing the SCV set of a SON instance might alter the behavior of other SON instances. In such cases, the linearity considered in the proposed approach is no longer valid, which may result in misleading learning feedbacks or even prevent the convergence of the process. Consequently, a thorough analysis of the correlation and interaction between the deployed SON functions and instances for a certain scenario should be done before launching the proposed C-PBSM learning process. Studying deeply the correlation between SON functions may not be easy as they are designed as black boxes with limited information. The other solution would be to classify the cells into geographical clusters with low correlations, in order to guarantee the linearity of the model. This clustering would take into account the topology of the network as well as the traffic distribution and evolutions.

Chapter 5

Softwarized and Distributed Learning for Cognitive SON Management

5.1 Introduction

In the previous chapter, we have presented an approach to enhance the Policy Based SON Management with cognition i.e. C-PBSM, through a RL loop using MAB algorithms. In 4.2 we have used a stochastic MAB algorithm, that is the UCB1, and showed its optimality. The regret of UCB1 scales linearly with the number of arms, which can considerably slow down the convergence when the number of SCV sets combinations increases. In 4.3 we have proposed a linear model to enhance the convergence speed of the C-PBSM. We have used the LinUCB algorithm, which scales logarithmically with the number of arms, hence leading to faster convergence. This said, the previous approaches perform well over a limited section of the network and after reducing the number of arms by considering intuitive cell classes and configuring the SON instances per cell class. In this chapter, we address the following concerns:

- How can the C-PBSM efficiently learn over the complete space of SCV sets combinations (without considering cell classes) and does it enhance the performance of the C-PBSM?
- How to make the C-PBSM scalable with the size of the network? In other words, how to generalize the C-PBSM and the RL loop to bigger network sections with hundreds of sites and more diverse SON functions, and hence very large action spaces?

In this chapter, we tackle these concerns by proposing a distributed RL approach. Several RL agents are deployed and learn in parallel the optimal configuration of SON functions with respect to global operator's objectives. Each of the RL agents is responsible of a specific area or cluster of the network. Ideally these clusters would be defined in such a way that the agents can learn independently, that is, configuration changes of a cluster would not affect neighboring clusters. As the interaction between these clusters may evolve in time due for instance to traffic dynamics, a flexible implementation of this C-PBSM framework with dynamic clustering to adapt to network's evolutions is proposed. We show how this flexible implementation is rendered possible under SDN framework.

The use case study consists of a heterogeneous network with distributed SON functions. Practically in the RAN, independent clusters as described in the previous paragraph (i.e. clusters that do not interact with neighboring clusters) are usually defined over large areas of the network such as cities, or neighborhood. They hence include multiple sites, leading to a big C-PBSM action space, which leads back to the scalability problem. During this study we found that defining smaller

independent clusters with limited number of sites is ambiguous and needs to be well investigated. Consequently, after deep investigations, it was decided not to tackle the clustering problem in this work. Instead, we propose an alternative solution where each cluster is defined as a single macro cell (and the deployed small cells in its coverage area possibly). Each agent configures the local SON functions instances that are deployed in its corresponding cluster and has a local view of the KPI outputs (local rewards). However, in the distributed learning framework, the i.i.d hypothesis of the local rewards of each agent is no longer valid because of the interaction generated by close learning agents. The state of the environment is hence changing and has to be observed and taken into account in the RL process.

5.2 Distributed Learning Under a SDN Framework

The scalability problem is a well known issue in the RL framework [2]. In our work, the scalability problem is the following: when the network size, the number of SON functions and instances, as well as the number of possible SCV sets for each SON function grow, the action space that has to be explored by the RL agent explodes, leading to a very slow convergence of the C-PBSM to the optimal policy. In fact most of real life problems have huge action and state space and standard RL algorithms fail to efficiently deal with such environments. Distributing the learning process is one of the most efficient method to deal with the scalability problem [45,46]. Distributed learning consists in decentralizing the learning process by deploying several RL agents, where each agent learns based on the observations of a sub-domain of the environment (local environment) where it is deployed. As stated in chapter 2, distributed learning has several advantages compared to the single agent learning: scalability, computational efficiency, maintainability and robustness.

RAN are complex dynamic environments. This is mainly due to the fact that in RAN, multiple devices such as user equipments, base stations, radio heads ... are accessing the same medium and interfering with one another. Changing certain configurations of a single cell, for example antenna tilt or transmit power, will affect neighboring cells in terms of interference, coverage, traffic load, etc. Furthermore, RANs are highly influenced by the users traffic dynamics, which in turn depend on the human and social activity. Traffic varies in two dimensions: time (hour of the day, period of the year) and space (office, residential or rural areas, etc). In [104], the authors show that there are four main traffic profiles, that depend on the main human activity in a certain geographical location: business area, residential area, entertainment area and transport area. Furthermore, the traffic profiles change on the long term due to the evolution of mobile services (services requiring higher data rates, machine type communications, ultra reliable and low latency services, etc), modifications of the RAN topology (the installation of new sites) and also due to the human activity and the urban expansion (construction of a mall or a new residential building).

Opting for a distributed learning approach in such environments is not straightforward, and must be considered with caution because learning agents may interact with one another. In fact, because of the complexity of the environment and the correlation between cells, changes made by the RL agent to a certain cell will affect the environment of neighboring cells that may belong to the domain of another agent, hence interfering with each other's observations. Hence, independent RL agents should be distributed, each being responsible of a cluster of neighboring cells. To guarantee that the learning processes are independent, neighboring clusters should have no or minimal interaction with each other. This said, the number of cells in a cluster should also be kept as low as possible. Otherwise, the sets of actions and states of the distributed agents would explode, leading to a slow convergence and hence to the initial scalability problem.

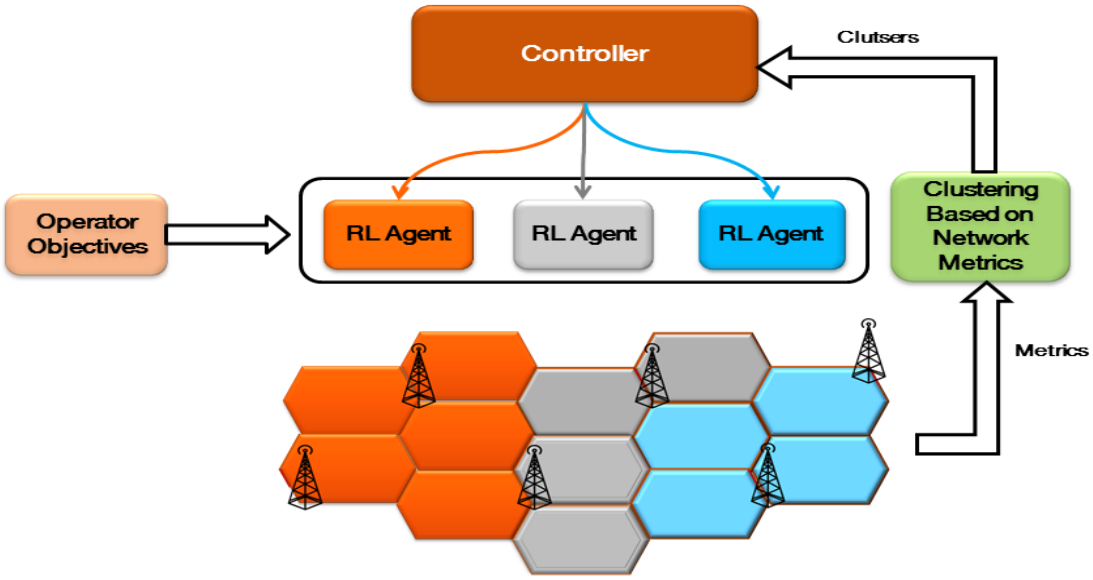


Figure 5.1: Distributed RL

Therefore, a clustering process based on network metrics should be performed, before deploying the RL agents, in order to define the clusters with minimal interaction. This interaction can be expressed in terms of interference level between the cells, the traffic flux between cells (expressed through HO metrics of incoming and leaving traffic between the clusters) and the coverage area overlapping (small cells deployed in the coverage area of macro cells for example) [105]. Defining clusters with minimal interactions with respect to the previously stated criteria, while keeping their size as minimal as possible (to preserve a limited size of actions and states), requires deep studies and investigations. In fact, the clustering process should be done using real network data and metrics about traffic flux, HO statistics, coverage maps, network topology and sites locations. These metrics should be accessed and sampled from large areas of the network in order to output relevant clusters. Unsupervised ML techniques, combined with the Big Data framework are a potential track to solve the aforementioned clustering problem, which is out of the scope of this work and remains an open problem [106, 107]. Alternatively, we consider a rather straightforward deployment of learning agents over small network clusters. These clusters are however not independent and will interact with each other during the learning process. We propose to tackle this issue using a distributed Q-learning approach as will be detailed in section 5.4.

On the other hand, because of the network evolutions mentioned previously (due to new mobile services, changes in the RAN topology, urban expansion, etc), the defined clusters should be kept monitored and adapted to these changes. The clustering process should hence be dynamic, meaning that new clusters can be introduced and existing clusters can be modified on the run. This requires a dynamic redistribution of new learning agents or modification of the environment of the deployed ones. Whence the need of an architecture able to provide such levels of modularity and flexibility. This can be provided by the SDN framework, as will be discussed in the following.

The deployment process is presented in figure 5.1. The clustering entity gathers the required metrics from the network and outputs the relevant clusters to a controller. The controller will then associate each RL agent to a network cluster. Once deployed, each RL agent learns a local optimal policy over a cluster of the network according to an operator defined objective: the agent

configures the SON functions deployed in the considered cluster, and observes the KPI outcomes and the states of the local system as well (traffic, network configuration ...).

5.3 Software Defined Networks for SON Management

The SDN concept is based on separating the control plane and the data plane, logically centralizing the control process and defining an Application Programmable Interface (API): the southbound API ensures the communication between the control plane (the SDN controller) and the data plane (network devices) and the northbound API ensures in its turn the communication between the control plane and the software applications and services in the network 5.2. Such a separation permits the two planes to evolve separately, hence introducing new levels of flexibility and abstraction. Also, it helps providing a global and centralized view of the network, thus facilitating the introduction and operation of control processes in the network [3].

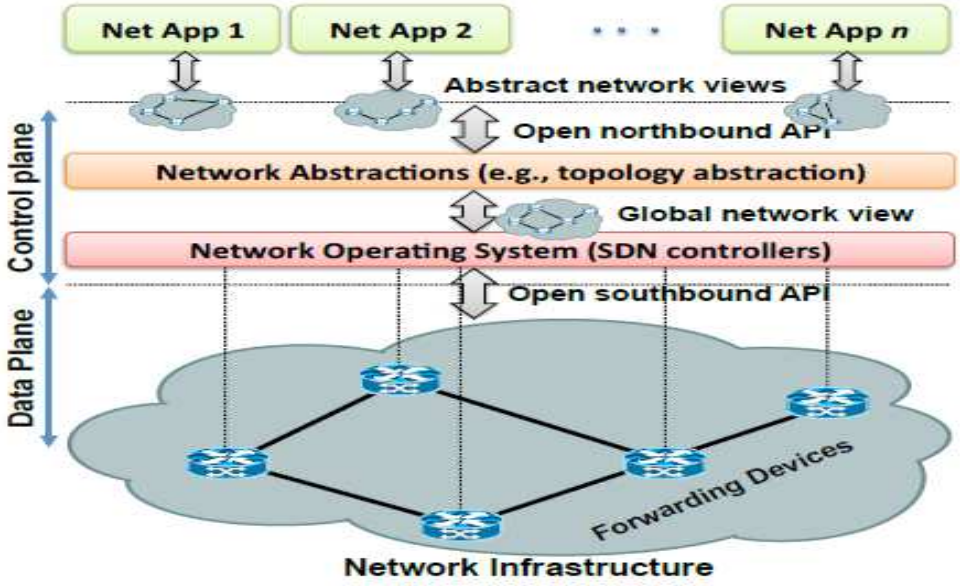


Figure 5.2: SDN Architectures and Abstractions [3]

Figure 5.2 shows the separation control and data planes, as well as the software applications running on top of the network controller. Such applications may include routing algorithms, traffic load balancing, security related applications, etc. The network application is hence only responsible of defining a certain network behavior. The task of implementing that behavior in the corresponding device is left to the SDN controller. In other words, network elements become simple forwarding devices, taking their instructions from the SDN controller through the southbound interface. Consequently, the integration of new services and applications in the network becomes easier, more flexible and vendor-agnostic.

Although the first notable applications of SDN architectures were oriented towards core networks and data centers [108, 109], the SDN concept gained also popularity in mobile networks such as OpenRadio and SoftRAN [110, 111]. AutoSDN in its turn is a SDN controller for RAN oriented towards autonomic-network management [112, 113]. AutoSDN's objective is to introduce a new abstraction level to self organized networks that enables SON programmability. With AutoSDN, the SON functions become software applications that the controller compiles and ex-

ecutes. The parameters and metrics of the network elements are then monitored and updated by the controller's southbound interface. This approach ensures higher flexibility to the autonomic management of RAN. In this work however, our objective is to push self organization to a higher level, by automatically translating operator objectives into proper SCV sets for the SON functions instances deployed in the network.

SDN and self-organization are related in the sens that they both seek to simplify and improve the management of complex heterogeneous networks, hence reducing the operational cost and delivering better QoS to the clients [114]. Furthermore, SDN can be used as a mean to facilitate the introduction of self-organization and intelligent control loops in complex networks with multiple technologies, layers and vendors. In this paragraph we discuss a SDN architecture, based on the SDN for RAN in [113] that could be adopted to deploy a dynamic and distributed RL for SON management as represented in figure 5.3. The learning agents are instantiations of the RL

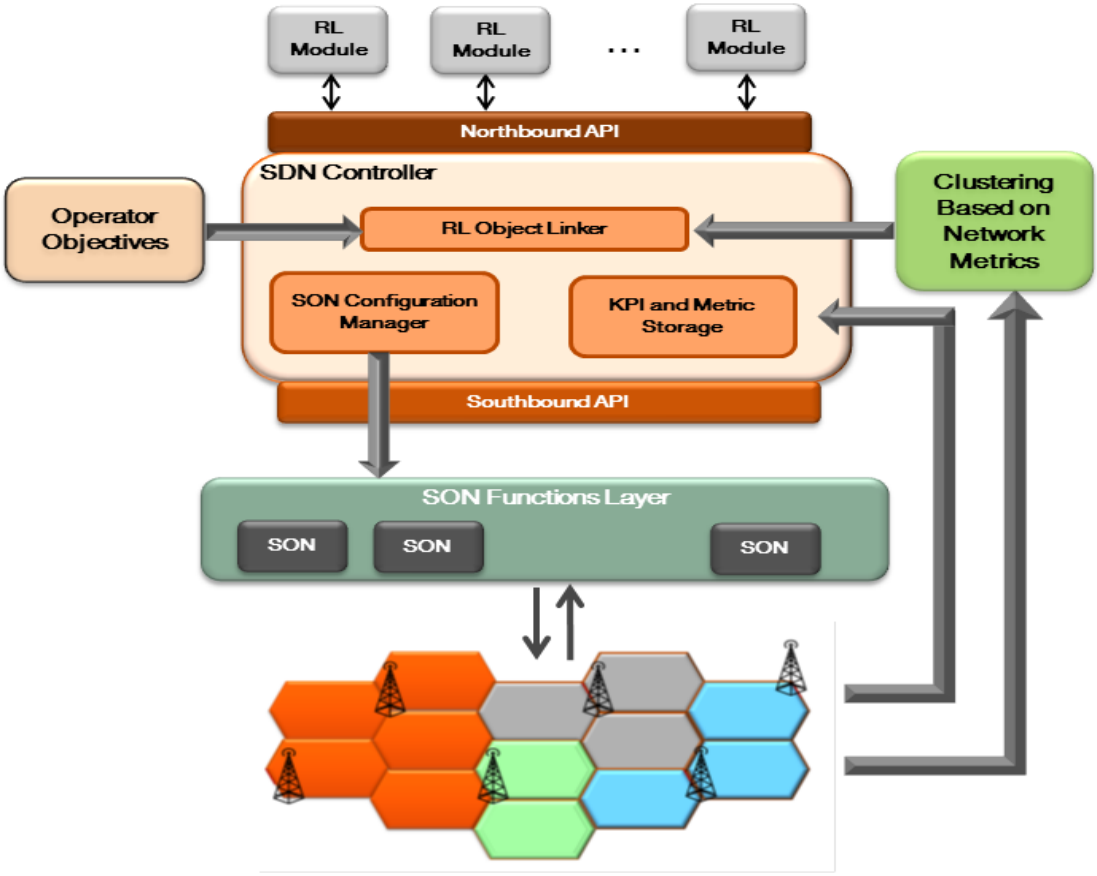


Figure 5.3: C-PBSM based on Distributed RL deployment through SDN

modules, each of them encapsulates the RL code, inputs and output. The inputs of a RL algorithm include the operator's objective and the set of KPIs, metrics and indicators describing the observed environment and the reward functions. The output is the agent's decision, which is in this case the SCV sets of the SON function instances monitored by the agent. The code of the RL object can be any RL algorithm that the agent runs and that permits it to explore and learn the optimal policy such as Q-learning, SARSA, MAB algorithms, etc [2]. The RL modules are loaded and compiled in the RL object linker, where the operator's objective is specified as well as the network

cluster that will be controlled by each agent. The modules are then executed in the controller. The clusters are specified after clustering based on network metrics and topology and are continuously monitored and modified as stated previously. Updated clusters can hence be reintroduced on the fly in the controller, enabling the RL process to adapt accordingly. This applies also for the eventual objective changes. The RL algorithms require constant observations of the environment and the generated rewards. The KPI and metrics storage entity keeps an inventory of network KPIs and metrics that can be accessed by the agents. After observing their environments' states and reward, the agents take local decisions and forwards them to the SON configuration manager. The SON configuration manager then enforces the RL decisions in the concerned deployed SON functions instances.

Note that in this architecture, the RL modules are merely high level software applications, which simplifies the introduction of new RL algorithms in the system and makes their deployment more flexible and hence adapted to the dynamics of the radio environment.

5.4 Scenario Description

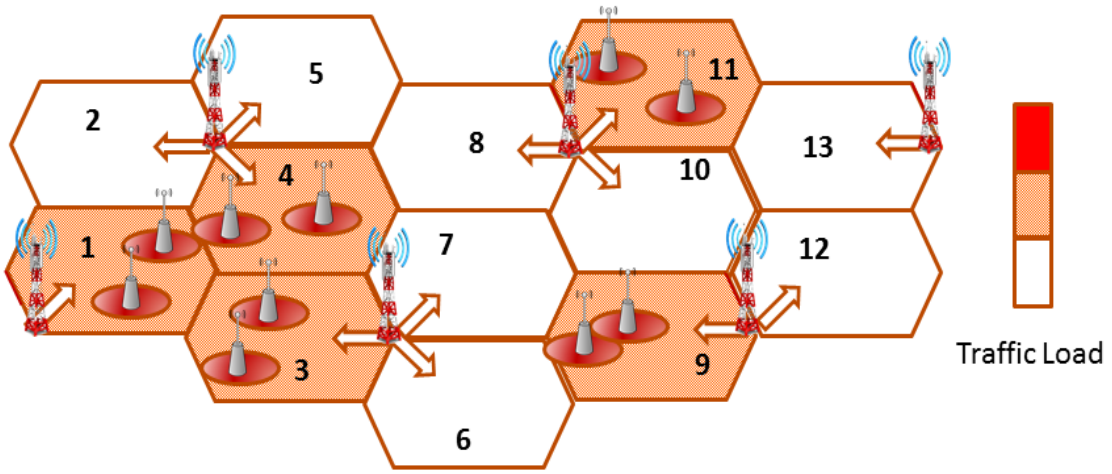


Figure 5.4: Network Model

We consider a heterogeneous network, as depicted in figure 5.4, where several distributed SON functions are deployed: MLB on each macro cell, CRE on each small cell and eICIC on each macro cell having small cells deployed in its coverage area. The main simulation parameters are summarized in table 5.1. As mentioned previously in this chapter, independent network clusters are usually defined by operators over large areas that include multiple sites. This is due to the fact that changing certain configurations of a cell will impact close cells, depending on the traffic flux between them, as well as on the interference level and coverage map. On the other hand, if the number of cells in a cluster is not small, then the number of SCV sets combinations over the cluster will explode, leading back to the scalability problem that we are attempting to tackle. Consequently, deploying independent RL agents with reasonable action sets is complicated in the RAN, and still requires deep investigation and research studies as explained in section 5.2.

In this work, we do not tackle the previously mentioned clustering problem. Instead, we consider a straightforward deployment of the learning agents. That is, we consider a RL agent

Parameters	Settings
Bandwidth	10 MHz
PRBs per eNB	50
Carrier Frequency	2 GHz
Macro ISD	1732 m
Macro Path Loss to UE	$128.1 + 37.6 \times log_{10}(d[Km])$
Pico Path Loss to UE	$140.1 + 37.6 \times log_{10}(d[Km])$
Antenna Gain	macro: 14 dBi; pico: 5 dBi
Transmit Power	macro: 46 dBm; pico: 30 dBm
Shadowing Standard Deviation	macro: 8 dBm; pico 10 dBm
Traffic model	Dowlink File Transfer Protocol (FTP), file size: 14 Mbits
RL Simulation Time Iteration	20 min (in simulation time)

Table 5.1: Simulation parameters

i on each macro cell. If the macro cell has small cells in its coverage area (slave cells), they will belong to the same learning cluster as the macro. The cluster is hence the single macro cell or the macro cell with the small cells, if they are deployed in its coverage area. This means that the learning agents are not independently distributed. Consequently, the learning processes that will run simultaneously in the network will be interfering with each other. Therefore, the assumption that for the same configuration combination c, the observations of r are i.i.d. does not hold anymore. Each learning agent should hence observe the state of its local environment, and adapt with its changes. The stochastic MAB such as UCB and LinUCB fail to deal with such environments, because they assume i.i.d. observations of the reward, whereas in this case, the reward at each iteration depends on the environment's state. Consequently, we propose to use a distributed Q-learning approach.

Recall that, as explained in chapter 2, in the distributed Q-learning, the learning model is no longer strictly MDP. This is due to the fact that an agent's decision can change its environment, but also the environment of other agents. Instead, the agents are approximated as part of the environment, and the learning model is an approximated MDP. Even though the theoretical convergence proofs for single agent Q-learning do not hold, this model has however been used successfully in many cases [47, 48, 115].

In the following, we consider that each agent has an action space C_i, depending on the SON deployed in its cluster, and a state space S_i:

- Action space of agent i: $C_i = (V_{s_1})^{N^i_{s_1}} \times (V_{s_2})^{N^i_{s_2}} \times ... \times (V_{s_{|F|}})^{N^i_{s_{|F|}}}$ where N^i_s is the number of instances of SON function s in network cluster i. At each iteration, the agent picks an action $c_i \in C_i$.

- The state of an agent i is defined as follows:

$$s_i = \{I_i, I'_i\} \tag{5.1}$$

  where I_i is the load indicator of the macro cell where the agent i is deployed and I'_i is the average load indicator in the first tier macro cell neighbors. In other words, we consider that the interaction between the learning clusters is limited to first-tier neighbors, and that it impacts the load of the macro cells only (we neglect the impact of neighboring learning agents on the small cells).

- A reward function: $r^i_{t,c} = \omega_1(1-\sigma^i_{t,c}) + \omega_2\overline{T}^i_{t,c} + \omega_3\overline{T'}^i_{t,c}$ where $\sigma^i_{t,c}$ is the load variance, $\overline{T}^i_{t,c}$ the average user throughput and $\overline{T'}^i_{t,c}$ the average pico cell edge user throughput when action c is applied in network cluster i at iteration t.

The pseudo-code of the distributed Q-learning algorithm for C-PBSM is presented the following,where $rand(C_i)$ refers to picking a random action from the set of actions C_i of agent i, γ is the discount factor and α is the learning step.

Algorithm 7 Distributed Q-Learning

for each agent i
Initialize $Q_i(s,c)$ arbitrarily
Initialize s_i
for t=1,...,T
- for each agent i
- - pick action c_i for state s_i according to ϵ-greedy policy:

$$c_i = \begin{cases} \underset{c_i \in C_i}{\operatorname{argmax}}\{Q_i(s_i,c_i)\} & \text{with probability } 1-\epsilon \\ rand(C_i) & \text{with probability } \epsilon \end{cases}$$

- - observe new state s'_i
- - observe perceived reward $r^i_{t,c}$
- - update Q_i function as follows:

$$Q_i(s_i,c_i) \leftarrow Q_i(s_i,c_i) + \alpha[r^i_{t,c} + \gamma \max_{c'_i \in C_i} Q_i(s'_i,c'_i) - Q_i(s_i,c_i)]$$

- end for
end for

5.5 Simulation Results

We run the C-PBSM for different operator objectives, while considering that all agents have the same operator objectives at each simulation. SON functions details are given in chapter 3 and the SCV sets we use are summarized in table 5.2. In the following we study the behavior of learning agents, and the optimality of the output policy. We compare the distributed learning performances with the centralized stochastic MAB approach. Note that the distributed Q-learning algorithms, as any other learning scheme, need a learning phase to learn the optimal decision policies. During the learning phase, the Q-learning algorithm updates and stores the Q-functions values in a lookup table (Q-table), where each state is mapped to the corresponding optimal action. Consequently, once the learning phase completed and the policy acquired, each agent will have an optimal Q-table that can be directly exploited by observing the current state and taking the corresponding optimal action.

Figure 5.5 shows that the evolution of the perceived reward for one of the learning agents. The results show that the reward converges towards a stationary reward, for different operator objectives. Note that the stationary reward has different values depending on the operator objectives. This is due, as in the MAB cases, to the normalization of different KPIs with different distributions, ranges and behaviors, and do not reflect the optimality of the policy.

SON	SCV sets
MLB	Off: Function is turned off SCV1: Soft configuration $(\Phi_{min,i} = 0dB; \Phi_{\max,i} = 6dB; l_{L,i} = 0.55; l_{H,i} = 0.85; \Delta_i = 1dB)$ SCV2: Aggressive configuration $(\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.725; l_{H,i} = 0.775; \Delta_i = 2dB)$
CRE	SCV1: Soft configuration $(\Phi_{min,i} = 0dB; \Phi_{\max,i} = 8dB; l_{L,i} = 0.6; l_{H,i} = 0.85; \Delta_i = 2dB)$ SCV2: Aggressive configuration $(\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.8; l_{H,i} = 0.8; \Delta_i = 2dB)$
eICIC	Off: Function is turned off SCV1: Function is turned on $(\kappa_{max,i} = 8ABS; R_{L,i} = 1; R_{H,i} = 2.5; \Delta_i = 1ABS)$

Table 5.2: SCV Sets Description

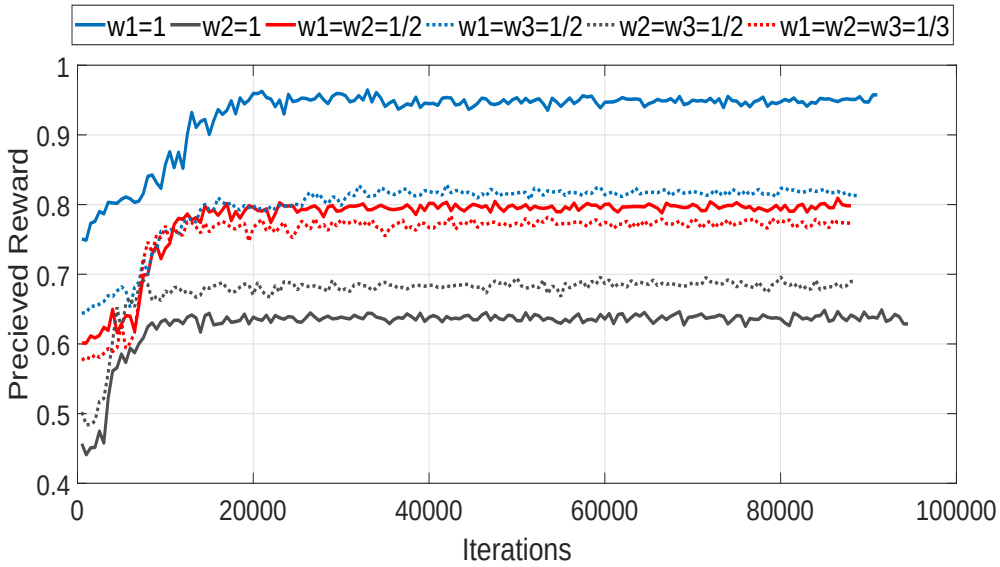


Figure 5.5: Q-Learning Iterations

As for the performance of the optimal policy acquired by the distributed Q-learning process, we simulate the network where each agent applies its local policy that was acquired during the learning phase, for a set of operator objectives. We evaluate the reward over the whole considered network section (as defined in 4.2.3) and we compare the performances of the distributed Q-learning policy with the centralized learning through stochastic MAB (namely the UCB1). In order to reduce the action space of the MAB, we consider a cell classification similar to the one we adopt in chapter 4, i.e. we consider 2 cell classes: A class of macro cells with high traffic and where a network layer of small cells is deployed $\{1, 3, 4, 9, 11\}$ and a class of single layer macro

cells with low traffic $\{2, 5, 6, 7, 8, 10, 12, 13\}$, leaving the UCB1 with an action space of 36 arms (details in chapter 4). Results are depicted in figure 5.6 for a set of operator objectives. The results show that the distributed Q-learning and the centralized MAB have similar performances in terms of final average perceived reward for different operator objectives except for $\omega_2 = 1$ case. This improvement can be explained by the loss that could possibly be induced by the classification we use to reduce the action space of the MAB. There is hence possibly a room to further improve and optimize the cell classification process through more rigorous techniques. However, the intuitive classification we use showed to be useful, for we were able to drastically reduce the action space of the MAB (from the whole combinatorial space of SCV sets that is of order 5×10^{10} to 36 actions) while maintaining good performance for most of the operator objectives.

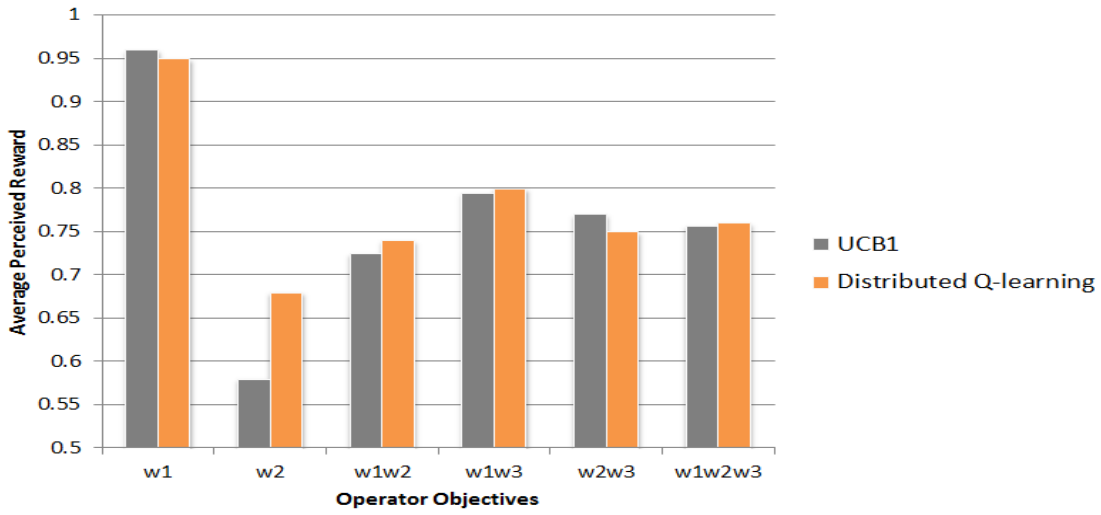


Figure 5.6: Final Policy Average Perceived Reward

In this chapter, we have proposed an efficient and scalable approach, that can be generalized over big network sections, by distributing the learning processes. This approach is indeed scalable with the size of the network and is able to manage hundreds of sites. It also enjoys a high level of modularity and flexibility, rendered possible through a SDN based deployment of the learning agents. This approach is also able to learn over the whole action space of SCV sets combinations. Results have shown that cell classification is efficient in reducing considerably the action space while keeping good performances in terms of average perceived reward for most tested operator objectives (except for ω_2, figure 5.6). The classification can still be improved through more rigorous machine learning techniques. We have also explained that finding small independent clusters in the network is practically hard. Instead, we proposed to manage the interactions between the clusters through distributed Q-learning. However, distributed Q-learning still assumes a stationary environment, and hence a stationary traffic. Also, the local optimal strategies and the knowledge of the learning agents do not take into consideration the network context where they are deployed. They are hence specific to the local environment of the agent and are not easily transferable to other cells in different sections of the network. Finally, the adaptability with the operator objective changes is not as straightforward as with the previous chapter. Because in the stochastic MAB case, namely in UCB1 and LinUCB, the action selection strategy is based on an empirical estimate of the reward and an upper confidence bound. Also, the reward function is a linear combination of KPI targets realizations and priority weights, and the upper confidence bound depends only on the number of times an action was tested and the number of iterations of the algorithm. There-

fore, memorizing the estimate of each possible KPI target of each of the tested actions, regardless of the operator priorities, permits an instant evaluation of the new rewards estimates in case of priority weights changes. This permits a fast adaptation with the operator's objective changes. The Q-learning on the other hand relies for its action selection strategy on the Q-values, which depend on the perceived reward but also on the environment dynamics and the Q-values of future states. Consequently, even if it is possible for the algorithm to memorize logs and estimates of each KPI target of each state-action pair, adapting the Q-values and the Q-tables with operator priority changes is not immediate and requires offline training and learning in order to exploit the information gathered during the online learning of previous operator objectives.

In the following chapter, we propose a different approach, where a centralized agent learns simultaneously and collaboratively over different network clusters. We show that this approach is able to learn for different network contexts, tackles the network dynamics, specifically the traffic variations, and builds a knowledge database that is context aware and hence transferable over different sections of the network.

Chapter 6

Context Aware Cognitive Policy Based SON Management

6.1 Introduction

The behavior of a SON function, as already explained, depends on the configuration of its algorithm, i.e. SCV sets, but also on the RAN environment context where it is deployed. The context can include the network density of the geographical area (rural, sub-urban,urban), the functionality of the geographical area (railway, highway, office, residential, etc), the traffic state, the radio access technology (2G,3G,4G,5G), the network topology (n layers heterogeneous network), the type of the network equipments, etc.

On the other hand, RAN environments are known to be non static environments. They are dynamic and change constantly. This requires continuous monitoring and optimization by the operational teams. One of the most impacting dynamics in the RAN is the traffic evolution. The traffic state in the network varies in two dimensions: spatial dimension, depending on the geographical location and temporal dimension, depending on the hour, the day, the week and the month. There are several traffic profiles in the RAN that depend on the previously mentioned parameters [104]. Furthermore, the average traffic in the network is continuously growing due to the increase in the number of devices, that are becoming smarter with more powerful processing capacities. The increase is also due to the emerging various services, with finer requirements in terms of throughput, latency and reliability [116, 117]. Other dynamics that impact the RAN environment include the installation of new radio sites, the introduction of new frequency bands, new radio access technologies, etc.

The learning process and the SON management policy should hence take into consideration the network context and adapt with its changes and variations. In the previous approaches, a strong hypothesis was taken: stationarity of the traffic. In fact, when considerable changes happen with the traffic distribution, the performances of the network change, hence distorting the stochastic observations that the learning algorithms rely on to learn, which slows considerably the convergence or leads to suboptimal policies that do not adapt with the environment's dynamics. This is a strong hypothesis because traffic distribution changes occur often in mobile networks as already mentioned and should hence be dealt with. The approaches studied so far: stochastic MAB, stochastic Linear MAB and distributed Q learning, require stationary environments to efficiently learn and preserve their theoretical convergence guarantees.

On the other hand, when it comes to applying online learning processes, the time for convergence is critical. During the exploration (learning) phase, the QoS in the network may degrade

before converging to the desired performance. Stochastic MAB, combined with linear action-reward models, have acceptable convergence time, even though the linear assumption should be used with caution as argued in chapter 4. However, the efficiency of such centralized approaches is still limited to relatively small network sections with rather similar contexts. They are hardly generalizable to the whole network because the action set would increase very fast. Also, when generalizing to very large chunks of the network, various contexts will appear. A more rigorous cell classification will hence be required to define classes of similar cells, in order to avoid a huge action space, which is a complicated task. In chapter 5 we have proposed a distributed learning approach, where learning agents are distributed all over the network, and learn simultaneously local optimal policies over small network clusters. This approach can indeed be generalized over the network and is scalable with respect to the action space. However, learning clusters should ideally be defined such that the interaction between them is minimal, so that simultaneous learning processes do not interferer with each other. Furthermore, the size of the clusters should be kept small so that the action space, that is the combination of the SCV sets of all the deployed SON instances, does not explode. In the previous chapter we have argued that defining small cluster in the RAN with very low interaction is in fact complicated in practice. Instead, we have proposed a distributed learning framework where learning agents are distributed over small clusters, namely each cluster encompasses a macro cell and the possibly deployed small cells in its coverage area. The learning processes over these clusters are however not independent. Consequently, each cluster has to observe the effect of the actions of neighboring clusters on the local state of its environment and adapt its policy accordingly. To do so we have proposed a distributed Q-Learning approach that showed to be able to find optimal policies.

In this chapter, we propose and study another RL approach, based on contextual MAB. We investigate three critical points in the C-PBSM framework that are:

- Learning over different network contexts, and deriving optimal policies according to the context. Knowing that in a RAN, there is possibly a large number of context variables and parameters. Then how to identify the ones that influence the most on the SON functions behaviors, and consequently the C-PBSM's performance.

- Leveraging the RAN environment dynamics, in particular the traffic dynamics and variations.

- Efficiently transferring the acquired knowledge over different sections of the RAN.

For this purpose, we propose and study a context aware C-PBSM approach as well as an implementation architecture. The proposed C-PBSM learns the optimal policy taking into consideration the environment context information it receives at each iteration. The learning is carried out by a centralized learning agent, over several network sections that we henceforward refer to as learning clusters. The learning processes on the different clusters run simultaneously and the learning clusters can be situated in different geographical locations. The knowledge acquired by the learning agent is stored in a centralized database that we refer to as knowledge database. The agent is continuously communicating with the simultaneously learning clusters as follows: it receives the reward and context information from the clusters, and chooses an action to be applied in each of them, depending on the reported context and its knowledge database. The knowledge is updated and stored in the centralized database at each cluster feedback. A functional scheme is represented in figure 6.1.

This approach permits the C-PBSM to find the optimal policy for each observed context and adapt this policy with network context changes. We consider traffic variations in the network

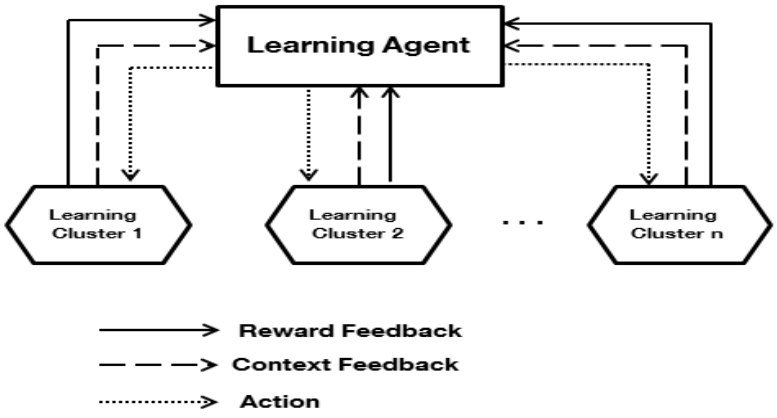


Figure 6.1: Context Aware C-PBSM Scheme

according to well known traffic profiles [104]. We also consider that these changes are piecewise-stationary with respect to the learning iterations. In other words, the traffic variation rate is considered to be much slower than the RL iteration time, so that we can consider the traffic to be stationary inside each RL iteration of each cluster, before changing to a new state according to the traffic profile. The traffic state of each cluster is given as part of the context information to the agent at each iteration, along with other information about the cluster's topology and environment.

Being able to simultaneously learn on different clusters in the network has two important advantages. First, if the picked clusters are able to cover the different contexts present in the network, so will the optimal policies of the acquired knowledge. This means that once this database completed (i.e. the learning processes on the different clusters have converged), the policies it contains can be transfered and applied to other untrained sections of the network, through an open control loop. Second, if similar contexts can be observed on different learning clusters, then the learning becomes collaborative. The reason is that the knowledge acquired for a certain context in a certain cluster, is valid for the same context on a different cluster. Since the knowledge is stored in a centralized database, and the learning agent updates this database at each cluster feedback and takes a new decision based on this feedback and on the knowledge at this iteration, then the learning becomes collaborative between the clusters, increasing hence the speed of convergence on all the clusters.

In short, the proposed C-PBSM is a capable of

- learning over different network contexts
- adapting to traffic variations
- transferring its knowledge and policies to different untrained sections of the network
- improving its convergence speed through simultaneous learning over different clusters and through collaborative learning, depending on the number of clusters and their similarities.

6.2 Context Aware C-PBSM

We consider a SON enabled network, with distributed SON functions. Recall that throughout this work, we consider that the SON functions are provided by SON vendors to the network operator,

and that they are seen by the latter as black boxes. In this chapter, we propose a RL approach based on contextual MAB, where the centralized agent trains and builds its knowledge over a possibly large number of distributed and various network clusters. The goal is to build a centralized model that is representative of the overall context-action-reward distribution in the network.

A learning cluster is typically a small group of limited neighboring sites, having similar characteristics, where the operator allows the testing of SON configurations and parameters. The cluster characteristics can be for example the geographical area (rural, urban, railway, highway, etc), the technology (2G, 3G, 4G, etc), the topology (n layer heterogeneous network) or the type of network equipments. All these characteristics affect the behavior of SON functions, and should hence be integrated in the context information that will describe the learning cluster. Moreover, the context should also include the traffic of the cells of the learning cluster. These clusters enjoy certain important characteristics. For instance because they are limited to few neighboring cells, the action space of each cluster stays reasonably small, which insures an efficient learning process with reasonable convergence time. Also, learning clusters can be distributed in different geographical areas. In this case, the different clusters do not face the problem of interfering with each others' learning processes (observations and decisions). However, defining the learning clusters still needs to take into consideration the traffic flux, interference and coverage overlapping with neighboring cells. First, because the learning environment needs to have a minimal level of stability in order to derive relevant and optimal policies. Second, because when exploiting the learned policies and enforcing them in different parts of the network, the interaction of the clusters with their surroundings (other clusters where policies are enforced) should be minimal,in order to guarantee the expected performance and gain. The proposed architecture is depicted in figure 6.2. The C-PBSM consists of the AI Layer and the SON Configuration Manager:

- AI Layer: This layer contains 2 blocks: the AI Agent and the Knowledge Database. The AI agent runs a contextual MAB algorithm. It observes the reward functions and the context information coming from the network. The measured KPIs are kept in a registry called Network Measurements Pool. Before being forwarded to the AI Agent, the raw KPIs measurements are statistically processed (typically averaging, smoothing, scalarization, normalization, etc). The operator objectives are given to the agent as input. The AI agent will learn the optimal policy for the specified operator objective, based on the reward and context information, but also on the centralized model it had already built and stored in the Knowledge Database. After taking an action and perceiving the reward, the AI agent updates the model of the Knowledge Database.

- SON Configuration Manager: This block receives the actions from the AI Agent and is in charge of enforcing the SCV set in the corresponding SON function instances in each of the learning clusters. The SON Configuration Manager can change certain network parameters directly as well.

Note that each learning cluster will normally contain a subset of all the contexts that can be encountered in the network. Hence, in order to build a model in the Knowledge Database that represents to the best the whole network, the clusters should be picked from different sections of the network. The more the clusters are diverse and sample well the context-action space of the network, the more the Knowledge Database is complete and transferable to other untrained sections of the network. Furthermore, besides helping construct a more complete model, increasing the number of learning clusters will increase notably the speed of convergence. First because there will be much more SCV sets combinations tested in parallel in different network clusters. Second because all the clusters contribute in building the same centralized model. Hence the knowledge

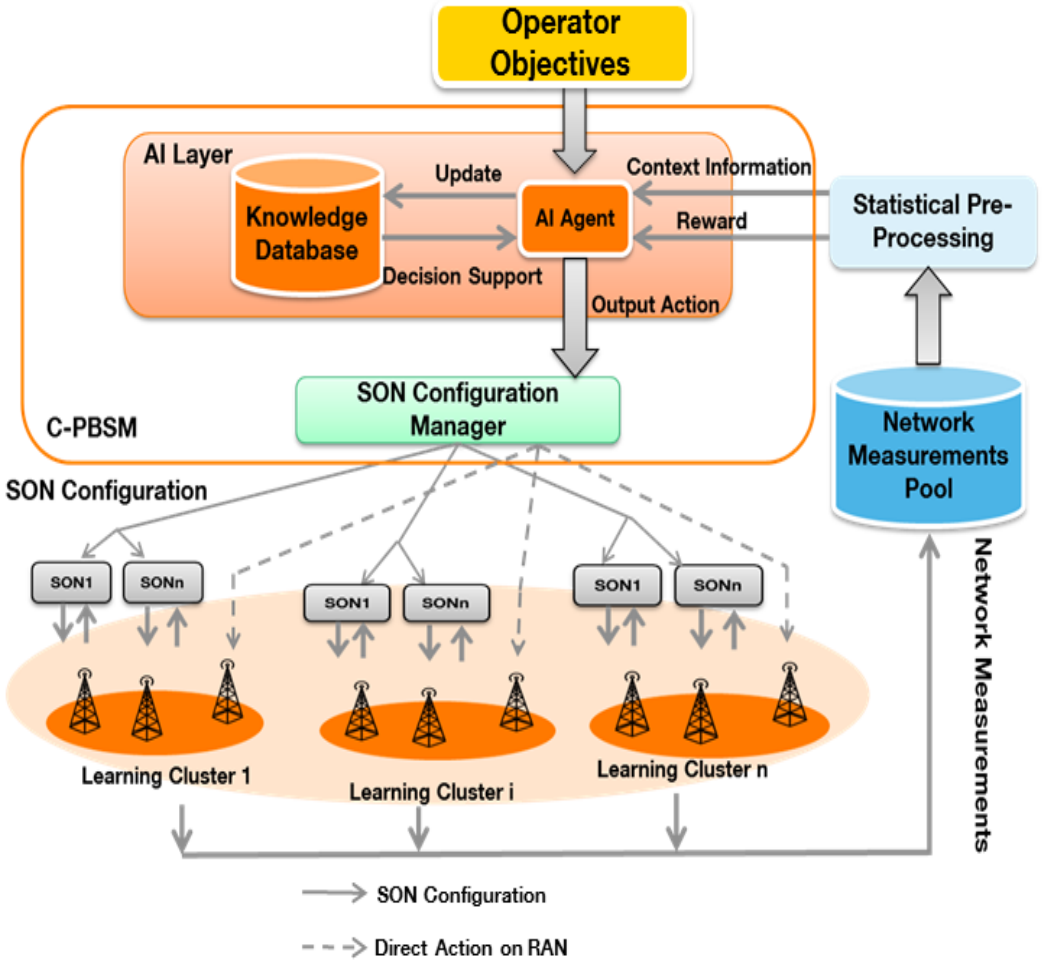


Figure 6.2: Context Aware Distributed C-PBSM

acquired in a certain cluster will be updated in the centralized model through the feedbacks sent to the AI Agent. The updated model will then be used as a support for the AI Agent to take a decision in different clusters. The learning is hence done collaboratively, which improves remarkably the convergence speed.

In the following section we motivate and discuss the contextual MAB algorithm we use to implement the AI agent and the Knowledge Database of the C-PBSM.

6.3 Context Aware C-PBSM: Implementation Based on Contextual Multi-Armed Bandit

In this chapter we propose an approach for C-PBSM based on a contextual MAB algorithm, namely the random forest algorithm for the contextual bandit problem [65], that we henceforward refer to as Bandit Forest (BF). The BF algorithm we implement belongs to the decision trees family. It considers that there is a subset of context variables that are more relevant than the rest, and grows its decision trees based on these variables. It hence reduces its exploration space by considering this subset of variables, and discarding the others. The BF algorithm was proposed and analyzed in details in [65]. The authors analyzed the optimality of the algorithm in terms of

sample complexity. Note that the sample complexity is the number of iterations needed to find an optimal policy with a certain success probability [56]. BF was shown to be optimal up to a logarithmic factor. Furthermore, the dependence of the algorithm's sample complexity with the number of context variables is logarithmic, which means that the algorithm scales well with the number of context variables. This is a very important factor, because as described previously, the complexity, heterogeneity and high dynamics of the RAN environment may require a huge number of context variables to describe a certain network section. The computational cost is as well linear with the time horizon and the number of context variables, allowing to process large sets of variables. Finally, the BF algorithm enjoys key characteristics for real applications. For instance, it does not consider linear dependencies between the perceived reward and the context variables. Even though linear model assumptions are often used in many learning frameworks and proved to be efficient in many cases, they are still strong assumptions and need to be handled carefully (chapter 4). Algorithms based on linear models may thus fail, for instance when the context variables have non-linear dependencies with the rewards. BF algorithm is therefore better suited for such cases. Also, multiple contextual MAB algorithms assume a similarity information about the context-arms pairs to be available and base their exploration/exploitation strategies on this information e.g. [63]. However, it is not simple to have such information in in practice. The BF algorithm does not need similarity information to find efficiently optimal policies. BF algorithm is hence well suited for real applications, notably for a context aware C-PBSM implementation.

6.3.1 Contextual MAB

The contextual MAB problem is formalized as follows. Let A and K be respectively the set and the number of possible arms. Let V be the set of context variables and M their number. $\mathbf{r_t} \in [0,1]^K$ is the vector of bounded rewards and $\mathbf{x_t} \in \{0,1\}^M$ the binary context vector at iteration t. $D_{x,r}$ is the joint distribution on $(\mathbf{x}, \mathbf{r})$. Let $\pi : \{0,1\}^M \rightarrow A$ be a certain policy and Π bet the set of policies. The objective of contextual MAB algorithms is to find the optimal policy π^*, that is the policy that maximizes the expected gain with respect to the distribution $D_{x,r}$:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \mathbb{E}_{D_{x,r}}[r_{\pi(\mathbf{x}_t)}] \tag{6.1}$$

The learning agent learns the optimal policy in a sequential manner as explained in the following:

Algorithm 8 Contextual MAB

for each iteration t
- Agent receives context information $\mathbf{x_t}$
- Agent plays arm c_t according to policy $\pi(\mathbf{x_t})$
- Reward r_{t,c_t} is perceived according to $D_{x,r}$
- Agent updates policy π_t

Note that in order to find the optimal policy, the algorithm has to explore the set of sub-optimal policies. The agent is faced once more with the exploration-exploitation dilemma. In fact, the algorithm identifies the optimal policy based on the knowledge it has about the context-reward distribution $D_{x,r}$. To build this knowledge, the agent has to interact with the environment by exploring different actions and policies. The more the agent explores, the more the knowledge it has about $D_{x,r}$ is reliable, and so is its optimal policy. However, exploration leads to sub-optimal decisions of the agent, and hence sub-optimal rewards. Whence the necessity to equilibrate exploration and exploitation.

A good MAB algorithm should therefore be able to find the optimal policy while minimizing the expected cumulated perceived regret. defined as:

$$\mathbb{E}_{D_{x,r}}[R_n] = \sum_{t=0}^{n} \mathbb{E}_{D_{x,r}}[r_{t,c^*{}_t} - r_{t,c_t}] \tag{6.2}$$

R_n is the cumulated regret after n plays and $c_t^* = \pi^*(\mathbf{x_t})$ is the action chosen by the optimal policy.

6.3.2 Random Forest for the Contextual MAB

The BF algorithm is based on decision trees. A decision tree can be seen as a combination of rules, where only one rule is selected for a given input vector, which is in this case the context vector. Finding the optimal tree structure is NP-hard [65]. Instead, a greedy approach can be used to grow the decision tree, based on decision stumps (a decision stump is a one node decision tree) [118, 119].

A decision stump takes decisions based on the observation of one context variable. To maximize the perceived reward, the decision stump should identify the best context variable. That is the variable that maximizes, when observed, the expected reward of the best action for each of its values. After identifying the best context variable, the decision stump identifies the best action while observing the best context variable. Therefore, a decision stump takes its decisions based on the observation of one context variable. In the BF algorithm, decision trees are grown by recursively stacking decision stumps, which means that a decision tree takes its decisions based on the observation of a subset of the best context variables, which is a stronger learning model than the decision stump. The random forest in its turn takes its decisions based on the vote of a number of random decision trees. In fact, the decision tree is grown through a greedy approach, by stacking greedy decision stumps. The random forest improves the model by adding additional randomness to the model, while growing the trees. That is, instead of searching for the most important context variables while splitting a node of a tree, it searches for the best context variable among a random subset of variables. This results in a wide diversity that generally results in a better model and improves the performances of the decisions [120]. The BF algorithm bases its decisions on the output of the random forest.

The decision stump is built in two sub-routines: Variable Selection followed by Action Selection.

6.3.2.1 Variable Selection

We consider the following variables:

- $\mu_c^i|\nu$ is the expected reward of arm c conditioned to the observation of variable x_i with value ν.
- $P(x_i = \nu)$ is the probability of observing variable $x_i = \nu$.
- $\mu_{c,\nu}^i = P(x_i = \nu).\mu_c^i|\nu = \mathbb{E}_{D_{x,r}}[r_c.\mathbf{1}_{x_i=\nu}]$ is the expected reward of arm c and observed value ν of context variable x_i.
- $\mu^i = \sum_{\nu\in\{0,1\}} P(x_i = \nu).\max_c {\mu_c}^i|\nu = \sum_{\nu\in\{0,1\}} \max_c \mu_{c,\nu}^i$ is the expected reward when selecting the best action while observing variable x_i.
- $i^* = \underset{i\in V}{\operatorname{argmax}}\mu^i$ is the best context variable for the decision stump.

At each time step, the Variable Selection sub-routine receives the context vector $\mathbf{x_t}$, plays an action c according to a round-robin strategy and updates its estimates. When all the actions have been played, the algorithm identities an estimated best variable i' and eliminates irrelevant variables according to the following criterion:

$\forall i \in V$, if:

$$\hat{\mu}^{i'} - \hat{\mu}^{i} + \epsilon \geqslant 4.\sqrt{\frac{1}{2.t_c} log \frac{4KM{t_c}^2}{\delta}} \tag{6.3}$$

then $V = V \backslash \{i\}$

The process is repeated until $|V| = 1$. Where:

- $\hat{\mu}^i$ is the algorithm's estimation of μ^i.
- $i' = \underset{i \in V}{\operatorname{argmax}} \hat{\mu}^i$ is the best estimated variable.
- t_c is the number of times action c was picked.
- ϵ is a parameter that tunes the optimality of the algorithm.
- $\delta \in (0, 1)$ is the failure probability.

The sub-routine outputs an ϵ-approximation of the best variable with a failure probability δ [56]. In [65], the authors show that the Variable Selection sub-routine is optimal up to logarithmic factors. In other words, on the one hand the sample complexity of Variable Selection needed to select the optimal variable i^* with failure probability δ is:

$$t^* = \frac{64K}{\Delta^2} log \frac{4KM}{\delta\Delta} \tag{6.4}$$

where $\Delta = min_{i \neq i^*}(\mu^{i^*} - \mu^i)$.
On the other hand, there exists a distribution $D_{x,r}$ such that any algorithm finding the optimal variable i^* has a sample complexity at least:

$$\Omega(\frac{K}{\Delta^2} log \frac{1}{\delta})$$

Equation 6.4 shows that the sample complexity for selecting the optimal variable is of order $O(logM)$, meaning that the algorithm is able to process large numbers of context variables, without impacting much the sample complexity.

6.3.2.2 Action Selection

Once the optimal variable identified, the decision stump has to chose the optimal action, depending on the value of the picked variable. Similarly to the variable selection, the action selection sub-routine will output an ϵ-approximation of the best arm with a failure probability δ. At each time step, the context information is received. The actions are picked according to round-robin strategy and the estimates are updated. When all the actions have already been picked, irrelevant actions are eliminated successively according to the following criterion:

$\forall c \in A$, if:

$$\hat{\mu}_{c'}^{i^*}|\nu - \hat{\mu}_{c}^{i^*}|\nu + \epsilon \geqslant 2.\sqrt{\frac{1}{2.t_{i^*,\nu,c}} log \frac{4Kt_{i^*,\nu,c}{}^2}{\delta}} \forall \nu \in \{0,1\} \tag{6.5}$$

then $A = A \backslash \{c\}$.

The process is repeated until $|A| = 1$. Where

- i^* is the identified best variable by the Variable Selection sub-routine.
- $t_{i^*,\nu,c}$ is the number of times when $x_{i^*}(t) = \nu$ and action c was picked.
- $\hat{\mu}_{c}^{i^*}|\nu$ is the estimation of $\mu_{c}^{i^*}|\nu$.
- $c' = \underset{c \in A}{\operatorname{argmax}} \hat{\mu}_{c}^{i^*}|\nu$ is the best estimated action for each value of ν (which are only 0 or 1 in our case since we are considering binary context variables).

[65] shows that the action selection is optimal up to a logarithmic factor:
The sample complexity of Action Selection necessary to find an optimal arm c^* with failure portability δ is:

$$t^* = \frac{64K}{\Delta'^2} log \frac{4K}{\delta \Delta'} \tag{6.6}$$

where $\Delta' = min_{c \neq c^*}(\mu_{c^*}^{i^*}|\nu - \mu_{c}^{i^*}|\nu)$.
And there exists a distribution $D_{x,r}$ such that any algorithm finding the optimal action c^* has a sample complexity at least:

$$\Omega(\frac{K}{\Delta^{2'}} log \frac{1}{\delta})$$

6.3.2.3 BF Algorithm

As stated previously, in the BF algorithm, the trees are grown by stacking decision stumps. This allows each tree to take its decisions based on a subset of contextual variables instead of only one, as it is the case with decision stumps. Note that deeper trees allow the algorithm to observe more context variables, and to adapt its policies accordingly. This improves the performances of the tree decisions but leads to slower convergence. Shallow trees on the other hand reduce the number of observed context variables under the risk of reducing the performances of the decisions but converge faster. Tuning the depth of decision trees to equilibrate sample complexity with the performance depends on the considered use case. Theoretical studies to optimize the trees' depth was not conducted in this framework. Therefore, in this work, the depth of the trees is tuned manually: after several test runs, a tree depth equal to 20% of the total number of context variables seemed to be the most adapted to our use case study. Indeed, the tree depth depends to a great degree on the nature of the studied data set and on the proportion of context variables that impact the most the reward feedback and that are hence the most relevant variables. Tree depth is hence a parameter that should be tuned according to the considered case study.

The BF algorithm grows a random forest of decision trees as described in the previous section. The algorithm takes its actions based on the combination of the decisions of the trees, i.e. each tree votes for an action and the decision is based on the most voted action. The pseudo-code of the BF algorithm is presented in algorithm 9, where RR is the Round-Robin function. d_θ and D_θ are respectively the tree current depth and the max tree depth of a tree θ. F is the number of trees

(the decision forest is constituted by a set of F decision trees: $\theta_1, \theta_2, ..., \theta_F$). p_θ is a path in tree θ, selected according to the observed context $\mathbf{x_t}$. Each path has its own set of context variables V_{p_θ} and set of actions A_{p_θ}.

NewPath function allocates and stacks a new decision stump to the tree path p_θ given as input, along with its set of remaining context variables. f is a random function that parametrizes V_{p_θ} as described in algorithm 10, in order to randomize the trees.

VE Function Refers to the Variable Elimination process described in algorithm 11. VE process is based on the Variable Selection sub-routine described in section 6.3.2.1. VE is carried out at each iteration, for each observed path p_θ of each tree θ. The elimination of variables is done based on a criteria similar to equation 6.3. Counters and estimators should be however defined for each selected path, that is:

- replace $\hat{\mu}^i$ with $\hat{\mu}^i|p_\theta$
- replace t_c with $t_{p_\theta,c}$

where $\hat{\mu}^i|p_\theta$ is the estimate of the expected reward when selecting the best action and observing variable x_i and path p_θ, $t_{p_\theta,c}$ is the number of times the path p_θ and action c have been observed. VE takes as input parameters the necessary variables and counters to evaluate the previously described estimators and returns S_{p_θ}, the set of remaining context variables for path p_θ.

AE Functions Refers to the Action Elimination processes described in algorithm 12. AE process is based on the Action Selection sub-routine described in section 6.3.2.2. Similarly to VE, AE process is carried out at each iteration, for each observed path p_θ of each tree θ, once the current depth of the path reaches the max depth of the tree, that is when $d_{p_\theta} = D_\theta$. The action elimination is based on equation 6.5. Counters and estimators should be also defined for each selected path:

- replace $\hat{\mu}_c^{i^*}|\nu$ with $\hat{\mu}_c|p_\theta$
- replace $t_{i,\nu,c}$ with $t_{p_\theta,i,\nu,c}$

where $\hat{\mu}_c|p_\theta$ is the estimate of the expected reward of arm c conditioned to the observation of variable path p_θ. $t_{p_\theta,i,\nu,c}$ is the number of times action c has been played when the path p_θ and the value ν of variable i were observed. AE takes as input parameters the necessary variables and counters to evaluate the previously described estimators, and returns A_{p_θ}, the set of remaining actions for path p_θ.

The BF algorithm was proposed and analyzed in [65]. The authors studied the optimality of the algorithm. They showed that the sample complexity needed to obtain the optimal random forest of size F with failure probability δ is:

$$t^* = 2^D(\frac{64K}{\Delta_1^2}log(\frac{4KMDF}{\delta\Delta_1}) + \frac{64K}{\Delta_2^2}log(\frac{4KF}{\delta\Delta_2})) \tag{6.7}$$

where $\Delta_1 = min_{\theta,p_\theta,i\neq i'}P(p_\theta)(\mu^{i'}|p_\theta - \mu^i|p_\theta)$ and $\Delta_2 = min_{\theta,p_\theta,k\neq k'}P(p_\theta)(\mu_{k'}|p_\theta - \mu_k|p_\theta)$. $\mu^i|p_\theta$ and $\mu_k|p_\theta$ are defined in section 5.3.2, $k' = \underset{k\in A}{\text{argmax}}\mu_k|p_\theta$ and $i' = \underset{i\in V}{\text{argmax}}\mu^i|p_\theta$, $D = \max_\theta D_\theta$ and $P(p_\theta)$ is the probability of observing path p_θ.

They have also proven a lower bound, by showing that there exists a distribution $D_{x,r}$ such that any algorithm finding the optimal forest of size F has a sample complexity of at least:

$$\Omega(2^D[\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2}]Klog(\frac{1}{\delta})) \tag{6.8}$$

Proving hence that the BF algorithm is optimal up to logarithmic factors. Besides, equation 6.7 shows that the sample complexity of the BF algorithm depends logarithmically on the number of context variables (the dependence of of order $O(log(M))$), whereas in linear contextual bandits, such as LinUCB, it is of order $O(\sqrt{(M)})$. This means that the BF algorithm scales well with the number of context variables. However, BF's sample complexity dependence of the depth of the decision trees D is exponential. This explains the trade-off between the trees depth and consequently the optimality of the output policy, and the convergence speed, as explained previously in this chapter (section 6.3.2.3).

In the next section we describe the system model and evaluate the performances of the proposed approach and compare them with a C-PBSM approach based on stochastic MAB.

Algorithm 9 BF MAB Algorithm

$d_\theta = 0, p_\theta = ()$
$NewPath(p_\theta, V)\ \forall$ tree θ
for each iteration $t = 0, 1, 2...$
- Receive context information $\mathbf{x_t}$
- for each tree θ
- - select context path p_θ depending on $\mathbf{x_t}$
- - if $d_{p_\theta} < D_\theta$ then
- - - $c_{p_\theta} = RR(A)$
- - else if $d_{p_\theta} = D_\theta$ and $|A_{p_\theta}| > 1$ then
- - - $c_{p_\theta} = RR(A_{p_\theta})$
- - else if $d_{p_\theta} = D_\theta$ and $|A_{p_\theta}| = 1$ then
- - - $c_{p_\theta} = a_{p_\theta}$
- end for
- Apply action $a = \underset{k \in A}{\operatorname{argmax}} \sum_{i=1}^{F} \mathbf{1}_{c_{p_\theta}=k}$
- Receive reward $r_{t,a}$
- for each tree θ
- - $t_{p_\theta,a} = t_{p_\theta,a} + 1$
- - $S_{p_\theta} = VE(t_{p_\theta,a}, a, \mathbf{x_t}, r_{t,a}, S_{p_\theta}, A, \theta, p_\theta, \epsilon, \delta)$
- - if $|S_{p_\theta}| = 1$ and $d_{p_\theta} < D_\theta$ then
- - - $V_{p_\theta} = V_{p_\theta} \backslash \{i'\}$ where $S_{p_\theta} = \{i'\}$ (i' is the remaining variable in S_{p_θ})
- - - $NewPath(p_\theta + (x_{i'} = 0), V_{p_\theta})$
- - - $NewPath(p_\theta + (x_{i'} = 1), V_{p_\theta})$
- - else if $|S_{p_\theta}| = 1$ and $d_{p_\theta} = D_\theta$ then
- - - $t_{p_\theta,i,\nu,a} = t_{p_\theta,i,\nu,a} + 1$
- - - $A_{p_\theta} = AE(t_{p_\theta,i,\nu,a}, a, \mathbf{x_t}, r_{t,a}, A_{p_\theta}, \theta, p_\theta, \epsilon, \delta)$
- - - if $|A_{p_\theta}| = 1$
- - - - $a_{p_\theta} = k$ where $A_{p_\theta} = \{k\}$ (k is the last remaining action in A_{p_θ})
- - - end if
- - end if
- end for
end for

Algorithm 10 NewPath

Function $NewPath(p_\theta, V_{p_\theta})$:
- $S_{p_\theta} = \{i \in V_{p_\theta} : f(\theta, p_\theta, i) = 1\}$
- $d_\theta = d_\theta + 1$
- $A_{p_\theta} = A$
- $t_{p_\theta,c} = 0 \; \forall c \in A$
- $\hat{\mu}^i|p_\theta = 0 \; \forall i \in V$
- $\hat{\mu}^i_c|p_\theta = 0 \; \forall i \in V, \forall c \in A$

Algorithm 11 VE Function

Function $VE(t, a, \mathbf{x_t}, r_{t,a}, S_{p_\theta}, A, \theta, p_\theta, \epsilon, \delta)$:
for each variable $i \in S_{p_\theta}$
- $\hat{\mu}^i_{a,0}|p_\theta = \frac{r_{t,a}}{t}.\mathbf{1}_{x_i=0} + \frac{t-1}{t}.\hat{\mu}^i_{a,0}|p_\theta$
- $\hat{\mu}^i_{a,1}|p_\theta = \frac{r_{t,a}}{t}.\mathbf{1}_{x_i=1} + \frac{t-1}{t}.\hat{\mu}^i_{a,1}|p_\theta$
- $\hat{\mu}^i|p_\theta = \sum_{\nu\in\{0,1\}} \max_a \hat{\mu}^i_{a,\nu}|p_\theta$

end for
$i' = \underset{i\in S_{p_\theta}}{\operatorname{argmax}} \hat{\mu}^i|p_\theta$
if $a = LastAction(A)$ then
- for each $i \in S_{p_\theta}$
- - if $\hat{\mu}^{i'}|p_\theta - \hat{\mu}^i|p_\theta + \epsilon \geqslant 4\sqrt{\frac{1}{2t} log \frac{4KMD_\theta Ft^2}{\delta}}$ then
- - - $S_{p_\theta} = S_{p_\theta} \backslash \{i\}$
- - end if
- end for

end if
return S_{p_θ}

Algorithm 12 AE Function

Function $AE(t, a, \mathbf{x_t}, r_{t,a}, A_{p_\theta}, \theta, p_\theta, \epsilon, \delta)$:
$\hat{\mu}_a|p_\theta = \frac{r_{t,a}}{t} + \frac{t-1}{t}.\hat{\mu}_a|p_\theta$
$a' = \underset{k\in A_{p_\theta}}{\operatorname{argmax}} \hat{\mu}_k|p_\theta$
if $a = LastAction(A_{p_\theta})$ then
- for each $k \in A_{p_\theta}$
- - if $\hat{\mu}_{a'}|p_\theta - \hat{\mu}_k|p_\theta + \epsilon \geqslant 2\sqrt{\frac{1}{2t} log \frac{4KFt^2}{\delta}}$ then
- - - $A_{p_\theta} = A_{p_\theta} \backslash \{k\}$
- - end if
- end for

end if
return A_{p_θ}

6.4 Scenario Description

Consider a set Λ of network clusters distributed in different locations of the network, where the learning process will take place. In each cluster $\lambda \in \Lambda$, we consider a set of deployed SON functions S_λ. N_s^λ is the number of instances of SON function s in sector λ ($s \in S_\lambda$). $N_s^\lambda = 0$ if $s \notin S_\lambda$. Each SON function has a set of SCV sets denoted as $V_s, \forall s \in S_\lambda, \forall \lambda \in \Lambda$. The set of possible SCV sets combination in a network cluster λ is then defined as:

$$C_\lambda = (V_{s_1})^{N_{s_1}^\lambda} \times (V_{s_2})^{N_{s_2}^\lambda} \times ... \times (V_{s_{|S|}})^{N_{s_{|S|}}^\lambda}$$

where $s_1, s_2, ..., s_{|S|} \in S_\lambda$. The reward perceived by the learning agent for an action c_t at iteration t is considered as a linear combination of perceived KPIs and weights:

$$r_{t,c_t} = \sum_{i=1} \omega_i . K_{i,t,c_t} \tag{6.9}$$

where K_{i,t,c_t} is the value of the i^{th} KPI when action c_t is taken, at iteration t, and ω_i are weights set by the operator and reflecting its priority to maximize the corresponding KPI target.

Note that we still consider that $\forall c \in C_\lambda$, all the SON functions converge after a sufficient time, $\forall \lambda \in \Lambda$. At each iteration t, each cluster λ reports to the AI Agent a numerical reward value, depending on the reward definition in equation 6.9, and a feature vector $\mathbf{x_t^\lambda}$ carrying the context information at iteration t of cluster λ as described in figure 6.2. Note that the feature vector depends on both the network cluster λ and the iteration t. In fact, the feature vector carries information about both the topological and technological aspects of the access network in each cluster (e.g. environment, technology, information about the vendor's hardware, multi-layer or not, etc) and time dependent network information (traffic information, types of services, special event etc). At each reporting of the clusters, the AI agent updates the knowledge database, and outputs new actions to be applied in the clusters. In our case, as the AI agent runs the BF algorithm, the knowledge database stores the binary decision trees.

For this scenario, we consider four network clusters for the learning process. Each cluster is composed of a macro cell (that we refer to as central macro cell) and the first tier neighboring macro cells as represented in figure 6.3. Small cells can be deployed in the central macro cell's coverage region, to serve an eventual traffic hotspot inside the coverage region of the macro cell. In this case the macro cell and small cell are referred to as master cell and slave cell respectively. We consider 3 SON functions in each cluster λ, each deployed in several instances according to the following:

- **MLB:** Deployed on each macro cell.
- **CRE:** Deployed on each small cell.
- **eICIC:** Deployed on each macro cell containing small cells in its coverage region.

We further consider that the AI Agent can also directly act on certain network parameters. For example in this scenario we consider that it can turn on or off small cells through a sleep mode process in order to optimize the energy efficiency in the network. Depending on the observed context and the operator's objectives, the AI agent takes the decision to either activate or deactivate the small cells. When a small cell enters sleep mode, the UEs served by the small cell are handed over to the master macro cell.

The following KPIs are considered to be the most relevant in our scenario: the load variance in the cluster, the average user throughput, the average throughput of users in the edge of small cells,

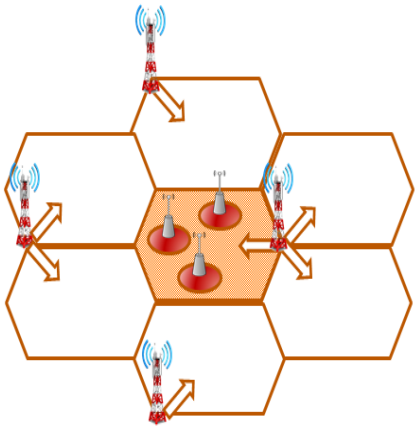


Figure 6.3: Example of a Network Learning Cluster

the energy efficiency and the load variance in the central macro cell and its slave small cells. For each iteration t and action c_t (SCV sets combination), the KPIs in a network cluster λ are defined as:

- l_{i,t,c_t} is the load of cell i in network cluster λ.
- $\bar{l}_{t,c_t}$ is the average load in the considered section.
- $\sigma_{t,c_t} = \frac{\sum_{i=0}^{|\lambda|} l_{i,t,c_t} - \bar{l}_{t,c_t})^2}{|\lambda|}$ the load variance in cluster λ. $|\lambda|$ is the number of cells in cluster λ.
- $\overline{T}_{t,c_t}$ is the average user throughput in the central macro cell.
- $\overline{T'}_{t,c_t}$ is the average small cell edge user throughput in the central macro cell coverage area.
- $\overline{P}_{t,c_t}$ is the power consumption of the small cells.
- σ'_{t,c_t} is the average load variance in the central macro cell and its slave small cells.

The vector of features describing the context reports to the AI agent information about: Macro inter-site distance (1732m for rural and 500m for urban areas according to 3GPP standards [94]), traffic status in the central cell, traffic status in each of the neighboring cells, traffic status in the small cells of the central cell, how many cell layers are there in the cluster (0 if homogeneous macro deployment, 1 if heterogeneous two layer network). We consider that the traffic status in the cells can be in four different states (Low, Medium, High, Very High). More precision can be considered depending on the use case. All categorical features are coded into disjunctive binary variables.

6.5 Simulation Settings and Results

In this subsection we present the simulation settings and results of the previously described scenario. We compare the performances of the contextual BF algorithm with a C-PBSM based on a

Parameters	Settings
Bandwidth	10 MHz
PRBs per eNB	50
Carrier Frequency	2 GHz
Macro ISD	1732 m
Macro Path Loss to UE	$128.1 + 37.6 \times log_{10}(d[Km])$
Pico Path Loss to UE	$140.1 + 37.6 \times log_{10}(d[Km])$
Antenna Gain	macro: 14 dBi; pico: 5 dBi
Transmit Power	macro: 46 dBm; pico: 30 dBm
Shadowing Standard Deviation	macro: 8 dBm; pico 10 dBm
Traffic model	Dowlink File Transfer Protocol (FTP), file size: 14 Mbits
RL Simulation Time Iteration	20 min (in simulation time)

Table 6.1: Simulation parameters

stochastic MAB algorithm, namely the Successive Elimination (SE) algorithm [57]. The SE algorithm corresponds roughly to the Action Selection sub-routine. That is, it successively eliminates actions according to the criterion in equation 6.5 until converging to a single action. Note that we also consider that the stochastic algorithm is learning simultaneously over the 4 considered training clusters. We discuss the algorithms optimality and convergence in the following.

For the simulation settings, we consider that all the clusters are in urban areas and are heterogeneous two layer networks. Simulation details are detailed in table 6.1. We consider daily traffic changes in the different cells of the clusters. As stated before, the traffic state of each cell can be in four different categories and is considered to be known by the AI agent. One can consider that this information is communicated to the agent based on known traffic profiles. We run the learning process for different operator objectives. We also consider that the same set of SON functions is deployed in all the clusters, and consequently all the clusters have the same action space. The SCV sets of the considered SON functions are presented in table 6.2 (SON functions are detailed in chapter 3). For each SON function, we configure all the instances in the same learning cluster with the same SCV sets. When the small cells are in sleep mode, CRE and eICIC functions are obviously off. This leaves us with a total number of actions of 36. The possible SCV sets combinations of each action are depicted in table 6.3.

The perceived reward evolution is plotted in figure 6.4. The stochastic MAB bandit is not able to observe the context changes. It estimates the average perceived reward of each arm regardless of the context changes. It identifies hence an action, that has the highest empirical average over all the contexts. The optimal policy identified by the stochastic MAB is thus a single action policy for all the contexts. The contextual BF algorithm on the other hand constructs its policy by observing the context. Eventually, the BF algorithm identifies an optimal action for each of the observed contexts, performing better than the single action policy of the stochastic MAB (figure 6.4). In figure 6.5 we plot the average perceived reward per contexts for a set of observed contexts in this scenario, for the previously considered objectives. We can see that the contextual BF algorithm performs always at least as good as the stochastic MAB. The average perceived reward of both algorithms is similar for some contexts, because the stochastic policy picks the action generating the highest average reward in global, without taking into consideration context changes. However, in other contexts, due to its ability to observe and adapt with the context variations, the BF algorithm's policy generates a higher average perceived reward than the stochastic MAB. Practically, the AI agent adapts the SCV sets of the SON functions, according to the observed context. This makes the contextual MAB more suitable for scenarios with context changes, with

different optimal actions for different contexts. Which is the case in real networks. The context analysis for other operator objectives is similar to this example.

The BF algorithm appears to be slower than the stochastic bandit in finding the optimal policy (figure 6.4). This can be explained by the fact that the sample complexity of the SE algorithm is bounded by:

$$O((\frac{K}{\Delta_0^2})log(\frac{K}{\delta\Delta_0})) \tag{6.10}$$

And it is optimal up to a logarithmic factor as shown in [56], where Δ_0 is the expected reward difference between the best arm and the second best arm. The sample complexity of the stochastic SE algorithm and the sample complexity of the BF algorithm (equation 6.7) are of the same order up to logarithmic factors, except for the exponential dependence of the BF on the trees depth D. This means that the price to be paid for better policies, through context observations, is a slower convergence compared to a stochastic approach. However, this slow convergence compared to the stochastic algorithm should not be a concern in practical cases, because as stated previously, the AI agent can learn simultaneously from different learning clusters in the network, and constructs a common knowledge database for all of them. If we consider that all the clusters take the same amount of time Δ_T to send their feedback, then the time needed to reach the sample complexity t^* with $|\Lambda|$ learning clusters is $T^* = \frac{\Delta_T.t^*}{|\Lambda|}$. Meaning that having a large number of learning clusters will reduce the learning time, making hence possible the deployment of such learning processes on real network.

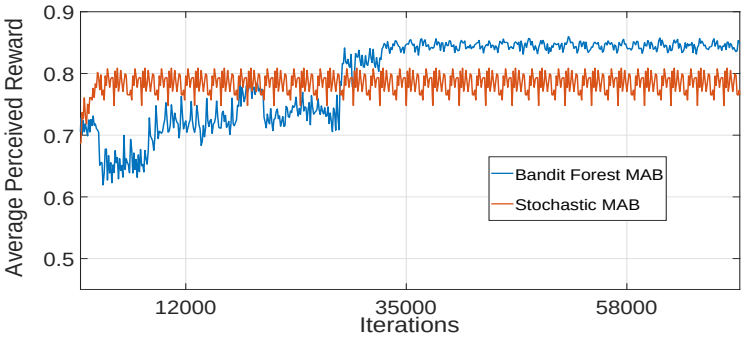
Finally, similarly to the distributed Q-learning, the BF's adaptability with the operator objective changes is not straightforward as in the stochastic UCB1 and LinUCB MAB framework. The reason is that in the BF framework, the decision trees are grown by choosing the best context variables with respect to the reward function. Thus, when the operator changes its objectives, the decision trees have to be reset and grown all over again while considering the new reward function as a criteria to pick the best context variables. A possible solution would be to memorize logs and estimates of each KPI target of each context variable for each tested action. This information can consequently be exploited offline to accelerate growing the trees and hence improve the BF algorithm's adaptability with the objective changes. This approach needs however further investigation and studies.

SON	**SCV sets**
MLB	Off: Function is turned off SCV1: Designed for mid loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.55; l_{H,i} = 0.60; \Delta_i = 2dB$) SCV2: Designed for high loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.75; l_{H,i} = 0.80; \Delta_i = 2dB$) SCV3: Designed for very high loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.80; l_{H,i} = 0.90; \Delta_i = 2dB$)
CRE	Off: Function is turned off SCV1: Designed for mid loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.55; l_{H,i} = 0.55; \Delta_i = 2dB$) SCV2: Designed for high loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.70; l_{H,i} = 0.70; \Delta_i = 2dB$) SCV3: Designed for very high loaded networks ($\Phi_{min,i} = 0dB; \Phi_{\max,i} = 16dB; l_{L,i} = 0.80; l_{H,i} = 0.80; \Delta_i = 2dB$)
eICIC	Off: Function is turned off SCV1: Function is turned on ($\kappa_{\max,i} = 8ABS; R_{L,i} = 1; R_{H,i} = 2; \Delta_i = 1ABS$)

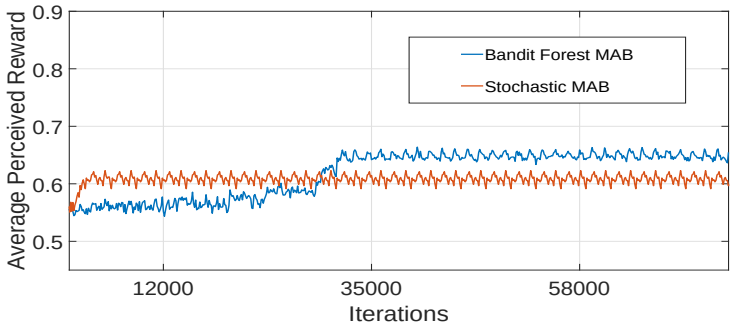
Table 6.2: SCV Sets Description

	SCV Sets Combinations			
Action Index	MLB	CRE	eICIC	Sleep
1	Off	Off	Off	Off
2	Off	Off	SCV1	Off
3	Off	SCV1	Off	Off
4	Off	SCV1	SCV1	Off
5	Off	SCV2	Off	Off
6	Off	SCV2	SCV1	Off
7	Off	SCV3	Off	Off
8	Off	SCV3	SCV1	Off
9	SCV1	Off	Off	Off
10	SCV1	Off	SCV1	Off
11	SCV1	SCV1	Off	Off
12	SCV1	SCV1	SCV1	Off
13	SCV1	SCV2	Off	Off
14	SCV1	SCV2	SCV1	Off
15	SCV1	SCV3	Off	Off
16	SCV1	SCV3	SCV1	Off
17	SCV2	Off	Off	Off
18	SCV2	Off	SCV1	Off
19	SCV2	SCV1	Off	Off
20	SCV2	SCV1	SCV1	Off
21	SCV2	SCV2	Off	Off
22	SCV2	SCV2	SCV1	Off
23	SCV2	SCV3	Off	Off
24	SCV2	SCV3	SCV1	Off
25	SCV3	Off	Off	Off
26	SCV3	Off	SCV1	Off
27	SCV3	SCV1	Off	Off
28	SCV3	SCV1	SCV1	Off
29	SCV3	SCV2	Off	Off
30	SCV3	SCV2	SCV1	Off
31	SCV3	SCV3	Off	Off
32	SCV3	SCV3	SCV1	Off
33	Off	Off	Off	On
34	SCV1	Off	Off	On
35	SCV2	Off	Off	On
36	SCV3	Off	Off	On

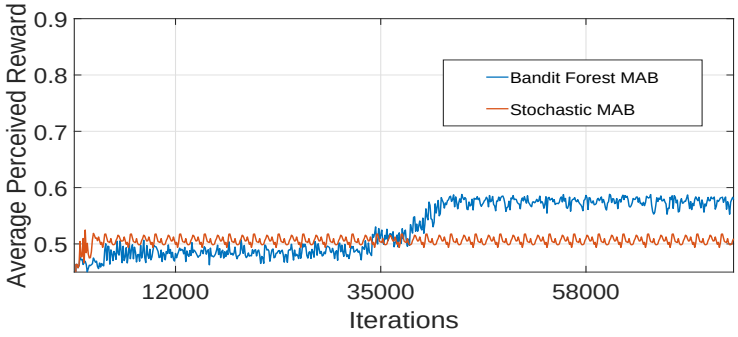
Table 6.3: SCV sets behavior description



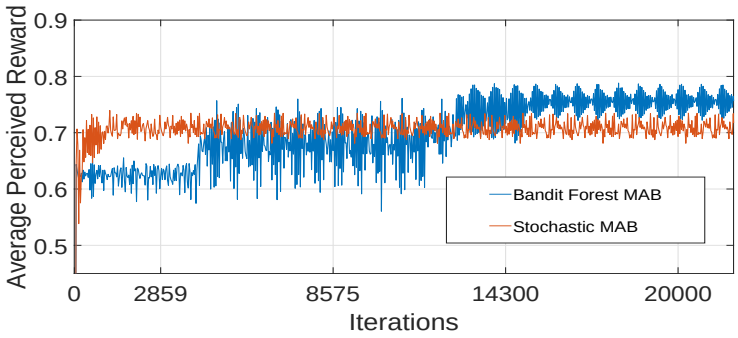
(a) w1=1



(b) w1=w2=0.5

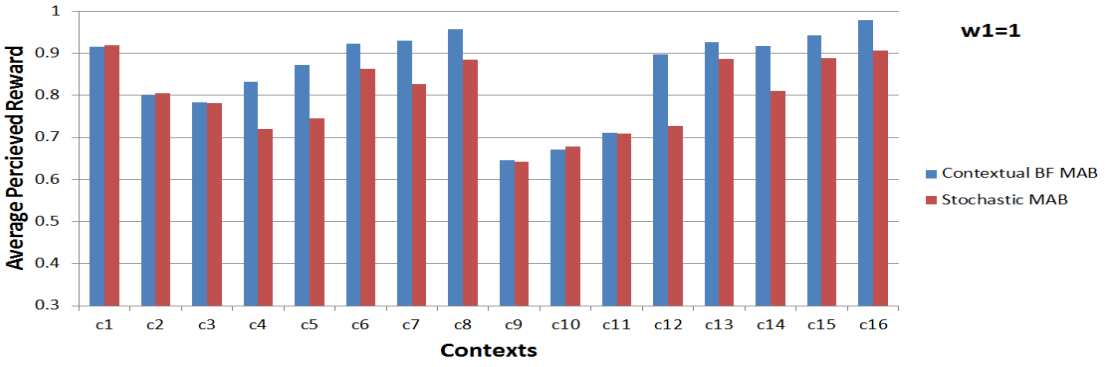


(c) w1=w2=w3=0.333

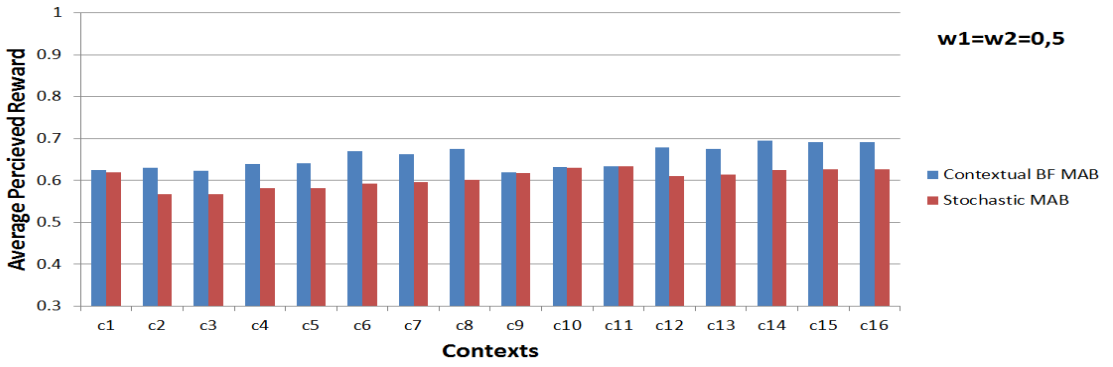


(d) w4=0.25; w5=0.75

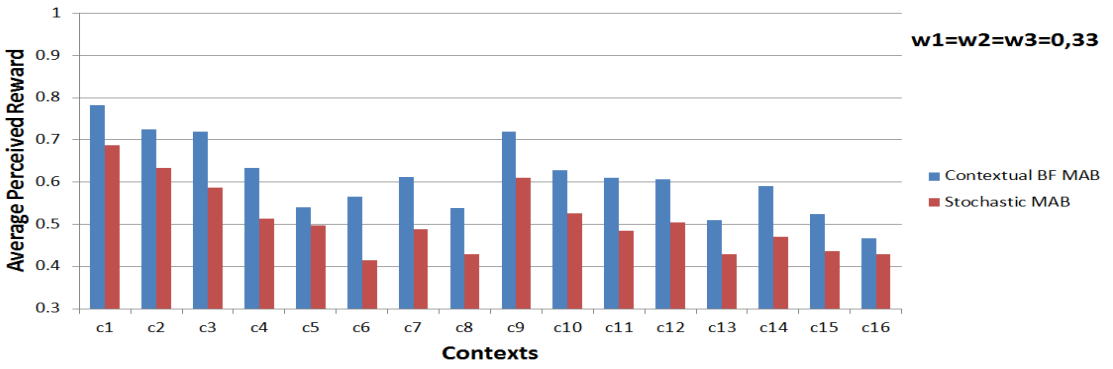
Figure 6.4: Perceived Rewards Comparison



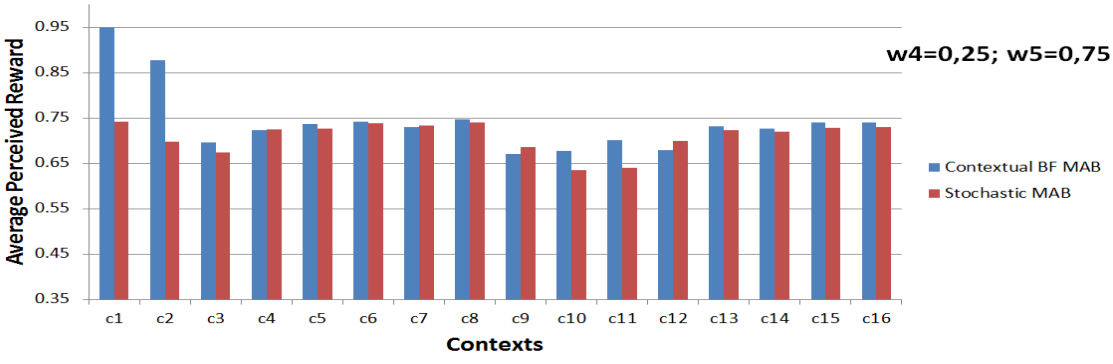
(a) w1=1



(b) w1=w2=0.5



(c) w1=w2=w3=0.333



(d) w4=0.25; w5=0.75

Figure 6.5: Average Perceived Rewards per Observed Context

Chapter 7

Conclusion and Perspectives

Today's networks are SON enabled networks, where several SON functions are deployed and insure the automation of operational tasks in the network such as load balancing, energy saving, interference mitigation, etc. However, because of the increasing complexity of future networks, a higher level of automation is required to efficiently manage such networks. In fact, SON functions need themselves to be efficiently configured and managed, depending on the operator's high level objectives and on the network context where they are deployed. In this thesis we have proposed and studied a Cognitive Policy Based SON Management framework, i.e. C-PBSM, that insures an intelligent and autonomic SON management in radio networks. The objective of the C-PBSM is to autonomously and efficiently translate high level operator objectives, formulated as KPI targets, into SON configurations in the network. In this work we propose to implement the cognitive loop of the C-PBSM through RL. The C-PBSM is hence able to learn the optimal SON configurations by interacting with the real network. We start by proposing an approach based on stochastic MAB, i.e. UCB1. This centralized approach showed to be able to learn the optimal SON configuration over a network section, maximizing hence a reward function that reflects the operator KPI targets. It is also able to quickly adapt with the operator objective changes. We then propose to enhance the convergence speed of this approach using LinUCB, which is a stochastic MAB that assumes a linear relation between the average perceived reward and the action features. This approach however lacks scalability when the network section size increases. To leverage this problem we propose a distributed approach based on distributed Q-learning. We show that it is scalable and able to efficiently explore large action sets. In this framework we also study a flexible deployment of learning agents, based on an SDN approach. In these approaches, the traffic stationarity is necessary for an efficient learning. It is however not the case in real networks, where the traffic is dynamic, depending on the time and the location. Hence, we finally propose and study a context aware C-PBSM based on a contextual MAB. This approach consists of a centralized agent learning simultaneously over different network clusters with different network contexts and traffic profiles. The contextual C-PBSM is thus able to learn over different network contexts, adapt to traffic variations, transfer its knowledge and policies to different untrained sections of the network and improve its convergence speed through simultaneous learning over different clusters and through collaborative learning.

This work is indeed an important step towards networks that are self-organized as a whole. It also opens perspectives to a number of research axes, to further improve network management, operations and automation. In the following we present the potential future work in this perspective. In fact, in this work we consider that the C-PBSM learns through RL from scratch. It build its knowledge only by interacting with the network. However, operators possess historic databases about network configurations and their impact as well as human expertise and skills in network

operation. These can be of use so that the C-PBSM does not learn from scratch, but instead exploit the operator's knowledge in order to accelerate its learning process. Furthermore, in this work, we consider that the operator objectives are expressed as target KPIs, that are manually determined according to the high level semantic objectives of the operator. Nevertheless, there is a possibility to use advanced ML techniques to translate the semantic objectives into technical KPI targets. This would be a step further towards the intend based network management paradigm [14]. Also, as stated previously in this work, an efficient cell clustering can drastically improve the scalability of the C-PBSM. Further research should hence be conducted to exploit real network metrics, as well as ML tools, to output relevant clusters of similar cells in terms of network context, behavior, and response to network configurations. On another hand, 5G networks will bring major changes in network operation. In fact, 5G introduces network slicing, which represents a new paradigm in network operation. Also, 5G RAN will bring new connectivity modes such as nomadic networks and D2D for instance. New services will also be introduced, as well as many devices that are not user mobile phones such as connected cars, machines, e-health devices, etc. This will bring major changes in the traffic dynamics, that should be thoroughly investigated. Not to forget also the network visualization and Cloud-RAN concepts. SON functions will consequently have to manage physical but also virtual resources. Besides, with the mobile edge computing concept, there will be a need to determine which resources and actions should be managed and handled on the edge and which are not. Furthermore, the concept of end-to-end network management through an SDN global network orchestrator has emerged. Hence, there is a need to study how the C-PBSM will be interfaced with such an orchestrator and how they are going to interact with each other. Finally, implementation studies of a C-PBSM framework should be done, taking into consideration the operator's and the vendor's equipments and software, as well as more practical deployment constraints.

Bibliography

[1] SEMAFOUR project. http://fp7-semafour.eu/. 2015.

[2] R. Sutton and A. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[3] D. Kreutz et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[4] Cisco. Cisco visual networking index: global mobile data traffic forecast update, 2016-2021 white paper. March 2017.

[5] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology. 2001.

[6] 3GPP TS 32.521. Umts; lte; telecommunication management; son; nrm; irp; requirements (Release 8). 2013.

[7] S. Hämäläinen, H. Sanneck, and C. Sartori. Lte self-organising networks (son): network management automation for operational efficiency. 2012.

[8] Osianoh Glenn Aliu, Ali Imran, Muhammad Ali Imran, and Barry Evans. A survey of self organisation in future cellular networks. *IEEE Communications Surveys & Tutorials*, 15(1):336–361, 2013.

[9] 3GPP TS 36.902. Evolved universal terrestrial radio access network (e-utran); self-configuring and self-optimizing network (son) use cases and solutions. 2012.

[10] 4G Americas. Self-optimizing networks - the benefits of son in lte. technical report. July 2011.

[11] 4G Americas. Self-optimizing networks in 3gpp release 11: The benefits of son in lte. technical report. October 2013.

[12] 3GPP TR 37.822. Study on next generation self-optimizing network (son) for utran and e-utran. 2015.

[13] 3GPP TR 32.860. Telecommunication management; study on enhancements of operations, administration and maintenance (oam) aspects of distributed self-organizing networks (son) functions. 2016.

[14] Gartner report. *https://www.gartner.com/doc/3599617/innovation-insight-intentbased-networking-systems*, 2018.

[15] 3GPP TS 35.522. Digital cellular telecommunications system (phase 2+); umts; lte; telecommunication management; son; nrm; irp; information system. 2013.

[16] T. Bandh et al. Policy-based coordination and management of son functions. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 827–840. IEEE, 2011.

[17] H. Sanneck T. Bandh and R. Romeikat. An experimental system for son function coordination. In *Vehicular Technology Conference (VTC Spring)*, pages 1–2. IEEE, 2011.

[18] L.C. Schmelz et al. A coordination framework for self-organisation in lte networks. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 193–200. IEEE, 2011.

[19] O. Iacoboaiea et al. Coordinating son instances: A reinforcement learning framework. In *Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, 2014.

[20] O. Iacoboaiea et al. Son conflict diagnosis in heterogeneous networks. In *,26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1459–1463. IEEE, 2015.

[21] O. Iacoboaiea et al. Son coordination in heterogeneous networks: A reinforcement learning framework. *IEEE Transactions on Wireless Communications*, 15(9):5835–5847, 2016.

[22] S. Lohmuller C. Frenzel and L.C. Schmelz. Dynamic, context-specific son management driven by operator objectives. In *Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE, 2014.

[23] S. Hahn and T. Kürner. Managing and altering mobile radio networks by using son function performance models. In *11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 214–218. IEEE, 2014.

[24] S. Lohmuller C. Frenzel and L.C. Schmelz. Son management based on weighted objectives and combined son function models. In *11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 149–153. IEEE, 2014.

[25] L.C. Schmelz S. Lohmüller and S. Hahn. Adaptive son management using kpi measurements. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 625–631. IEEE, 2016.

[26] 5G-NORMA project web page https://5gnorma.5g-ppp.eu/.

[27] METIS II project web page https://metis-ii.5g-ppp.eu/.

[28] FANTASTIC-5G project web page http://fantastic5g.eu/.

[29] 3GPP TR 22.891. Study on new services and markets technology enablers. 2016.

[30] View on 5g architecture. Technical report, 5G PPP Architecture Working Group, July 2016.

[31] J. Mitola. *Cognitive radio*. PhD thesis, Royal Institute of Technology, 2000.

[32] P.M. Luengo et al. Fuzzy rule-based reinforcement learning for load balancing techniques in enterprise lte femtocells. *IEEE Trans. Vehicular Technology*, 62(5):1962–1973, 2013.

[33] S. Mwanje and A. Mitschele-Thiel. A q-learning strategy for lte mobility load balancing. In *24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 2154–2158. IEEE, 2013.

[34] T. Kudo and T. Ohtsuki. Cell range expansion using distributed q-learning in heterogeneous networks. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):61, 2013.

[35] P. Coucheney M.S. Ali and M. Coupechoux. Load balancing in heterogeneous networks based on distributed learning in near-potential games. *IEEE Transactions on Wireless Communications*, 15(7):5046–5059, 2016.

[36] M. Bennis M. Simsek and A. Czylwik. Dynamic inter-cell interference coordination in hetnets: A reinforcement learning approach. In *Global Communications Conference (GLOBECOM)*, pages 5446–5450. IEEE, 2012.

[37] T. Clarke N. Morozs and D. Grace. Distributed heuristically accelerated q-learning for robust cognitive spectrum management in lte cellular systems. *IEEE Transactions on Mobile Computing*, 15(4):817–825, 2016.

[38] A. De Domenico and D. Kténas. Reinforcement learning for interference-aware cell dtx in heterogeneous networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.

[39] M. Wildemeersch et al. Cognitive small cell networks: Energy efficiency and trade-offs. *IEEE Transactions on Communications*, 61(9):4016–4029, 2013.

[40] R.C. Qiu et al. *Cognitive radio communication and networking: Principles and practice*. John Wiley & Sons, 2012.

[41] T. Hastie J. Friedman and R. Tibshiranit. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[42] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[43] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[44] L. Jouffe. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3):338–355, 1998.

[45] G. Laurent L. Matignon and N. Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'07*, pages 64–69, 2007.

[46] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[47] M. Dirani and Z. Altman. A cooperative reinforcement learning approach for inter-cell interference coordination in ofdma cellular networks. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 170–176. IEEE, 2010.

[48] A. Galindo-Serrano and L. Giupponi. Distributed q-learning for aggregated interference control in cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 59(4):1823–1834, 2010.

[49] N. Cesa-Bianchi P. Auer and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[50] S. Bubeck et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[51] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[52] N. Korda E. Kaufmann and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[53] H. Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

[54] A. Garivier and O. Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual Conference On Learning Theory*, pages 359–376, 2011.

[55] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[56] S. Mannor E. Even-Dar and Y. Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

[57] S. Mannor E. Even-Dar and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105, 2006.

[58] V. Dani et al. Stochastic linear optimization under bandit feedback. 2008.

[59] András Antos Yasin Abbasi-Yadkori and Csaba Szepesvári. Forced-exploration based algorithms for playing in stochastic linear bandits. In *COLT Workshop on On-line Learning with Limited Feedback*. Citeseer, 2009.

[60] A. J. Mersereau et al. A structured multiarmed bandit problem and the greedy policy. *IEEE Transactions on Automatic Control*, 54(12):2787–2802, 2009.

[61] P. Rusmevichientong and J. N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

[62] W. Chu et al. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[63] A. Slivkins. Contextual bandits with similarity information. *The Journal of Machine Learning Research*, 15(1):2533–2568, 2014.

[64] D. Pál T. Lu and M. Pál. Contextual multi-armed bandits. In *Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics*, pages 485–492, 2010.

[65] T. Urvoy R. Féraud, R. Allesiardo and F. Clérot. Random forest for the contextual bandit problem. In *Artificial Intelligence and Statistics*, pages 93–101, 2016.

[66] A. Baddeley et al. *Case studies in spatial point process modeling*, volume 185. Springer, 2006.

[67] 3GPP TS 36.331. Evolved Universal Terrestrial Radio Access: Radio resource control; protocol specifications. 2012.

[68] A. Adeyemi and D. Ike. A review of load balancing techniques in 3gpp lte system. *Int. J. Comput. Sci. Eng*, 2(4):112–116, 2013.

[69] C. Yang. Concurrent mobility load balancing in lte self-organized networks. In *IEEE 21st International Conference on Telecommunications (ICT)*, pages 288–292. IEEE, 2014.

[70] A. Lobinger and S. Stefanski. Coordinating handover parameter optimization and load balancing in lte self-optimization networks. In *Vehicular Technology Conference (VTC Spring)*. IEEE, 2011.

[71] R. Nasri and Z. Altman. Handover adaptation for dynamic load balancing in 3gpp long term evolution systems. *arXiv preprint arXiv:1307.1212*, 2013.

[72] FP7 ICT-SOCRATESt. *http://www.fp7-socrates.eu/index.html*, 2012.

[73] K. Sandrasegaran A. Daeinabi and P. Ghosal. A dynamic cell range expansion scheme based on fuzzy logic system in lte-advanced heterogeneous networks. In *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pages 6–11. IEEE, 2014.

[74] P. Tian et al. Deployment analysis and optimization of macro-pico heterogeneous networks in lte-a system. In *15th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 246–250. IEEE, 2012.

[75] K. Kikuchi and H. Otsuka. Parameter optimization for adaptive control cre in hetnet. In *24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 3334–3338. IEEE, 2013.

[76] G. Su B. A. Yasir and N. Bachache. Range expansion for pico cell in heterogeneous lte—a cellular networks. In *2nd International Conference on Computer Science and Network Technology (ICCSNT)*, pages 1235–1240. IEEE, 2012.

[77] W. Liao S. S. Sun and W. T. Chen. Traffic offloading with rate-based cell range expansion offsets in heterogeneous networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 2833–2838. IEEE, 2014.

[78] Z. Lei T. Quek and S. Sun. Adaptive interference coordination in multi-cell ofdma systems. In *20th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 2380–2384. IEEE, 2009.

[79] W. Tang et al. Joint resource allocation for eicic in heterogeneous networks. In *Global Communications Conference (GLOBECOM)*, pages 2011–2016. IEEE, 2014.

[80] Norbert A. Rácz, N. Reider and G. Fodor. On the impact of inter-cell interference in lte. In *Global Telecommunications Conference (GLOBECOM)*, pages 1–6. IEEE, 2008.

[81] B.J. Veancy S.G. Ruben and P. Yogesh. Scheduling for interference mitigation using enhanced intercell interference coordination. *IJRET*, 3(02), 2014.

[82] A. Weber and O. Stanze. Scheduling strategies for hetnets using eicic. In *International Conference on Communications (ICC)*, pages 6787–6791. IEEE, 2012.

[83] G. Bartoli et al. Adaptive muting ratio in enhanced inter-cell interference coordination for lte-a systems. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 990–995. IEEE, 2014.

[84] M. Behjati and J. Cosmas. Self-organizing interference coordination for future lte-advanced network qos improvements. In *International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2014.

[85] F.E. Salem et al. Advanced sleep modes and their impact on flow-level performance of 5g networks. In *IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–7. IEEE, 2017.

[86] S.H. Cheng L.C. Wang and A.H. Tsai. Bi-son: Big-data self organizing network for energy efficient ultra-dense small cells. In *84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2016.

[87] K. Samdanis et al. Self organized network management functions for energy efficient cellular urban infrastructures. *Mobile networks and Applications*, 17(1):119–131, 2012.

[88] E. Kisielius et al. Energy efficiency in self organizing networks. In *OPNETWORK*, 2013.

[89] S.E. Elayoubi L. Saker and T. Chahed. Minimizing energy consumption via sleep mode in green base station. In *Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2010.

[90] L. Saker et al. Optimal control of wake up mechanisms of femtocells in heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, 30(3):664–672, 2012.

[91] F. Boccardi I. Ashraf and L. Ho. Power savings in small cell deployments via sleep mode techniques. In *21st international symposium on Personal, indoor and mobile radio communications workshops (PIMRC Workshops)*, pages 307–311. IEEE, 2010.

[92] F. Boccardi I. Ashraf and L. Ho. Sleep mode techniques for small cell deployments. *IEEE Communications Magazine*, 49(8), 2011.

[93] O. Iacoboaiea. *Coordination of SelfOrganizing Network (SON) functions in next generation radio access networks*. PhD thesis, Telecom ParisTech, 2015.

[94] 3GPP TR 36.814. Evolved Universal Terrestrial Radio Access: Further advancements for E-UTRA physical layer aspects (Release 9). March 2010.

[95] M.2135. Guidelines for evaluation of radio interface technologies for imt-advanced. *Report ITU*, 638, 2009.

[96] 3GPP TS 36.214. Lte; e-utra; physical layer; measurements. 2012.

[97] S. Hahn et al. Classification of cells based on mobile network context information for the management of son systems. In *81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[98] S.B. Jemaa T. Daher and L. Decreusefond. Cognitive management of self organized radio networks based on multi armed bandit. In *28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5. IEEE, 2017.

[99] Y. Wang W. Chen and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159, 2013.

[100] D. Chakrabarti S. Pandey and D. Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, pages 721–728. ACM, 2007.

[101] L. Li et al. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[102] N. G. Pavlidis et al. Simulation studies of multi-armed bandits with covariates. In *Tenth International Conference on Computer Modeling and Simulation*, pages 493–498. IEEE, 2008.

[103] Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th international conference on World wide web*, pages 691–700. ACM, 2009.

[104] H. Wang et al. Understanding mobile traffic patterns of large scale cellular towers in urban environment. In *Proceedings of the Internet Measurement Conference*, pages 225–238. ACM, 2015.

[105] P. Bonnel et al. Passive mobile phone dataset to construct origin-destination matrix: potentials and limitations. *Transportation Research Procedia*, 11:381–398, 2015.

[106] X. Wu et al. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[107] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[108] N. McKeown et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[109] Sushant et al. S. Jain. B4: Experience with a globally-deployed software defined wan. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 3–14. ACM, 2013.

[110] M. Bansal et al. Openradio: a programmable wireless dataplane. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 109–114. ACM, 2012.

[111] A. Gudipati et al. Softran: Software defined radio access network. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 25–30. ACM, 2013.

[112] G. Poulios et al. Autonomics and sdn for self-organizing networks. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, pages 830–835. IEEE, 2014.

[113] K. Tsagkaris et al. An open framework for programmable, self-managed radio access networks. *IEEE Communications Magazine*, 53(7):154–161, 2015.

[114] K. Tsagkaris et al. Customizable autonomic network management: integrating autonomic network management and software-defined networking. *IEEE Vehicular Technology Magazine*, 10(1):61–68, 2015.

[115] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[116] I. da Silva et al. Impact of network slicing on 5g radio access networks. In *European Conference on Networks and Communications (EuCNC)*, pages 153–157. IEEE, 2016.

[117] 3GPP TR 28.801. Telecommunication management;study on management and orchestration of network slicing for next generation network (Release 14). March 2017.

[118] L. Breiman. *Classification and regression trees*. Routledge, 2017.

[119] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000.

[120] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

List of Publications and Communications

Journals

- T. Daher, S. Ben Jemaa and L. Decreusefond, "Contextual Bandit for Cognitive SON Management." Submitted to IEEE Transactions on Wireless Communications, 2019.

Conference Proceedings

- T. Daher, S. Ben Jemaa and L. Decreusefond, "Q-Learning for Policy Based SON Management in Wireless Access Networks." In 15th IFIP/IEEE International Symposium on Integrated Network Management (IM 2017) (pp. 1091-1096).
- T. Daher, S. Ben Jemaa and L. Decreusefond, "Cognitive management of self—Organized radio networks based on multi armed bandit." In IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2017) (pp. 1-5).
- T. Daher, S. Ben Jemaa and L. Decreusefond, "Cognitive policy based SON management demonstrator." In IEEE 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN 2018) (pp. 1-3).
- T. Daher, S. Ben Jemaa and L. Decreusefond, "Softwarized and distributed learning for SON management systems." In IEEE/IFIP Network Operations and Management Symposium (NOMS 2018) (pp. 1-7).
- T. Daher, S. Ben Jemaa and L. Decreusefond, "Linear UCB for Online SON Management." In IEEE 87th Vehicular Technology Conference (VTC Spring 2018) (pp. 1-5).

Demonstrators

- Part of Cockpit 5G Demonstrator for Network Management Optimization Through Artificial Intelligence, Salon de la Recherche Orange 2017.

université PARIS-SACLAY | École doctorale de mathématiques Hadamard (EDMH)

Titre : Gestion Cognitive des Réseaux Auto-Organisant de Cinquième Génération

Mots clés : Réseaux Mobiles, Auto-Organisation, Apprentissage par Renforcement

Résumé : L'optimisation de l'opération des réseaux mobiles a toujours été d'un très grand intérêt pour les opérateurs, surtout avec une augmentation rapide du trafic mobile, des attentes qualité de service encore plus élevées des utilisateurs, et l'émergence de nouveaux services requérant des contraintes spécifiques et différentes. Le concept de gestion autonome des réseaux (SON) a été introduit par la 3rd Generation Partnership Project comme étant une solution prometteuse pour simplifier l'opération et la gestion des réseaux complexes. Aujourd'hui, plusieurs fonctions SON sont déjà déployées dans les réseaux. Cependant, les actions conduites par les fonctions SON dans le réseau dépendent de la configuration de l'algorithme même de ces fonctions, et aussi du contexte du réseau et de l'environnement ou cette fonction est déployée. D'autre part, un réseau radio mobile auto-organisant serait idéalement un réseau où toutes les fonctions autonomes (SON) fonctionnent de manière coordonnée et cohérente pour répondre à des objectifs de haut niveau de l'opérateur. L'entité autonome serait donc le réseau capable de s'autogérer pour répondre à une stratégie globale de l'opérateur, exprimée en termes d'objectifs de haut niveau de l'opérateur. A cette fin, nous proposons dans cette thèse une approche qu'on appel ≪ Cognitive Policy Based SON Management ≫ (C-PBSM). Le C-PBSM est capable d'apprendre des configurations optimales des fonctions SON selon les exigences de l'opérateur. Il a également la capacité d'améliorer sa décision au cours du temps en apprenant de son expérience passée, et de s'adapter avec les changements de l'environnement. Nous étudions plusieurs approches pour mettre en place la boucle cognitive en se basant sur l'apprentissage par renforcement (RL). Nous analysons la convergence et la scalabilité de ces approches et proposons des solutions adaptées. Nous prenons en compte la non stationnarité des réseaux, notamment la variation de trafic. Nous proposons également des solutions pour mettre en œuvre un apprentissage collaboratif et un transfert des connaissances. Une architecture SDN (software defined networks) est proposée pour le déploiement des agents d'apprentissage dans le réseau.

Title : Cognitive Management of Self-Organized Fifth Generation Radio Networks

Keywords : Mobile Networks, Self-Organization, Reinforcement Learning

Abstract : The pressure on operators to improve the network management efficiency is constantly growing for many reasons: the user traffic that is increasing very fast, higher end users expectations, emerging services with very specific requirements. Self-Organizing Networks (SON) concept was introduced by the 3rd Generation Partnership Project as a promising solution to simplify the operation and management of complex networks. Many SON modules are already being deployed in today's networks. Such networks are known as SON enabled networks, and they have proved to be useful in reducing the complexity of network management. However, SON enabled networks are still far from realizing a network that is autonomous and self-managed as a whole. In fact, the behavior of the SON functions depends on the parameters of their algorithm, as well as on the network environment where it is deployed. Besides, SON objectives and actions might be conflicting with each other, leading to incompatible parameter tuning in the network. Each SON function hence still needs to be itself manually configured, depending on the network environment and the objectives of the operator. In this thesis, we propose an approach for an integrated SON management system through a Cognitive Policy Based SON Management (C-PBSM) approach, based on Reinforcement Learning (RL). The C-PBSM translates autonomously high level operator objectives, formulated as target Key Performance Indicators (KPIs), into configurations of the SON functions. Furthermore, through its cognitive capabilities, the C-PBSM is able to build its knowledge by interacting with the real network. It is also capable of adapting with the environment changes. We investigate different RL approaches, we analyze the convergence time and the scalability and propose adapted solutions. We tackle the problem of non-stationarity in the network, notably the traffic variations, as well as the different contexts present in a network. We propose as well an approach for transfer learning and collaborative learning. Practical aspects of deploying RL agents in real networks are also investigated under Software Defined Network (SDN) architecture.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France