Universidad de Málaga

Escuela Técnica Superior de Ingeniería de Telecomunicación

Programa de Doctorado en Ingeniería de Telecomunicación

TESIS DOCTORAL

Data Analytics and Knowledge Discovery for Root Cause Analysis in LTE Self-Organizing Networks

Autor:

EMIL JATIB KHATIB

Directora:

RAQUEL BARCO MORENO

2017

UNIVERSIDAD
DE MÁLAGA

AUTOR: Emil Jatib Khatib

http://orcid.org/0000-0001-8518-7297

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

UNIVERSIDAD
DE MÁLAGA

Programa de Doctorado en Ingeniería de Telecomunicación
Universidad de Málaga

Por la presente, **Dra. Dª. Raquel Barco Moren**, profesora doctora del *Departamento de Ingeniería de Comunicaciones* de la Universidad de Málaga,

CERTIFICA:

Que **D. Emil Jatib Khatib**, Ingeniero de Telecomunicación, ha realizado en el Departamento de Ingeniería de Comunicaciones de la Universidad de Málaga bajo su dirección, el trabajo de investigación correspondiente a su TESIS DOCTORAL titulada:

"Data Analytics and Knowledge Discovery for Root Cause Analysis in LTE Self-Organizing Networks"

En dicho trabajo se han propuesto aportaciones originales para la gestión de problemas en redes móviles. En particular, se han propuesto métodos para la adquisición de datos de fallos y para la extracción de sistemas de diagnosis automática. Además, se ha desarrollado una herramienta interactiva mediante la cual se facilita el proceso de adquisición de datos. Los resultados expuestos han dado lugar a publicaciones en revistas y aportaciones a congresos.

Por todo ello, consideran que esta Tesis es apta para su presentación al Tribunal que ha de juzgarla. Y para que conste a efectos de lo establecido, AUTORIZA la presentación de esta Tesis en la Universidad de Málaga.

En Málaga, a 12 de enero de 2017

R Barco

Fdo: Raquel Barco Moreno

Acknowledgements

I started the work on this thesis in April, and I am now defending it also in April. In the meantime, five years have already gone by. Five years that have been full of experiences, friendly people, papers, work, deadlines... ups and downs, but mostly learning and growing. This experience lived up to the expectations set by every doctor I have ever met; all the satisfactions provided by seeing my work grow step by step, sometimes in disorder, but always moving forward to the ultimate goal of building something that works. I am also grateful for the great luck that I have had in collaborating with big enterprises such as Ericsson and Nokia that kept me in touch with the real world, and always updated with a market that moves very fast.

In the first place, I would like to express my profound gratitude towards Raquel Barco, who has been my supervisor in this thesis and previously in two master theses. I thank her the great opportunity she offered me in April 2012, for encouraging me in all my work and building an environment where working is enjoyable and learning is easy. I would also like to mention her support in crucial moments, her help with all the publications and for being there whenever I needed assistance.

I would like to thank all the people in Ericsson with whom I have had the opportunity to work and learn from. Special thanks to Inmaculada Serrano, who attended us in the very long team meetings and from whom I learned a lot about the inner workings of big telecom companies. I would not like to forget the help, support and advice I received from Nizar Faour and Ismael Ruiz, and special thanks to Javier Romero and Salvador Pedraza for making this collaboration possible.

I would also like to thank the people in Nokia; although the stay was short, it was a very productive period, both professoinally and personally. In the first place, thanks to Preben Mogensen for accepting me as part of his team for three months. Special thanks to Daniela Laselva, Beatriz Soret, Istvan Kovacs, Klaus Pedersen and Mads Lauridsen for all their help, insight and patience throughout long meetings. I would like to highlight the help that Lucas Chavarría provided from the first instant I arrived in Aalborg to the very last minute.

These acknowledgements would be far from complete if I did not mention each and every of my coworkers and friends in the research group. Thanks to Pablo for his help, both professional and personal, and for setting an example. Thanks to David for all the fruitful collaborations we have had both in Aalborg and Málaga; and also to Ana and Isabel for their help with the complicated processes for finishing and handing in the thesis, and, of course for all these years of work and friendship. Finally, I would like to thank Sergio and Alejandro for their companionship throughout these years and the many fruitful conversations that we have had. The list would go on, since I cannot mention all the things I am grateful from each of them, nor do I have the space to name each and all of the people that have shaped my professional experience these years.

I must also acknowledge the financial support given by the projects cited below, together with the TIC-102 Ingeniería de Comunicaciones research group and the University of Málaga, which made this work possible and allowed me to attend conferences, workshops and research stays abroad and publish in journals.

Last but not least, I would like to acknowledge the contribution of my family and friends. Special thanks to my parents, Borhan and Olya, for raising and supporting me throughout all my life; and my sister, Sheima, for setting an example and guiding me all this time. I cannot write in words all the gratitude I have for you! Thanks also to Cristina, for the support she gave me and the patience she had with me these years. Finally, I would like to thank all my friends for their support; special mention to Rubén, Raúl, Israel, Adrián, Javi, Loren and Tomás.

This work has been partially funded by Optimi-Ericsson, Junta de Andalucía (Agencia IDEA, Consejería de Ciencia, Innovación y Empresa, ref. 59288 and Proyecto de Investigación de Excelencia P12-TIC-2905) and ERDF.

Contents

Abstract

In the last decades, mobile networks have gained an ever increasing importance in the world of telecommunications. What started mainly with the objective of providing global voice service has moved recently towards an almost exclusively broadband data oriented service, embodied in the LTE network. With new services launching continuously, users demand faster networks, with better quality of service and at lower prices. This forces a harsh competition among operators, that need to reduce costs as much as possible and reduce downtimes due to improvement works or problems in the network. To achieve this, Self-Organizing Network (SON) functionalities provide the tools to automate Operation and Maintenance tasks, making them faster and maintainable by a small team of experts. SON functionalities are divided into three main groups: Self-configuration (new elements are automatically configured), Self-optimization (the network parameters are automatically updated for best service) and Self-healing (the network automatically recovers from problems).

In the competitive arena of mobile communications, downtimes due to problems in the network cause a great cost of opportunity, because they affect the user experience. Self-healing is the SON function that deals with the automation of troubleshooting. The main objective of Self-healing is reducing the time required for solving a problem and freeing experts from repetitive tasks. Self-healing has four main processes: detection (identifying that there are problems in a cell), compensation (redirecting the resources to cover the affected users), diagnosis (finding the root cause of a problem) and recovery (performing the required activities to restore the affected elements to normal operation).

Of all the SON functionalities, Self-healing (especially diagnosis) is the most challenging, and therefore it is the least developed. There are no commercial systems that perform automatic diagnosis with enough reliability to convince network operators. This underdevelopment is due to the lack of information for the design of automatic diagnosis systems. There are no databases that collect performance data of problematic cases tagged with their root cause that can be analyzed and studied to find the best solutions. Nevertheless, Artificial Intelligence (AI) methods have been proposed for diagnosis, based on the limited knowledge available. These AI algorithms require a training stage that must be fed with realisitic information. Again, since there are no real fault databases, the training data normally comes from simulations and is unrealistic. The cause of the lack of data is that troubleshooting experts do not register the cases as they are solved. In the competitive scenario of mobile network services, the time of the troubleshooting experts is a scarce resource that must be invested in resolving problems, and not in registering them. In the case that such databases were collected, one major aspect to take into account would be the volume, variability and generation velocity of the collected data, that qualifies as a Big Data problem.

The main problem of automated diagnosis systems is the lack of expert knowledge. To solve this, expert knowledge must somehow be converted into a usable format. This process is known as Knowledge Acquisition (KA). There are two main approaches to KA: manual (through interviews or direct involvement of the experts in the development) or through data analytics (datamining of databases that contain the results of the work of the experts). This thesis studies the data analytics approach, using Knowledge Discovery and Datamining (KDD). For this approach to work, a fault database is required, which is currently a major challenge.

The overall vision of the KA system proposed in this thesis is one where each time that a new diagnosis is done by an expert, he or she can easily and with minimum effort report it and store it in the system. The central part of the system is an evolving diagnosis algorithm (in this thesis, a fuzzy logic controller) that improves and learns with each new example, until experts can rely on its precision for the most commonly identified problems and move on to new ones. New problems can then be added one at a time as they are discovered by troubleshooting experts. In the end, experts are free from repetitive tasks, and can spend their valuable time on more rewarding tasks. Therefore the first objective of this thesis is collecting a fault database. A data collection interface is designed taking into account that experts cannot have a deep involvement, and therefore simplicity is a major requirement. Once data is collected, it will be analyzed to better understand its properties and acquire the information needed for the design of the algorithms. Another objective of this thesis is to create a model of LTE faults, finding the relation between the performance of the network and the occurrence of certain problems. KA is done by applying data analytics on the collected fault data. A KDD process that extracts the parameters of a fuzzy logic controller is designed and used over the collected database. Finally, this thesis aims to do an analysis of the Big Data aspects of the Self-healing problem, and taking them into account throughout the design of the algorithms.

Resumen

En las últimas décadas, las redes móviles han cobrado cada vez más importancia en el mundo de las telecomunicaciones. Lo que empezó con el objetivo de dar un servicio de voz a nivel global, ha tomado recientemente la dirección de convertirse en un servicio casi exclusivo de datos en banda ancha, dando lugar a la red LTE. Como consecuencia de la continua aparición de nuevos servicios, los usuarios demandan cada vez redes con mayor capacidad, mejor calidad de servicio y a precios menores. Esto provoca una dura competición entre los operadores, que necesitan reducir costes y cortes en el servicio causados por trabajos de mejora o problemas. Para este fin, las redes autoorganizadas SON (Self-Organizing Network) proporcionan herramientas para la automatización de las tareas de operación y mantenimiento, haciéndolas más rápidas y mantenibles por pequeños equipos de expertos. Las funcionalidades SON se dividen en tres grupos principales: autoconfiguración (Self-configuration, los elementos nuevos se configuran de forma automática), autooptimización (Self-optimization, los parámetros de la red se actualizan de forma automática para dar el mejor servicio posible) y autocuración (Self-healing, la red se recupera automáticamente de problemas).

En el ambiente competitivo de las redes móviles, los cortes de servicio provocados por problemas en la red causan un gran coste de oportunidad, dado que afectan a la experiencia de usuario. Self-healing es la función SON que se encarga de la automatización de la resolución de problemas. El objetivo principal de Self-healing es reducir el tiempo que dura la resolución de un problema y liberar a los expertos de tareas repetitivas. Self-healing tiene cuatro procesos principales: detección (identificar que los usuarios tienen problemas en una celda), compensación (redirigir los recursos de la red para cubrir a los usuarios afectados), diagnosis (encontrar la causa de dichos problemas) y recuperación (realizar las acciones necesarias para devolver los elementos afectados a su operación normal).

De todas las funcionalidades SON, Self-healing (especialmente la función de diagnosis) es la que constituye el mayor desafío, dada su complejidad, y por tanto, es la que menos se ha desarrollado. No hay sistemas comerciales que hagan una diagnosis automática con la suficiente fiabilidad para convencer a los operadores de red. Esta falta de desarrollo se debe a la ausencia de información necesaria para el diseño de sistemas de diagnosis automática. No hay bases de datos que recojan datos de rendimiento de la red en casos problemáticos y los etiqueten con la causa del problema que puedan ser estudiados para encontrar los mejores algoritmos de tratamiento de datos. A pesar de esto, se han propuesto soluciones basadas en la Inteligencia Artificial (IA) para la diagnosis, tomando como punto de partida la limitada información disponible. Estos algoritmos a su vez necesitan ser entrenados con datos realistas. Nuevamente, dado que no hay bases de datos de problemas reales, los datos de entrenamiento suelen ser extraídos de simulaciones, lo cual les quita realismo. La causa de la falta de datos es que los expertos en resolución de problemas no registran los casos conforme los van solucionando. En el ambiente competitivo en el que trabajan, su tiempo es un recurso limitado que debe ser utilizado para resolver problemas y no para registrarlos. En el caso en que tales bases de datos fueran recogidas, un aspecto importante a tener en cuenta es que el volumen, variabilidad y velocidad de generación de los datos hacen que éste sea considerado un problema Big Data.

El problema principal de los sistemas de diagnosis automática es la falta de conocimiento experto. Para resolver esto, el conocimiento experto debe convertirse a un formato utilizable. Este proceso se conoce como adquisición del conocimiento. Hay dos aproximaciones a la adquisición del conocimiento: manual(a través de entrevistas o con la implicación de los expertos en el desarrollo) o a través de la analítica de datos (minería de datos en bases de datos que contienen el resultado del trabajo de los expertos). Esta tesis estudia la aproximación de la analítica de datos, utilizando las técnicas KDD (Knowledge Discovery and Datamining). Para que esta aproximación pueda ser utilizada, se requiere la existencia de una base de datos de casos reales de fallo, lo cual es un gran desafío.

La visión general de esta tesis es una plataforma en la que cada vez que un experto diagnostica un problema en la red, éste puede reportarlo con un esfuerzo mínimo y almacenarlo en el sistema. La parte central de este sistema es un algoritmo de diagnosis (en esta tesis un controlador de lógica borrosa) que evoluciona y mejora aprendiendo de cada nuevo ejemplo, hasta llegar al punto en el que los expertos pueden confiar en su precisión para los problemas más comunes. Cada vez que surja un nuevo problema, se añadirá a la base de datos del sistema, incrementando así aún más su potencia. El fin es liberar a los expertos de tareas repetitivas, de modo que puedan dedicar su tiempo a desafíos cuya resolución sea más gratificante. Por tanto, el primer objetivo de esta tesis es la colección de una base de datos de casos reales de fallos. Para ello, se diseña una interfaz de usuario para la recolección de datos teniendo en cuenta como requisito prioritario la facilidad de uso. Una vez que se dispone de datos recogidos, se analizarán para comprender mejor sus propiedades y obtener la información necesaria para el diseño de los algoritmos de analítica de datos. Otro objetivo de esta tesis es la creación de un modelo de fallos de LTE, encontrando las relaciones entre el rendimiento de la red y la ocurrencia de los problemas. La adquisición del conocimiento se realiza mediante la aplicación de algoritmos de analítica sobre los datos recogidos. Se diseña un proceso KDD que extrae los parámetros de un controlador de lógica borrosa y se aplica sobre la base de datos recogida. Finalmente, esta tesis también tiene como objetivo realizar un análisis de los aspectos Big Data de las funciones Self-healing, y tenerlos en cuenta a la hora de diseñar los algoritmos.

Acronyms

AI	Artificial Intelligence
ANR	Automatic Neighbor Relation
API	Application Programming Interface
BN	Bayesian Network
CAPEX	Capital Expenditure
CBR	Case Based Reasoning
CDMA	Code Division Multiple Access
CM	Configuration Management
CPU	and the eNodeB
CPU	Central Processing Unit
CQI	Channel Quality Indicator
CS	Circuit Switched
CSV	Comma Separated Values
DB	Data Base
DI	Degraded Interval
DL	Downlink
DM	Data Mining
DSS	Decision Support Systems
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
EMD	Entropy Minimization Discretization
EPC	Evolved Packet Core
EPS	Evolved Packet System
ERAB	E-UTRAN Radio Access Bearer
FFT	Fast Fourier Transform
FLC	Fuzzy Logic Controllers
FM	Fault Management
GERAN	GSM/EDGE Radio Access Network
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HO	Handover
HOSR	Handover Success Rate
HSPA	High-Speed Packet Access
HW	Hardware
IE	Inference Engine
IP	Internet Protocol
K-S	Kolmogorov-Smirnov
KA	Knowledge Acquisition
KB	Knowledge Base

KBS	Knowledge Based Systems
KDD	Knowledge Discovery and Data Mining
KPI	Key Performance Indicator
LOM	Largest Of Maximum
LTE	Long Term Evolution
MAC	Medium Access Control
MDA	Minimum Degree of Activation
MID	Minimum Inclusion Degree
MME	Mobility Management Entity
MMF	Mutation Multiply Factor
MVC	Model-View-Controller
NGMN	Next Generation Mobile Networks
NRMSE	Normalized Root Mean Square Error
OFDMA	Orthogonal Frequency Division Multiple Access
OPEX	Operational Expenditure
OSS	Operations and Support System
PDCP	Packet Data Convergence Protocol
PDF	Probability Density Function
PI	Performance Indicator
PM	Performance Management
PPV	Positive Predictive Value
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RB	Rule Base
RF	Radio Frequency
RLC	Radio Link Control
ROP	Report Output Period
RRC	Radio Resource Control
RSSI	Received Signal Strength Indicator
SC-FDMA	Single Carrier Frequency Division Multiple Access
SMS	Short Message Service
SOM	Smallest Of Maximum
SON	Self-Organizing Networks
SW	Software
UE	User Equipment
UI	User Interface
UL	Uplink

UMTS	Universal Mobile Telecommunications System
UTRAN	Universal Terrestrial Radio Access Network
VoIP	Voice over IP
WLAN	Wireless Local Area Network
WM	Wang-Mendel
eNB	eNodeB
iRAT	Inter-Radio Access Technology

Chapter 1

INTRODUCTION

In this chapter, the purpose and the motivation of this thesis are explained, along with a more detailed list of objectives and the document structure.

1.1 Motivation

In the last decades a paradigm shift towards mobile communications has taken place. Communication used to involve transmitters attached to a *place*, but after this paradigm switch, the focus is set on communication among *users* that may be moving at great speeds. This new approach involves a wide and ever growing range of services, to a continuously increasing user base. The consequence is that mobile communications represent an increasing proportion of the telecommunications industry, and are acquiring a role in modern society that is growing in importance.

Mobile communications for the masses started with telephony, and gradually moved towards data services. The precedents of mobile telephony date back to the Second World War. Nevertheless, it is commonly acknowledged that the first mobile phone call using a prototype of a modern cellular network took place in Manhattan on April 3, 1973, by Motorola employee Martin Cooper, who called the headquarters of Bell Labs in New Jersey. In the early 90s, a minor feature of the Global System for Mobile Communications (GSM) began gaining attention from the users: the first machine-generated message was sent using the Short Message Service (SMS) in the UK on December 3 1992. In 1993, the first person-to-person SMS was sent in Finland. SMS was adopted widely by the users due to its low cost, marking a new and unexpected trend in mobile communications. The voice service was no longer the only option. In the late 90s, data transfer started to be offered by mobile network operators. Mobile communications have evolved to a point where voice is no longer the mainly used service. Nowadays, mobile networks are mainly data-oriented, and mobile communication terminals have evolved to smartphones, that are more close to portable computers than the portable phones sold in the 90s.

To cope with these changes over time, several *generations* of cellular networks have been developed, deployed and operated to provide service. The First Generation (1G) of cellular networks started in Japan in 1979 and in the Nordic countries in 1981. Throughout the 80s, 1G networks were deployed in other regions. These networks were all analog and incompatible among them. In the 90s, the Second Generation (2G) of cellular networks appeared. Two competing standards arised; GSM in Europe and Code Division Multiple Access (CDMA) in the USA. These networks improved 1G with digital transmission instead of analog, increased security, SMS and mobility among countries that used the same standard. With 2G, mobile communications became widespread among users, and the demand for bandwidth started growing and pushing for new and improved technologies. As a response to this growth, the Third Generation (3G) was introduced in the first decade of the new millennium. 3G was designed with data transmission as the main use case (packet switching instead of circuit switching, typically used in telephony; although 3G supports both). The 3G network had only one protocol, Universal Mobile Telecommunications System (UMTS), finally enabling worldwide roaming. With an ever growing demand for data bandwidth, the Fourth Generation (4G) was soon needed. Long Term Evolution (LTE) was adopted as the protocol for 4G, and has recently been deployed in networks worldwide. LTE not only introduced a greater bandwidth, but the network architecture was fully redesigned and simplified and new mechanisms for improving the quality of service were introduced. Currently, research and standardization is ongoing towards the Fifth Generation (5G) of mobile networks. Higher bandwidths are predicted, along with improved Operation and Maintenance (O&M) that will reduce costs and downtimes.

In all this scenario of growing demand for data bandwidth, coverage and quality, the investment in infrastructure (Capital Expenditure, CAPEX) must be high. O&M acquires a key role since networks must always be optimized to offer the best possible service and troubleshooting must be done with the minimum impact on service. As a consequence, the required investment in O&M (Operational Expenditure, OPEX) increases. In a market with a very high competition, minimizing OPEX is a key competitive advantage.

To reduce OPEX, automation is used where possible in O&M tasks, freeing human experts for non-repetitive tasks. Networks with a high degree of automation are called *Self Organizing Networks* [1] (SON, Figure 1.1). The SON functions belong to three categories: *Self-configuration* (new elements are automatically configured), *Self-optimization* (the network parameters are automatically updated for best service) and *Self-healing* (the network automatically recovers from problems). The interest of both network operators and research groups has resulted in the development of seveal research projects and consortiums: CELTIC Gandalf [2], FP7 E3 [3], FP7 SOCRATES [4], SELF-NET [5], UniverSelf [6], SEMAFOUR [7] and COMMUNE [8]. These projects do not cover the different SON topics equally; Self-healing has been the least studied problem because of the limitations and challenges present in this line of work, as it will be explained in Section 1.3.

Self-healing has four main processes: *detection* (identifying that there are problems in a cell), *compensation* (redirecting the resources to cover the affected users), *diagnosis* (finding the root cause of a problem) and *recovery* (performing the required activities to restore the affected elements to normal operation). Among these processes, there is significant research in detection [9][10][11][12], whereas diagnosis has not received as much attention, despite being a key component of Self-healing. This thesis studies the problem of automated diagnosis, exploring

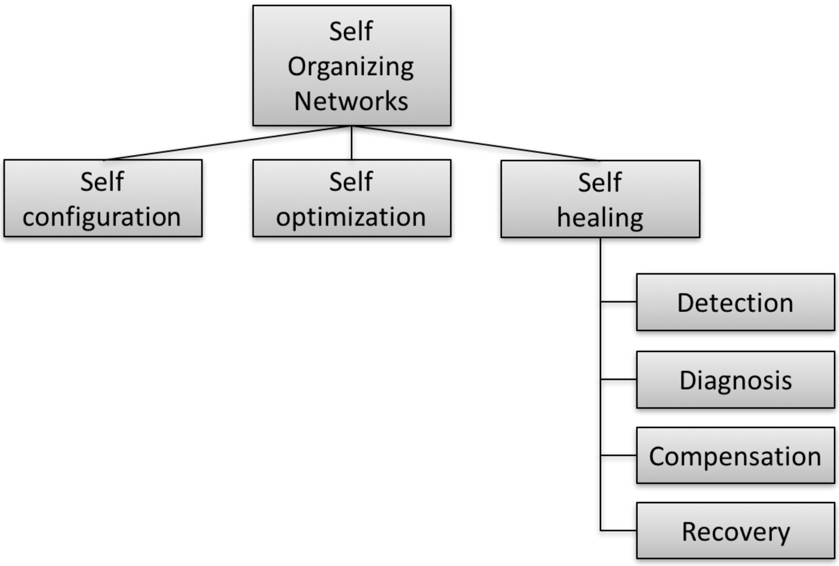


Figure 1.1: Self-Organizing Network and Self-healing functionalities

Artificial Intelligence (AI) algorithms for performing this operation. Although some studies have been done on the subject [13][14][15][16][17][18][19][20][21], at this moment automated diagnosis does not have a wide adoption from the market because of the lack of techniques and platforms to easily train and tune the AI algorithms. These algorithms usually need to be trained by troubleshooting experts, that usually have no availability to spend the required time and effort. Therefore, the main aim of this thesis is to devise methods to train those algorithms with minimal intervention from human experts in order to obtain accurate fault diagnoses.

1.2 Preliminaries

The Mobile Network Optimization (MobileNet) research team belongs to the Ingeniería de Comunicaciones Group (TIC-102). The main research line of the group is the development of techniques for the improvement of current and future mobile networks. MobileNet was created in 2003 by six associate professors from the Communications Engineering Department of the University of Málaga (UMA), in the advantageous occasion of a 4 year joint venture with Nokia Networks to develop a mobile system engineering centre at the Parque Tecnológico de Andalucía (PTA). Over these years, the group has grown to hire more than 30 engineers, and it has taken part in many regional, national and international projects, as well as in projects with the main international operators and vendors: Nokia-Siemens, Ericsson, Alcatel-Lucent, Telefónica, Orange-France Telecom, etc. The group is a leader in the subject of SON, having started as early as in 2G, when the term SON was not even created yet. The team participated in Gandalf (CP2-014 EUREKA/GANDALF: Monitoring and self-tuning of radio resource management parameters in a multi-system network, 2005-2007), which is one of the first and mostly cited projects in the topic, in a partnership with France Telecom R+D, Ericsson R+D Ireland, Moltsen Intelligent SW, Telefónica R+D and the University of Limerick. The Gandalf project aimed to develop self-optimization techniques for networks with multiple radio access technologies and automatic diagnosis tools. In 2012 a new partnership with Optimi-Ericsson was started, with the focus set on the development of SON functions for LTE networks. This partnership is still ongoing and has supported part of this PhD.

1.3 Research challenges and objectives

The main objective of this thesis is to lay the foundations of a system that overcomes the limitations currently holding back the development, testing, deployment and active use of automatic diagnosis systems. Figure 1.2 shows the main objectives of this thesis on the current scenario in automatic diagnosis.

Although the automation of LTE troubleshooting (specifically diagnosis) is a necessity for operators because of the gains in time, quality of service and costs, it has not been widely adopted yet. In spite of the fact that several approaches based on AI algorithms have been proposed [13][14][15][16][17][18][19][20][21] in literature, they have not been implemented in commercial tools. The cause of this low adoption rates is the lack of well performing automatic diagnosis systems that caters for the needs of operators in real scenarios (**Challenge 1**). Therefore, currently, fault diagnosis is mainly a manual task, engaging human experts for hours or days and increasing the costs. Valuable expert time is spent in the resolution of repetitive tasks that can otherwise be efficiently handled by Decision Support Systems (DSS) based on well known AI algorithms. In this thesis, Fuzzy Logic Controllers [22] (FLCs) are chosen as the core of a DSS for diagnosis (**Objective 4**), due to their easily understandable structure and the relative portability of their knowledge base, that ease the tasks of importing, exporting and integrating expert knowledge.

Generally, DSS are created following several steps: *development* (or selection among the numerous readily available solutions [23][24][22][25][26][27]) of a core AI algorithm that performs the analysis of the available data (where the lack of the information on the properties of data from real troubleshooting scenarios limits the reach and realism of the tests that have been done in prior studies for diagnosis), *training* of the solution over the target system (that requires either real data in order to train the AI algorithms or experts that are willing to spend the time and effort to manually configure them) and *exploitation* over the lifetime of the solution (involving regular checks and updates of the system that, again, require either real data or manual configuration). Although many AI solutions are available and well studied, in the case of diagnosis, the selection of the specific algorithm is hindered by the lack of an analysis of the nature of the data (**Challenge 2**). Therefore, this thesis aims to study the properties of data that troubleshooting experts use for manually performing diagnosis (**Objective 2**), in order to have a well-informed choice.

DSS systems need to have a codified expert knowledge that drives their decision making capabilities. This knowledge is stored in a codified manner in the system, with a format that widely varies according to the type of AI algorithm used. The inclusion of this knowledge requires a process of *Knowledge Acquisition* (KA), where the experience of troubleshooting engineers is elicited and saved into machine-understandable format. This can be done by two different paths: either with human intervention (by means of an interview [28][29][30][31] or by teaching the experts how to configure the chosen AI method, which is usually a cumbersome process leading to a lack of collaboration -**Challenge 4**- and ultimately to an underdevelopment in the field of automatic diagnosis -**Challenge 1**), or by *Knowledge Discovery and Datamining* (KDD [32][33]), which performs *data analytics* of the inputs and outputs of the experts work in order to find reproducible patterns and translate them into usable knowledge. KDD greatly reduces the burden of the training stage of DSS, decreasing the intrusiveness of the process in the workflow of the troubleshooting engineers. But even with the availability of KDD methods for training automatic diagnosis systems [34][35][36][37], they still have not seen a great adoption by major network operators. KDD has been used in some mobile network studies, such as [38], where KDD is used for automatic Radio Resource Management, and [39], where it is used for detecting cells with bad Radio Access Channel configuration.

One common aspect of all the steps in the creation of DSS based automatic diagnosis is the need for representative data. As already stated, the design (or selection) of the AI algorithm that will perform the diagnosis requires prior knowledge on the kind of data that will be processed. This information cannot be obtained unless real troubleshooting data is available. Training the chosen algorithm with data analytics is also impossible without real cases. The lack of such data (**Challenge 3**) often leads to training the algorithms using simulated data, which does not produce optimal results and is often unconvincing for the network operators. Another common solution is using unsupervised learning methods, which in the case of diagnosis entrails learning patterns without knowing the associated problem. This leads to KA systems that require the intervention of troubleshooting engineers to identify and tag the detected patterns, making the whole system again depend on the availability of such experts. There is a clear need for real troubleshooting cases, so one of the targets of this thesis is creating a database of real troubleshooting cases (**Objective 1**). The volume of data available from current networks for troubleshooting, as well as the variety of formats it is given in, can be overwhelming for traditional computation algorithms (**Challenge 5**), which may require the use of Big Data techniques.

With a dataset of real troubleshooting cases, most of these reported shortcomings can be addressed. The lack of information on the nature of the data can be solved by analyzing this dataset, finding the best techniques for processing it (**Objective 2**, **Objective 6**) and creating a model (**Objective 3**) that clearly identifies the behaviour of each variable under different situations. A model of the data can also help with the lack of data, since it can be used to improve and validate the results of simulated scenarios; as well as for generating new datasets that imitate network faults.

The specific objectives (summarized in Figure 1.2) are the following:

Objective 1: **Collection of a fault database**: Although performance data is widely available in operators databases, the root cause or diagnosis related to these data are not usually saved or documented (**Challenge 3**) together with the affected data, because it is a task out of the scope of the daily work of troubleshooting experts (**Challenge 4**). The main reason for this is that the process of manually fetching the data, joining it with a report and saving it in a common database is a cumbersome process that has little value in the short-term for the troubleshooting experts. Therefore, a system that can quickly perform this task with the minimum intervention of the expert is required. If the information is collected from a tool that experts already use to inspect the values of performance data to carry out the diagnosis process, the required effort is minimal. In that case, the collection process would be tightly integrated with the observations of the experts, only expecting them to tag data with a diagnosis and letting the system do the rest of the collection, processing and storage tasks. In this thesis, a software platform that only requires three variables (name of the affected sector, date when the problem was diagnosed and diagnosis) is developed. When integrated into a data visualization tool that experts use frequently, two of these three variables (name of the affected sector and date) can be extracted from the context of use (i.e. the data that is displayed on the screen). Therefore, experts only need to give a diagnosis (responding to **Challenge 4**). This system is then used to collect a number of real troubleshooting cases. This database contains performance data for a specified time window of cells affected by a known problem, together with a tag that identifies the problem as diagnosed by troubleshooting experts. The existence of such a database solves the lack of real data (**Challenge 3**) that hinders the development of automated diagnosis.

Objective 2: **Analysis of the collected troubleshooting data**: Once the problem of lack of data is solved by the collected database, the missing knowledge on the nature of this data (**Challenge 2**) can be approached. This stage of the study includes the characterization of the different types of data that have been collected in order to better design the AI algorithms that will perform diagnosis. Also, since the collected data has a set of properties that may not be compatible with the requirements of the Data Mining (DM) methods that will train the AI algorithms, one of the main tasks in this stage is to determine the required processing and design the methods to carry it out. In fact, these preprocessing steps are part of the data analytics process. This stage solves the lack of knowledge and paves the way for designing better AI algorithms as well as training them with real data from the LTE network.

Objective 3: **Creation of models for the most common LTE faults**: with the double purpose of generating new plausible fault data and also for having a better understanding of the problems under study, a model of the collected database is generated. The individual performance measurement variables are characterized using statistical models conditioned to the occurrence of each of the most common faults. This model can also be used to validate simulations, since it gives a ground truth to which simulated results must conform. The final product of this stage helps to better understand the behaviour of the data (responding to **Challenge 2**) and solves the lack of cases for testing AI algorithms by enabling the generation of new realistic cases (**Challenge 3**).

Objective 4: **Study of AI methods for diagnosis**: with the available knowledge on the behaviour of the network performance data, the most common problems and the manual process of troubleshooting, the next logical step is to design an AI algorithm that serves to the purpose of automatic diagnosis. In this thesis, Fuzzy Logic Controllers [23][24] are chosen as the best option, since they are easily understandable both by humans (by using a language that is close to spoken language) and by machines. FLCs make it easy to create troubleshooting rules automatically and fuse them with previously known ones. This solution is also attractive for network operators, since its understandability makes it less obscure than other alternative methods, such as Bayesian Networks [25] or Neural Networks [27].

Objective 5: **Design and test of a KA platform**: in order to obtain good diagnosis results, the chosen AI method must be trained with real network data, so that it has knowledge of the problems it will be encountering once deployed. For this, a data analytics process for the FLC is designed in this thesis, based on the information and the processes extracted from the stage of analysis of the collected fault database. The product of this stage will be a software system that when given a full database of real faults, returns a set of diagnosis rules that can be used in an FLC. These rules are adapted to the real scenarios where they will be used (responding to **Challenge 1**). This software can also be coordinated with the collection platform, so when new data arrives to the database, the knowledge model of the FLC is improved.

Objective 6: **Analysis of Big Data aspects and design considerations**: in modern networks, especially in LTE, the amount of performance and configuration data is large. This data comes from varied sources (such as the user devices, the access nodes, etc.) and in very diverse formats (different file types, temporal resolutions, etc.), which requires additional homogenization steps the data analytics process. Also, due to the large amount of events that occur throughout the network, data is continuously generated in a fast velocity. All these aspects (**Challenge 5**) conform to the Big Data [40] paradigm. In Big Data analytics problems, traditional processing techniques do not satisfy the performance requirements (i.e. they are unable to cope with the amount of work in the available time), therefore new techniques must be used. This thesis will study the Big Data aspects of the troubleshooting data used in diagnosis and take them into consideration for the design of all the algorithms and processes.

The combination of these parts constitutes the overall system that this thesis envisions: one where each time that a new diagnosis is done by an expert, he or she can easily and with minimum effort report it and store it in the system. The central part of the system is an evolving diagnosis algorithm that improves and learns with each new example, until experts can rely on its precision for the most commonly identified problems and move on to new ones. New problems can then be added one at a time as they are discovered by troubleshooting experts. In the end, experts are free from repetitive tasks, and can spend their valuable time on more rewarding tasks. Also, an array of additional helpful intermediate results can be used as feedback to the troubleshooting engineers: models of the most common problems, results of data preprocessing, etc.

1.4 Document structure

This thesis is divided into seven chapters. These chapters can be grouped into three blocks (Figure 1.3).

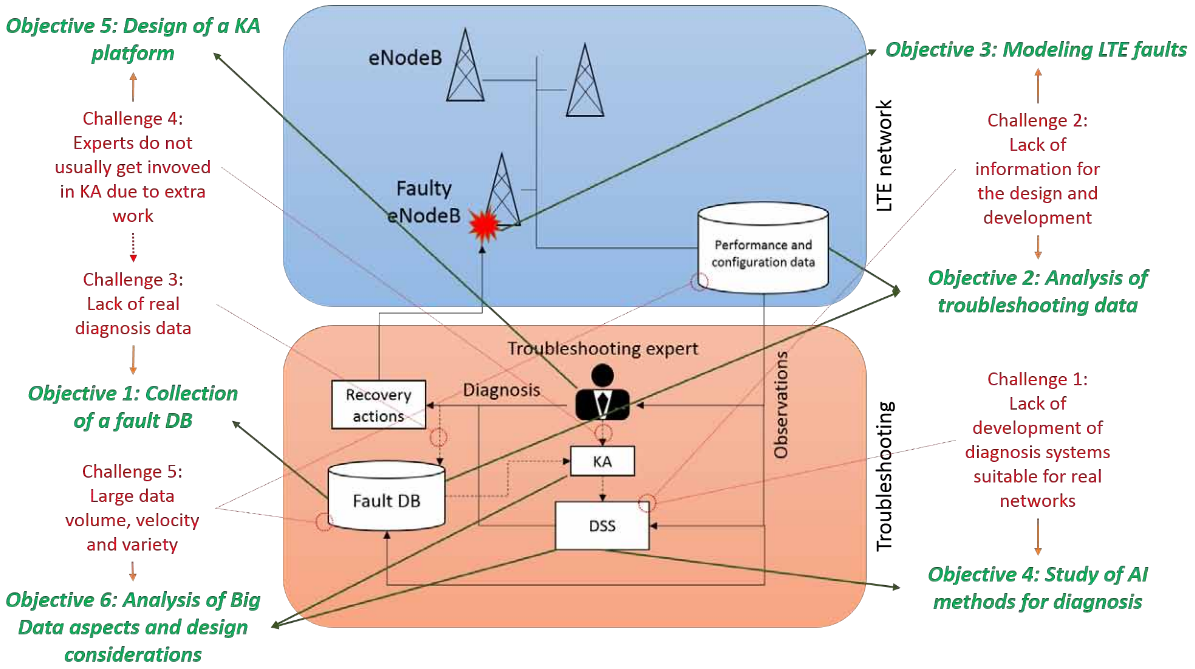

Objective 5: Design of a KA platform
Challenge 4: Experts do not usually get invoved in KA due to extra work
Challenge 3: Lack of real diagnosis data
Objective 1: Collection of a fault DB
Challenge 5: Large data volume, velocity and variety
Objective 6: Analysis of Big Data aspects and design considerations
eNodeB
Faulty eNodeB
Performance and configuration data
LTE network
Troubleshooting expert
Recovery actions
Diagnosis
KA
Fault DB
DSS
Observations
Troubleshooting
Objective 3: Modeling LTE faults
Challenge 2: Lack of information for the design and development
Objective 2: Analysis of troubleshooting data
Challenge 1: Lack of development of diagnosis systems suitable for real networks
Objective 4: Study of AI methods for diagnosis

Figure 1.2: Main challenges and objectives

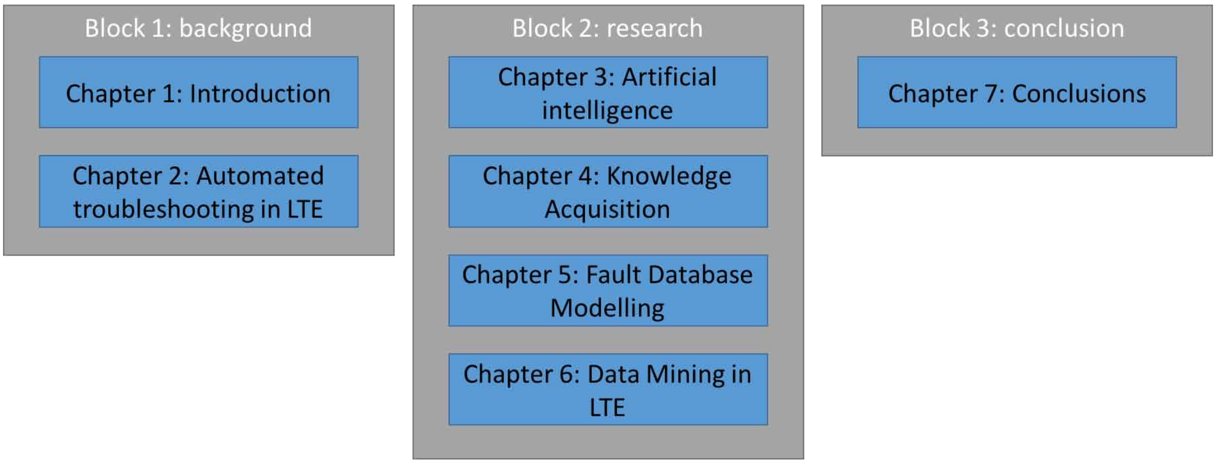

Block 1: background
Chapter 1: Introduction
Chapter 2: Automated troubleshooting in LTE
Block 2: research
Chapter 3: Artificial intelligence
Chapter 4: Knowledge Acquisition
Chapter 5: Fault Database Modelling
Chapter 6: Data Mining in LTE
Block 3: conclusion
Chapter 7: Conclusions

Figure 1.3: Organization of the thesis

The first block contains the chapters that describe the research background and State of the Art in the diverse areas that this thesis covers. In Chapter 1 an introduction has been given, briefly describing the scope of the thesis and giving an outline of the challenges and objectives. Next, in Chapter 2 a brief introduction to LTE networks is provided to help understand the troubleshooting process. A more in-depth review of Self-Organizing Networks, with emphasis on automated troubleshooting is given in this Chapter, along with an exploration of the State of the Art.

The second block contains the bulk of the research work of this thesis. In this block, the main objectives described in Figure 1.2 are addressed. Chapter 3 will explore the AI algorithms commonly used in diagnosis. The FLC algorithm will be explored in full detail, and its use on diagnosis will be explained (**Objective 4**), explaining details of the FLC algorithm used in this thesis and how it interacts with the process of KA. In Chapter 4 the KA process is fully described, with emphasis on its application to the problem of diagnosis. The data collection platform will be described in this Chapter (**Objective 1**). Chapter 5 analyzes the collected data, describing in detail its main features. With a better understanding of the properties of the collected data, as well as the needs of the KA process, a set of preprocessing algorithms is proposed for treating the data (**Objective 2**). In this Chapter the process for modelling the database is also explained (**Objective 3**). In Chapter 6 the Data Mining algorithms designed for the KA process are described and tested using the data extracted from the network, thus completing the full data process from collection to extraction of expert knowledge (**Objective 5**).

The last block contains Chapter 7, where the conclusions of this thesis are presented and the achievement of the objectives is summarized.

Chapter 2

Automated troubleshooting in LTE

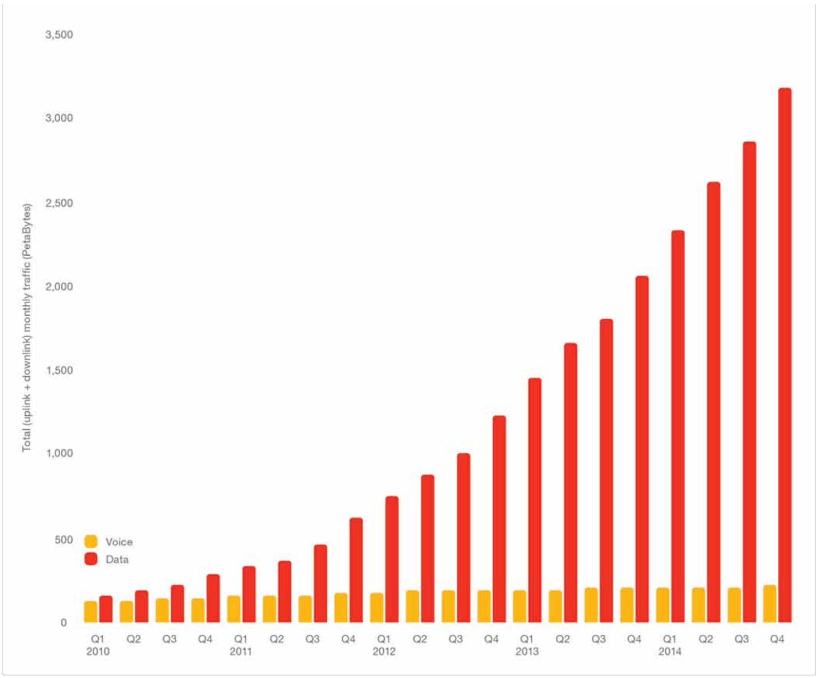
In this chapter, the basic network technologies will be described. The LTE network will first be introduced, showing its features and its emergence and adoption by operators. Next, the need for automating its Operation and Maintenance (O&M) will be discussed, and the solution to this need, Self-Organizing Networks (SON), will then be presented. In subsequent chapters, these concepts will be expanded, and solutions to open issues will be proposed.

2.1 LTE networks

2.1.1 Introduction

In the last decade, cellular mobile networks have grown rapidly in size and complexity. More specifically, the advent of mobile multimedia devices, such as smartphones, tablets and laptops, has driven an increase in the demand of data rates and quality of service. Figure 2.1 shows the increase in data usage over the last years. In this scenario, the bandwidth used by the traditional voice services is dwarfed by the demand of newer Internet-based traffic, showing that the market demands a essentially data oriented network.

Responding to this paradigm shift, new technologies are required to complement the existing GSM and 3G networks. Long Term Evolution (LTE) [41], commercially known as 4G, is the technology that was developed by the 3rd Generation Partnership Project (3GPP) to this avail. It introduces a new network architecture, the *Evolved Packet System* (EPS), where data transmission is the target use-case and voice services are provided as Voice over IP (VoIP). LTE defines a new Radio Access Network (RAN), the *Enhanced RAN* (E-UTRAN), and a new packet switched network, the *Evolved Packet Core* (EPC).



Source: Ericsson Mobility Report (http://www.ericsson.com/mobility-report)

Figure 2.1: Increase of data traffic over the last years vs voice traffic in mobile networks.

2.1.2 LTE development

The initial design requirements for LTE were defined with respect to the existing HSPA technology [42]:

- Packet switched, simplified network with simple *User Equipments* (UEs).
- Peak user throughput of 100 Mbps in the downlink and 50 Mbps in the uplink for a 20 MHz channel. The throughput must escalate linearly with the allocated bandwidth. Average throughput three to four times higher than HSPA in the downlink and two to three times higher in the uplink under loaded conditions.
- Transition time to active state lower than 100 ms.
- At least 200 active users per cell with a 5 MHz spectrum allocation.
- Round trip time lower than 10 ms.
- A spectral efficiency (bits/sec/Hz/site) three to four times higher than HSPA in the downlink and two to three times higher in the uplink under loaded conditions.
- Mobility transparent to the user, optimized for 0-15 km/h and able to maintain communications to up to 350 km/h or 500 km/h (depending on the selected frequency).
- All requirements must be achieved for UEs located up to 5 km from the cell tower, and with a slight degradation, up to 30 km.
- Flexible bandwidth allocation from 1.25 MHz up to 20 MHz.
- Able to coexist, share traffic and hand over connections with legacy 3GPP RAT (GERAN/UTRAN). Handovers between technologies should take less than 300 ms.

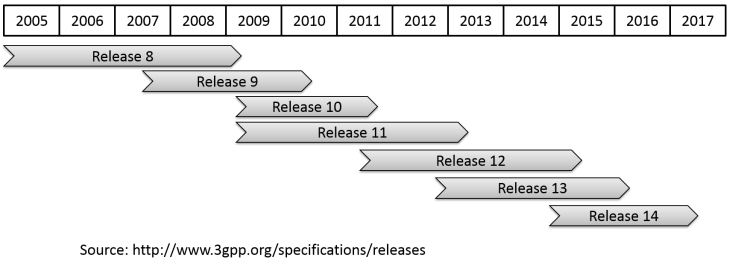


Figure 2.2: Timeline of 3GPP Releases related to LTE

Figure 2.2 shows the timeline of the definition of LTE by 3GPP. The first LTE specifications were defined in 2005 as part of the LTE Study Item (SI), that is, the feasibility study. After the standardization process (Work Item, WI), the first LTE specifications were included in 3GPP Release 8 in September 2006. Further work improved LTE in 3GPP Release 9. In Release 10, new features were described for LTE-Advanced (LTE-A), mainly improving data rates, increasing coverage and reducing latency. Release 11 defines new Quality of Service (QoS) specifications for LTE, as well as Self Organizing Network (SON) enhancements. Further enhancements have been subsequently added in Releases 12 (enhancements over the physical layer, such as small cells, carrier aggregation, dual connectivity, MIMO, elevation beamforming, etc) and 13 (LTE in unlicensed bands, improvement for Machine Type Communications). Future development in Release 14 is expected to add enhancements in energy efficiency, mission critical communications and Massive Machine Type Communications, among other aspects.

2.1.3 LTE architecture

EPS (the network architecture of LTE, Figure 2.3), is composed of the E-UTRAN and the EPC. The EPS is optimized to provide IP connectivity to the UEs, using the concept of *EPS bearer*. An EPS bearer defines a pipe where IP traffic is routed between the UE and the data network, with some Quality of Service (QoS) parameters associated.

The E-UTRAN offers a downlink based on Orthogonal Frequency Division Multiple Access (OFDMA) in order to maximize the number of simultaneous connections and variety of offered QoS. The uplink uses Single Carrier Frequency Division Multiple Access (SC-FDMA) to minimize the power consumption of the UEs. The UEs interface with the eNodeBs (through the Uu interface), which are the providers of the radio service, and therefore contain all the radio functionalities, protocols and layers (Figure 2.4): the Physical layer, the Medium Access Control (MAC) and the Radio Link Layer (RLC). The Packet Data Convergence Protocol (PDCP) is an additional layer that adds security to the communications with the UE. In the Control Plane, and additional Radio Resource Control (RRC) layer applies QoS policies for admission control, scheduling and resource administration. The E-UTRAN is the mesh composed of the eNodeBs interconnected through the X2 interface.

The EPC offers a packet switched service to the end user over the E-UTRAN. It has three kinds of logical elements (Figures 2.3 and 2.4):

- Mobility Management Entity (MME): manages the UE access to the network, the setup of connections through the EPC, etc.

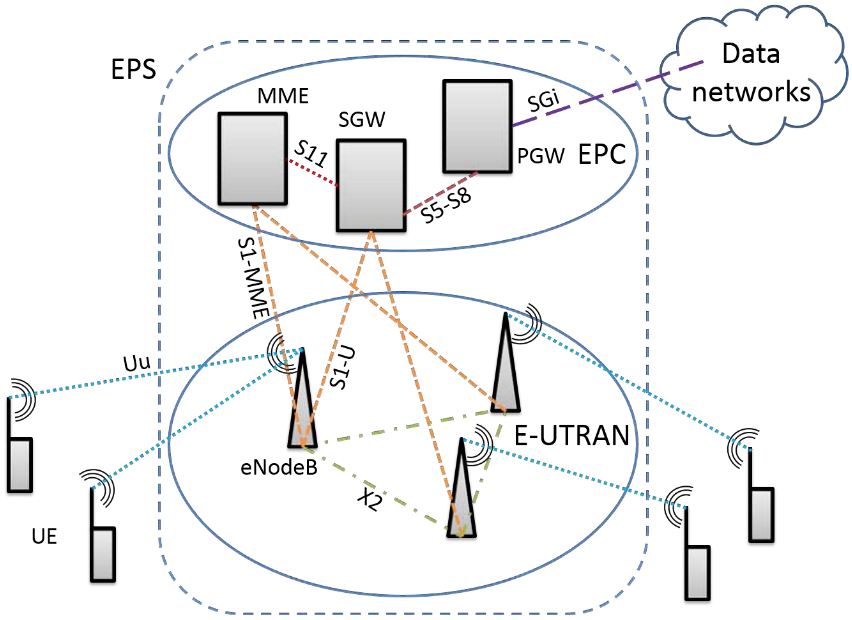


Figure 2.3: LTE architecture

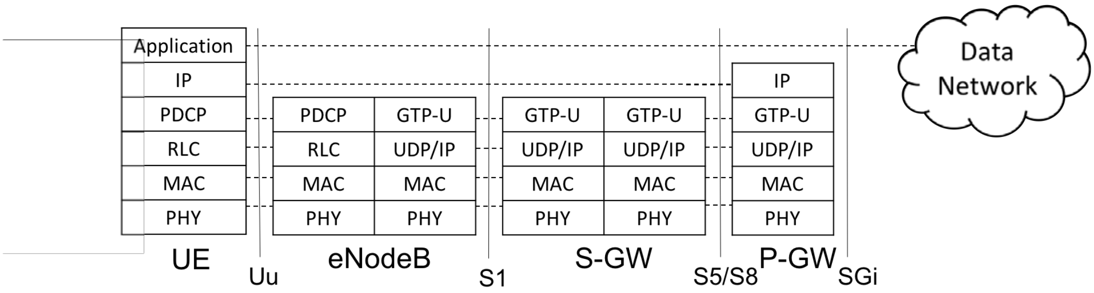
- Serving Gateway (S-GW): provides access for a UE over the E-UTRAN.
- Packet Data Network Gateway (P-GW): provides connectivity with external packet switched networks.

2.1.4 LTE adoption and deployment

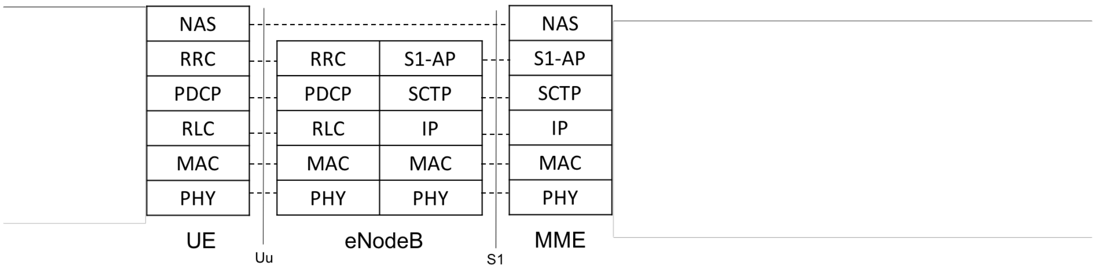
LTE has been adopted by the operators as the latest radio access technology. The most common scenario of LTE adoption is where operators first introduce LTE sectors in restricted urban areas, usually reusing infrastructure of the GSM and 3G networks to provide a limited experimental service. These patches of LTE network gradually expand to cover all the urban areas. The main requirement in this scenario is to integrate LTE with the existing technologies. In zones where the traffic is very high, small cells may complement the usual macro cells to add capacity.

The deployment of LTE networks must also cope with regulatory policies, such as the available spectrum of the digital dividend (the spectrum freed after the change from Analog to Digital TV). An example of policies affecting the deployment of LTE is the german regulation for the 800 MHz band, where rural areas must be covered before urban areas.

The LTE standard defines a broad range of carrier frequencies. The use of these frequencies depends on the regulation of each country and the policies of the operator. For instance, in Europe the 700, 800, 900, 1800 and 2600 MHz bands are used, whereas in North America, the 700, 750, 800, 850, 1900, 1700/2100 (AWS), 2500 and 2600 MHz bands are used.



(a) User plane



(b) Control plane

Figure 2.4: LTE protocol stack

2.2 Self Organizing Networks

2.2.1 Motivation

A mobile network is composed by all the elements of the infrastructure (in the case of LTE the eNodeBs, the components of the EPC, etc.), and also the terminals that access and interact with it (the UEs). The degree of management complexity of a network arises from the large number of elements, the entangled interdependencies among the configurations of each one of them and the large number of events that take place during the operation and service. Another dimension of complexity is added to the management when several different technologies must coexist and cooperate (*Heterogeneous Networks* or HetNets), which is the most common scenario in LTE. This causes an increase in O&M costs, since as the network grows, the workforce requirements grow with it. Also, the demand of the users for a higher quality, and therefore, the reduced tolerance for downtime increases the opportunity cost whenever a problem is not quickly solved or the network is not correctly optimized. In a very competitive market, operators cannot cope with this pressure, and therefore, an increase in automation in the O&M of the network is required. Automation reduces the need for repetitive work from human operators, and therefore the workforce requirements are reduced. Also, the downtime is reduced when problems are automatically detected and solved. Summarizing, automation reduces the costs of O&M in mobile networks.

2.2.2 SON functionality

The NGMN Alliance [43] defines SON as a set of principles and concepts to add automation to mobile networks so that they require less maintenance than traditional networks while improving service quality. Ideally, human operation is restricted to adjusting high-level guidelines that are more in sync with commercial decisions than low level functionality.

NGMN defined a list of SON use cases, based on common problems faced by network operators. These use cases belong to four categories:

- Planning: Site location, configuration parameters and network integration.
- Deployment: installation and initial configuration, testing, network authentication, setup transport/radio and network integration.
- Optimization: transport and radio parameters.
- Maintenance: HW/SW upgrade or replacement, failure recovery and network monitoring.

These operational use cases influenced the standardization of SON functions by 3GPP in Release 11 [44] [45]. SON functionalities are classified in three large categories:

- Self-Configuration [46]: functionalities that automate the planning and deployment of the network. Self-Configuration reduces the cost of deployment, since adding elements to the network becomes much easier and faster.
- Self-Optimization [46]: functionalities that automate the optimization of the network, that is, functionalities that keep the configuration parameters always working in the optimal level to offer the best QoS, adapting them to changes in the environment. Self-Optimization reduces the need for new elements in the network and increases the quality perception of users.
- Self-Healing [1]: functionalities that automate the solution of problems, reducing human intervention and minimizing downtime. Ideally, Self-Healing functions can do proactive healing, that is, solve problems before they occur. Self-Healing reduces the cost of maintenance by reducing the workforce needed for fixing common problems and the cost of opportunity by reducing downtime in the service.

2.2.3 SON implementation

SON functionalities are usually implemented with Artificial Intelligence (AI) algorithms. The inputs to these functionalities are measurements taken on the network. These measurements have a wide variety of origins: the eNodeBs[47] [48][13][31][14][34][9][49][50][51], the UEs[52], the complaint department of customer service, etc. This variety, along with the large volume and the needs for quick results qualify the SON scenario as a *Big Data*[40] problem. The output of these SON functions are also varied, and their nature depends on the target of the specific algorithm. The output may range from human-readable reports to automated action commands that trigger other SON functions (for instance, a Self-healing algorithm may detect and diagnose a problem and its output may be the activation of an automated recovery function that solves the problem without human intervention; or a Self-optimization algorithm may indicate that part of the traffic of one cell should be handled by a neighboring cell and therefore it may send commands to both cells in order to adjust their handover parameters). Some AI algorithms that have been used in SON are Fuzzy Logic Controllers (FLCs)[47] [48][53][54], Bayesian Networks (BNs) [13][31], Case Based Reasoning (CBR) [14][34] or Neural Networks [9][49][50][51].

An important decision when implementing SON functions is their location inside the LTE network infrastructure. The main decision is among a distributed (among the elements of the network) or centralized (in the O&M center) location. Each option has some advantages and drawbacks. If SON functions are distributed, they usually will have a quicker access and reaction time to the events of the network, since there is no time lost in transmissions to a centralized location. On the other hand, their local scope prevents that these implementations have a comprehensive view of the network, and therefore, many SON functionalities cannot be implemented this way. Centralized implementations have access to all the data of the network, with the associated delay of collection and an increased execution time due to the large amount of required processing.

2.2.4 Research projects

Some major research projects have been done to advance SON functionalities:

- Gandalf (2005-2006): Part of the Celtic European research and development programme. The main objectives of the Gandalf project were the automation of HetNets (GSM, 3G and WLAN). Some investigated issues were Joint Radio Resource Management (JRRM), automated configuration and diagnosis of problems. The project made a wide use of AI algorithms, such as BNs for diagnosis or Q-Learning and fuzzy logic for automated configuration of some parameters. The results were validated with simulations and a reduced network testbed, as well as with real data collected from networks (for automated troubleshooting).
- COST 2100 SWG 3.1 (2006-2010): This project was mainly targeted at improving the available data for SON, because traditionally only data from simulations was used to test the SON algorithms. This project aimed to collect live network data (for instance, from drive tests) for its use in SON research.
- E3 (End-to-End Efficiency) (2008-2009): European research project targeted towards the investigation of Cognitive Management and Control. Learning algorithms were used to improve the Self-Optimization functionality.
- Socrates (2008-2011): European research project that had as main objective the solution of eight SON use cases: Load Balancing, Handover Parameters, Home eNB, Admission Control, Packet Scheduling, Interference Coordination, Cell Outage Management and Automatic Generation of Initial Parameters for eNB Insertion. The developed solutions were validated with simulations.
- UniverSelf (2010-2013): European research project with the main objectives of unifying the management of networks based on diverse technologies and the integration of SON functions.
- COMMUNE (2012-2014): This project was targeted towards the study of uncertainty in mobile networks. It highlighted uncertainty as the main cause of the low development and adoption of the SON functions.
- SEMAFOUR (2012-2015): European research project with focus on Heterogeneous Net-

works. Multi-RAT and multi-layer SON functions are studied, with the aim of creating an integrated SON management system.
- Selfnet (2015-2018): Part of the Horizon 2020 program, Selfnet is a framework for self-organized network management in virtualized and software defined networks. The main objective is to design new SON functions for future 5G networks.

Summarizing, SON functions help operators reduce the costs of the O&M of the LTE network. Also, the downtimes due to optimization or fault recovery are reduced, so the overall QoS of the network increases.

2.3 Self-healing

2.3.1 Manual troubleshooting

One of the most important O&M tasks is troubleshooting. The main objective of troubleshooting is finding and fixing malfunctions in the network, minimizing the impact on the quality of the offered service. In the currently deployed LTE networks, this task is mainly manually done, that is, human experts monitor some variables that reflect the state of the network (Performance Indicators, Alarms, Call Traces, etc.). Within SON, self-healing aims to automate troubleshooting tasks. The manual troubleshooting workflow (Figure 2.5) has 4 main subtasks:

- Detection: the process of determining that there is a problem in the network, and pinpointing the element or elements that are affected. To do this, troubleshooting engineers will usually monitor a very reduced set of Key Performance Indicators (KPIs), that is, variables that reflect the high level behaviour of the network. When one or more of these KPIs are degraded, a list of the *Worst Offenders* is extracted, that is, the elements (eNodeBs, for instance) that are degrading mostly the KPI averages. This helps to find the problematic elements.
- Diagnosis: also called Root Cause Analysis. Once the problematic elements have been determined, troubleshooting engineers must find out the *Root Cause* (why they are failing). The study of low level *Performance Indicators* (PIs), as well as logs or event records may help in the determination of the Root Cause. In some cases, the analysis of the available data is not enough, so active measurements are taken, such as drive tests.
- Recovery: once the root cause is known, the required actions to fix it are taken. These actions are diverse in complexity, ranging from simple resets or configuration changes that can be ordered remotely, to hardware fixes or replacements that need on site reparations. Since the cost of resetting an element is very low, it is often the first action taken. If it does not resolve the problem, then more complex actions are taken. The recovery action may or may not solve a problem, so the results of the action are taken into account on subsequent repetitions of the Diagnosis subtask.
- Compensation: the troubleshooting process may take anywhere between minutes to several days. Therefore, it is important to redirect the resources of the network temporarily to provide service to the users in the affected area, e.g. by providing coverage with neighboring cells to a cell in outage . Since this temporary configuration is sub-optimal, the users may

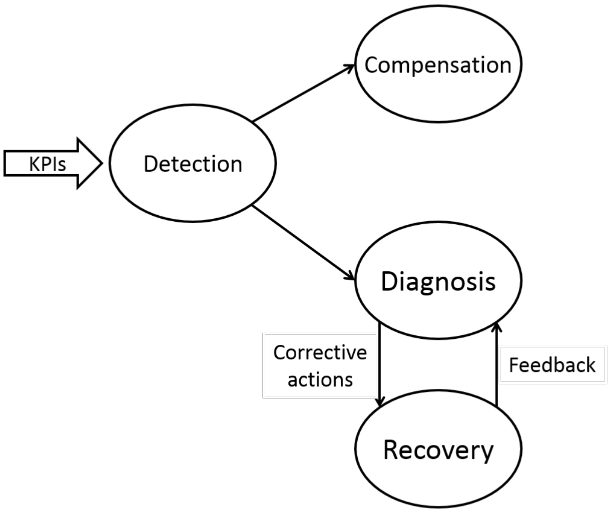


Figure 2.5: Manual troubleshooting workflow

perceive a reduced QoS, but the alternative is the total lack of service, and therefore a huge cost of opportunity.

The troubleshooting process uses data sources that indicate the state and behaviour of the network. These data sources contain information about the performance of the eNodeBs (Performance Management -PM- counters), the faults that occur during their operation (Fault Management -FM- alarms) and their configuration (Configuration Management -CM- parameters). Other data sources are sometimes used, such as measurements of the performance of individual calls (call traces) or previously reported problems (trouble tickets). These data sources are defined in Chapter 4.

All these data sources are manually processed by troubleshooting experts. To better understand the meaning of the data, experts often use statistical techniques to simplify the representation. For instance, to detect degradations on PIs or KPIs, *thresholding* is often used, that is, the value is considered degraded if it is above or below a certain threshold. Thresholding is specially important in the detection stage, but it is also used to discretize the value of PIs, classifying them in either *good* or *bad* values. With quantized variables, heuristic rules of the *"if ... then ..."* type are often used in the diagnosis process. This pattern is generally used by experts in their observation of low level PIs, and sometimes these rules are used in the database containing the PM, FM and CM variables to test for known fault states. Another technique that is sometimes used to test the relations among variables is *correlation*. For instance, alarms or engineer actions are correlated with PIs to see to what extent are they responsible for the observed behaviour.

The results of all these tests are pondered by troubleshooting experts to come up with a solution for a problem. All this process relies entirely on the experience of the engineer, and it is therefore prone to human errors. Also, since it is an iterative process, it may take long for a problem to be effectively solved.

Automation of all these actions will bring speed to the overall process, and since human intervention is reduced, there is no dependence on the expertise of the engineer. Automation also reduces the workload of experts by solving common and repetitive problems, therefore enabling them for more challenging tasks.

2.3.2 Automated troubleshooting: Self-healing

Self-healing [1] is defined as the set of functionalities that automate troubleshooting. According to the NGMN use cases [43], Self-healing covers four specific scenarios:

- Hardware extension/replacement: In some cases, a problem in a network may be caused by a hardware component. This may be due either to malfunction or lack of resources. The task of Self-healing in this case should be to detect the exact problem and to determine the solution that would cost less. Self-healing may also play a role of preventive troubleshooting if the need for hardware extension or replacement is done in a predictive manner.
- Software upgrade: Other scenarios may be caused by software issues. Since software is usually improved on a continuous basis, the solution to known bugs is usually available soon after it is detected. Self-healing should intervene in the task of identifying and reporting these problems, applying the necessary corrections and automatically incorporating software fixes as they are available. Software upgrades may also be done as part of preventive troubleshooting, when a problem is known to exist in the platform even if it has not caused faults.
- Network monitoring: Faults may happen at any time in any place in the network. The task of troubleshooting is often difficult to detect the affected elements and to determine the root cause. Therefore, Self-healing plays a major role in minimizing the effort and costs of troubleshooting. Again, Self-healing may play a role of preventive fault diagnosis before a root cause has triggered degradations in the service.
- Failure recovery: Once a problem is detected and diagnosed, the course of action to be taken may sometimes be difficult to plan. The task of Self-healing is to reduce this complexity by communicating with the affected elements to automatically change the settings that cause a problem or at least by determining the best solution that can be done with the lowest possible cost.

To perform all these tasks, Self-healing (just as the other SON functions) uses AI techniques. These algorithms usually take some *observations* as their inputs, and produce *actions* at their outputs. These actions are either commands to other network elements or reports that tell operators what changes should be done on the network. The observations are data that are available from many different and varied sources (such as the PM counters or PIs), that usually match those that human experts query in order to perform their tasks. In fact, Self-healing algorithms usually try to imitate human experts in order to translate the observations into actions.

Self-healing functions therefore need an additional input: expert knowledge. This knowledge is what lets the algorithms know what to do with the input data. Nevertheless, expert knowledge is neither simple to acquire (it is usually a product of many years of field experience) nor to transfer to an AI algorithm (KA [28][29][30][31]). Two approaches are used in order to fit human knowledge into AI algorithms: either manually, by letting experts tune the appropriate functions, or automatically, by using KDD [32][33] techniques. Such techniques normally require the availability of large databases of variables that experts have observed paired with their conclusions.

This thesis is focused specifically in the diagnosis functionality, that is, the identification of the root cause of a problem once the affected element is known. The development and adoption by operators of this functionality greatly depends on the efficiency of the KA processes since they are a crucial part of creating AI algorithms for diagnosis. Therefore, in this thesis, the task of KA is covered both in theoretical and practical levels. This thesis stresses the importance of KA being an unintrusive process (i.e. that does not force experts to do tasks that are out of the scope of their daily work), because otherwise it is prone to being disregarded by troubleshooting experts and ultimately leading to bad diagnosis systems that would not be commercially useful.

2.3.3 State of the art

Self-healing systems have seen a great research effort in the last years, especially in the context of the progress towards 5G networks. This increase in activity is also demanded from the industry, since the increase in competition among operators and the users demand for better service creates the inevitable need for automation of the O&M tasks. Self-healing reduces the costs caused by network faults, either by preventing them, or by fixing them quickly with minimum impact on the service.

The most studied problem in Self-healing is detection [55][9][10][11][12][56]. Detection algorithms monitor a certain number of PIs to determine when a cell is having a problem. Some algorithms that have been proposed for this task are correlation [12][57], neural networks [9] or clustering [10]. In [9] a method for improving the capability of the detection system by adding a learning functionality is proposed. In some cases, the detection task is limited to only one kind of problem; a typical example is the detection of sleeping cells [11][49][58].

Diagnosis has also raised interest in the research community [14][15][16][17][18][19][20][21][52][59], although the complexity of the task has lead to a lower number of functional systems. KBS such as Bayesian Networks (BNs) have been proposed [13] for this task. In BNs, probabilistic relations between the root causes and the symptoms (i.e. the observable effects, such as abnormal values in PIs, etc...) are represented. With these relations, the BN assigns probabilities of occurrence of each root cause for each diagnosed case. The UniverSelf project [6] proposes a diagnosis system based on a combination of BNs and Case Based Reasoning (CBR) [14][34]. The proposed method first builds a BN based on the observations of the problem, and then uses CBR (i.e. past stored cases) to optimize the BN inference process by only using a subset of the network. The COMMUNE [8][60] project uses a system based on comparing cases under study with stored instances labeled with their root causes [15]. The system holds a group of profiles of normal/abnormal values of PIs grouped by root cause, and calculates the relative frequencies of occurrence of anomalies for each group. New cases are then diagnosed with the root cause which best matches their anomaly pattern.

The processes of compensation [49][50][51][61][62][63] have been covered in several studies. Compensation is usually performed by modifying the antenna tilt of neighboring sectors so that affected users are given service by their coverage while their original cell is down.

In this study, FLCs are used for diagnosis in the LTE RAN. FLCs have been used in other processes, such as Self-optimization [47][48][53][54] and for diagnosis in other fields, such as industrial processes [64], machinery operation [65] [66] or medical diagnosis [67].

2.4 Conclusions

This Chapter introduced the most current cellular network technology (LTE) and the troubleshooting process.

The concept of SON was introduced afterwards. SON techniques automate the process of operation and maintenance in current LTE networks, greatly reducing costs and downtimes. The state of the art was also reviewed, emphasizing that, of the three SON functions, Self-Healing was the least studied.

Finally, the manual troubleshooting process in LTE networks was described, and the Self-Healing technologies that aim to automate it were described. In the next Chapter, the AI technologies that power Self-Healing are described in detail.

Chapter 3

Artificial Intelligence

In this chapter, Knowledge Based Systems (KBS) and the problem of Knowledge Acquisition (KA) in the context of Big Data will be introduced. First, the implementation based on KBS of Self-Healing functionalities described in Chapter 2 are presented. These KBS systems are built using KA techniques. A high level KA solution based on Knowledge Discovery and Data Mining (KDD) is proposed in this chapter, along with a brief introdution to Big Data systems. Both KA and Big Data concepts applied to Self-Healing will be further discussed in Chapter 4.

3.1 Decision Support Systems

3.1.1 Expert Knowledge in Decision Making

When confronted with a problem, human experts often use *heuristic reasoning* [68], that is, by using guesses or rules of thumb. One or several lines of thought are followed towards the solution of a problem and only those which look promising are followed. It is not always possible to follow a single algorithmic way of solving a problem.

An *expert* [28] is a person that has specific knowledge and experience on a specific subject (or *domain*) that allows him to perform a task effectively (with a high success rate), efficiently (as quickly and straight forward as possible), with versatility (performing well when unexpected factors come into play) and with awareness of his limitations (knowing when additional information is required or a problem cannot be solved).

Because of these capacities, experts are extremely valuable. Their expertise as *problem solvers* allows certain tasks to be performed quicker and with a lower cost. In mobile network troubleshooting, it is especially important that problems are solved as soon as possible, therefore, expert workforce is crucial. Experts are also consulted as *providers of information* when important decisions must be taken, and as *explainers* when certain events rooted on their domain occur. Troubleshooting experts therefore have important roles in other areas such as network optimization and commercial expansion.

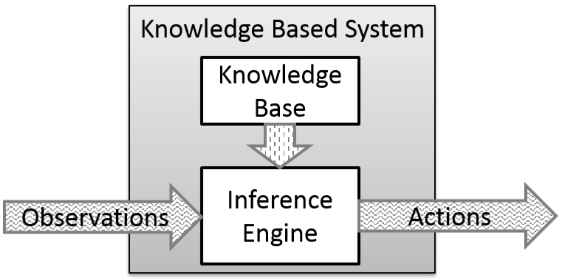


Figure 3.1: Generic KBS components

3.1.2 Knowledge-Based Systems

Expert's time is a valuable, scarce and often expensive resource. Therefore, for repetitive tasks it is a waste of resources to assign experts if the same task can be done by a machine. By automating such tasks with Decision Support Systems (DSS), the costs decrease, the process is done faster and the expert's time can be used on more challenging tasks. DSS provides algorithms that imitate the human thinking and decision making processes that may be applied to these tasks.

As stated in Section 3.1.1, experience in a specific field is the differentiating capacity of experts. Knowledge-Based Systems (KBS) [69] are DSS systems that use this expert knowledge in order to automate their work. The objective of a KBS is to obtain the same results as an expert when confronted to the same problem in the same knowledge domain. There are several key advantages of KBS over any other type of algorithm that could be used to solve a specific problem [70]:

- Separation between knowledge and its application.
- Use of specific domain knowledge.
- Heuristic approach to the knowledge and its use.

The separation of knowledge and application permits the reuse of KBS among problems that have the same structure in completely different domains. Therefore, there are two components in a generic KBS (Figure 3.1):

- Knowledge Base (KB): The compilation and/or model of the domain knowledge that will be used in the KBS. The KB contains usually domain knowledge that is highly specific for the target task.
- Inference Engine (IE): The methods that are used to apply the knowledge in order to solve the problem. The IE is usually composed of heuristic algorithms, such as rule-based systems or probabilistic methods.

Once an IE is implemented, it can be used with any KB, regardless of the meaning of its contents. The only condition is that the KB is always coded in a format that can be used by the IE. At the same time, the creation of the KB is independent of the IE, so it can be created and debugged with a minimum amount of work. The same KB can also be used with any IE that understands its format. Therefore, the IE and the KB can be created at different times, by different experts and reused in different scenarios. As it will be shown in Section 3.2, this property is key in the process of creation of the KB.

3.1.3 Knowledge-Based Systems in Diagnosis

In Section 2.3, diagnosis was introduced as a part of Self-Healing. This process is usually done by experts using domain knowledge and experience, so DSS can be used to automate it. Experts use the values of CM, PM and FM parameters as input to the diagnosis process, the output being a hypothesis of the possible root cause that is later confirmed by additional measurements or by applying the corrective actions and observing the result.

A high number of well known Knowledge-Based Systems are available for the implementation of DSS for diagnosis. Among others, Bayesian Networks (BNs) [71][13][14][34] and Case Based Reasoning (CBR) [72][14][34][15] have been used. BNs represent the probabilistic relations between the observed symptoms (e.g. unusual values on certain PIs, the activation of an alarm, etc.) and the underlying root causes. Using the Bayes Theorem, BNs calculate the probability of each root cause given the presence or absence of the symptoms. BNs have the advantage of being computationally easy to process, since only numerical computations are required. On the other hand, their structure is not similar to the though process of human experts, since they do not explicitly use probabilistic relationships with specific conditioned probabilities. CBR uses a simpler approach that fits more closely the human thought process, with the disadvantage of having a higher computational cost. In CBR, a database of previous solved cases is searched each time that a new case is diagnosed. The diagnosis of the current case will be the same as the diagnosis of the solved case that mostly resembles it. The resemblance is measured using the euclidean distance or similar methods. A large and comprehensive data base is required in order to guarantee that the most resembling case is not an instance of a completely different problem. A larger database implies a higher the precision of the CBR system, but also a higher computational cost.

In order to solve the diagnosis problem, the experts commonly use a heuristic process based on *"if ... then ..."* rules. Therefore, an appropriate way to express the knowledge is through the coding of these rules in the KB. These rules are composed of an *antecedent* (the *if* part) and a *consequent* (the *then* part). The antecedent contains assertions about the values of the inputs, such as "is equal to" or "is higher/lower than". But more often, the assertions that experts do on the values of parameters are imprecise, expressing approximate behaviours such as "is high/low" or "is normal". Therefore, another aspect that should be coded in the KB is the mapping between exact numerical values and imprecise descriptive behaviours. Fuzzy Logic Controllers (FLCs) [22] are KBS that match this way of working. In Section 3.1.4 a detailed description of FLCs is given.

To imitate diagnosis experts, DSS will predict a possible root cause based on the value of the inputs. In order to assess the quality of these predictions, some figures of merit must be defined. In the case of diagnosis, three measurements are important:

- Diagnosis Error Rate: number of incorrect diagnosis over the total number of problems, excluding the problems that are not diagnosed (i.e.: are labelled as being normal) and the false positives. This measurement indicates the accuracy of the diagnosis system. It is given by:

$$E_d = \frac{N_{de}}{N_p} \tag{3.1}$$

Where N_{de} is the number of wrong diagnosis and N_p the total number of problems. In the case of multiple diagnosis, N_{de} will be increased by a fraction that depends on the number of simultaneous diagnoses (i.e. the number of equally activated rules). The increase in the total number of errors for each diagnosed problem is given by $\Delta N_{de}(i) = 1 - (1/N_d(i))$ where $N_d(i)$ is the number of diagnoses for problem i. In this scenario, N_{de} is given by:

$$N_{de} = \sum_i \Delta N_{de}(i) \tag{3.2}$$

- Undetected Rate: number of problems that are not diagnosed at all over the total number of problems. This measurement indicates the ability to detect a problem, that is, the reliability of the diagnosis system. It is given by:

$$E_u = \frac{N_{un}}{N_p} \tag{3.3}$$

Where N_{un} is the number of problematic cases that are labelled as normal.

- False Positive Rate: number of normal cases that are diagnosed as a problem over the total number of normal cases. A high False Positive Rate indicates that there is a high chance of false alarms. It is given by:

$$E_{fp} = \frac{N_{fp}}{N_n} \tag{3.4}$$

Where N_{fp} is the number of normal cases diagnosed as having a problem and N_n is the total number of normal cases.

Note that $E_p = E_d + E_u$ constitutes the total error rate over the input problems (that is, the probability that a problematic case in the input of the diagnosis system produces a wrong diagnosis in the output). The overall error (that is, the probability that a specific diagnosis is wrong) is given by:

$$E = P_n \cdot E_{fp} + P_p \cdot E_p \tag{3.5}$$

Where P_n and P_p are respectively the proportion of normal and problematic cases. In a normal live network, it is expected that $P_n >> P_p$ for an acceptable service, therefore even a small E_{fp} can greatly degrade the diagnosis of the KBS. A large number of false alarms will decrease the usability of the diagnosis system. The probability that a given positive diagnosis is a false positive (the complementary of the Positive Predictive Value) is given by:

$$P_{fp} = 1 - PPV = \frac{P_n \cdot E_{fp}}{P_n \cdot E_{fp} + P_p \cdot (1 - E_u)} \tag{3.6}$$

When $P_n >> P_p$, $P_{fp} \to 1$, so a separate *detection* system must be added that separates normal behavior from problematic cases. This detection system would reduce P_n, and therefore P_{fp}. The addition of such a system also changes the meaning of E_u, since cases that are not assigned any diagnosis are still detected as problematic. Therefore E_u becomes the proportion of problematic cases that cannot be diagnosed (instead of being completely ignored and marked as normal).

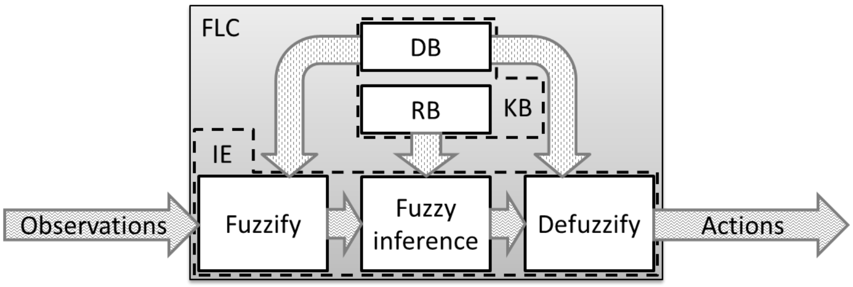


Figure 3.2: Fuzzy Logic Controller

3.1.4 Fuzzy Logic Controllers

Fuzzy Logic [23] is a branch of artificial intelligence modelling human thinking. To do this, it transforms numerical values of variables into descriptive values (such as *"high"* or *"low"*). FLCs are KBS that use the principles of fuzzy logic to assign values to output variables based on some input values. FLCs are often used in control systems where the input variables reflect the state of the system and the output variables are control actions. Therefore, FLCs are a good implementation for a diagnosis system that takes PM, FM and CM variables as an input and returns a list of possible diagnoses. Figure 3.2 shows the diagram of a generic FLC. For diagnosis, the observations would be the CM/PM/FM variables and the actions, the diagnosis. The KB of an FLC is divided into two parts:

- *Data Base* (DB): contains information about the input and output variables.
- *Rule Base* (RB): contains heuristic *"if ... then ..."* rules.

An FLC applies three consecutive processes in its IE: *fuzzification* that converts *crisp* numeric values to *fuzzy* descriptive values, a *fuzzy reasoning* that assigns fuzzy values to output variables based on fuzzy values of input variables, and *defuzzification*, that transforms the fuzzy value of the output variable into a crisp value.

To transform normal crisp values to fuzzy values, several *fuzzy sets* (S_1, S_2, ...) are defined over the domain of the crisp variable. A crisp variable is a common numerical non-fuzzy variable, and its domain is the *universe of discourse* (U). A fuzzy set comprises the values of U that have a common characteristic as perceived by a human (for instance, *"high"* or *"low"* values for a PI). Each fuzzy set S_i has an associated *membership function* $\mu_{Si} \in [0, 1]$ that defines the degree of truth of each crisp value belonging to that fuzzy set. Membership functions are defined and stored in the DB of the FLC. In Figure 3.3 an example of how membership functions work is depicted. A crisp variable x has two fuzzy sets defined on its domain: S_1 and S_2. For a given crisp value x_1 of x, the membership degree for each set is given by $\mu_{S1}(x_1)$ and $\mu_{S2}(x_1)$, the membership functions of sets S_1 and S_2, respectively. A *fuzzy variable* is formed by the linguistic labels identifying a fuzzy set and their membership functions. In Figure 3.3, the fuzzified value of x_1 is "S_1 in $\mu_{S1}(x_1)$ degree and S_2 in $\mu_{S2}(x_1)$ degree".

For the purposes of diagnosis in LTE, two fuzzy sets will be defined for each PI, representing *"low"* and *"high"* values. The membership functions for each set will be trapezoidal, similar to those depicted in Figure 3.3, where S_1 is the *"low"* set and S_2 is the *"high"* set. Both membership functions share two points:

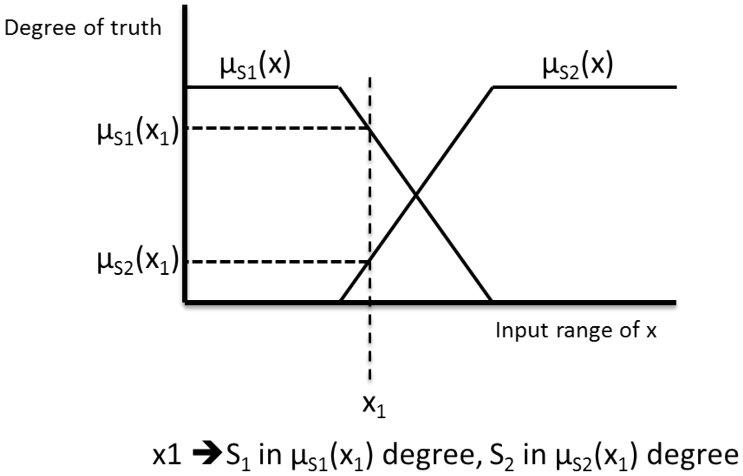


Figure 3.3: Fuzzification process using the membership functions

- *Low threshold*: below which $\mu_{low}(x) = 1$ and $\mu_{high}(x) = 0$
- *High threshold*: above which $\mu_{low}(x) = 0$ and $\mu_{high}(x) = 1$

For some KPIs, the *"high"* values are normal and the *"low"* values will represent a degradation, whereas for others the opposite is true.

Fuzzy reasoning in a FLC is done through *"if ... then ..."* rules, similar to those used by experts in the diagnosis process (Section 3.1.3). Just as expert rules, these *fuzzy rules* are composed of two main parts: the *antecedent* and the *consequent*. The degree of truth of the consequent is obtained by calculating the degree of truth of the antecedent.

The antecedent contains assertions about input variables belonging to fuzzy sets (for example "x_1 is S_1"). The degree of truth of these assertions is the degree of membership of the variables ($\mu_{S1}(x_1)$). Several assertions can be done in the same antecedent, joined by *AND* or *OR* operators. Usually in these cases, the degree of truth of the antecedent is the minimum or product of all the individual assertions (with *AND* operators) or the maximum (with *OR* operators).

The consequent contains an assertion about an output variable. The degree of truth of the antecedent modifies the membership function of the fuzzy set of the value assigned in the assertion, either by truncating it or by obtaining the product. The full process of assigning a degree of truth to a variable in the consequent based on the degree of truth of the antecedent is depicted in Figure 3.4. The antecedent of the rule has two assertions: "x is S_1" and "y is S_3". The minimum degree of truth of both assertions (in this case, $\mu_{S3}(y_1)$) is assigned to the consequent. The degree of truth of the consequent truncates the membership function $\mu_{SO1}(z)$ of the fuzzy set S_{O1} assigned by the assertion "z is S_{O1}".

The crisp value of an output variable z can be inferred through the aggregation of the outputs of individual rules. A truncated membership function $\mu_{SO1}^{(T)}(z)$, $\mu_{SO2}^{(T)}(z)$, ... is obtained for each rule on the domain of the output variable according to the results of the linguistic reasoning. For each point of the domain of the output variable, the maximum degree of truth among the output membership functions is taken, that is a combined function $\mu_O(z)$ is defined as

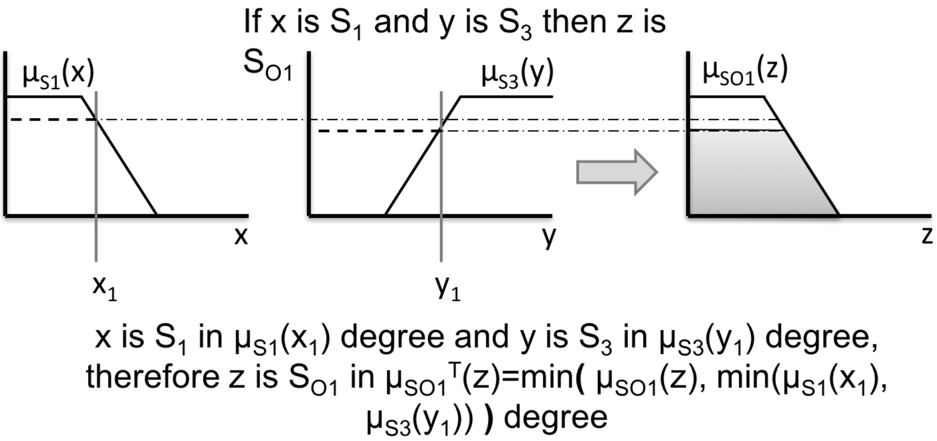

If x is S1 and y is S3 then z is SO1
μS1(x)
μS3(y)
μSO1(z)
x
y
z
x1
y1
x is S1 in μS1(x1) degree and y is S3 in μS3(y1) degree, therefore z is SO1 in μSO1T(z)=min(μSO1(z), min(μS1(x1), μS3(y1))) degree

Figure 3.4: Linguistic reasoning with fuzzy rules

$\mu_O(z) = max(\mu^{(T)}_{SOi}(z))$. A crisp value is then obtained from this function in the *defuzzification* process. The crisp value can be taken according to a specific policy:

- *SOM* (Smallest Of Maximum): The smallest point with the maximum degree of truth.
- *Centroid*: The average of all points with the maximum degree of truth.
- *LOM* (Largest Of Maximum): The largest point with the maximum degree of truth.

Figure 3.5 depicts this process. The output of two individual rules (functions $\mu^{(T)}_{SO1}(z)$ and $\mu^{(T)}_{SO2}(z)$) are aggregated, creating a combined membership function. The crisp value for variable z is one of the points where this new function is maximum. z_1, z_2 and z_3 are the values for z if the defuzzification method is SOM, centroid and LOM, respectively.

In diagnosis, the output variable will contain several *delta* membership functions, each one representing a distinct root cause (Figure 3.6). After aggregating the output and obtaining the combined membership function, the tallest delta will mark the selected diagnosis. In case of draw, the policy (either SOM or LOM), will decide which diagnosis is chosen. A modification of a traditional FLC can also be done, and select all the diagnoses that have the highest score. In this case, the system will be able (either correctly or incorrectly) to return multiple diagnoses. Another option is to extract the aggregated function without defuzzifying, and obtain a list of the diagnoses ordered by the height of their deltas as their weight.

3.2 Knowledge Acquisition

KBS use knowledge and replicate the work of the experts in a certain domain. Since computers have a high processing power, they may even outperform experts in their own work where complex calculations and data analysis are required. Nevertheless, they cannot do this by themselves; the knowledge must be inserted into the KB (*elicited*) somehow. The process of collecting expert knowledge and compiling it into a KB that can be used by a KBS is called Knowledge Acquisition (KA). KA has traditionally been described as a *knowledge transfer* process that is performed manually. Several manual KA techniques have been used:

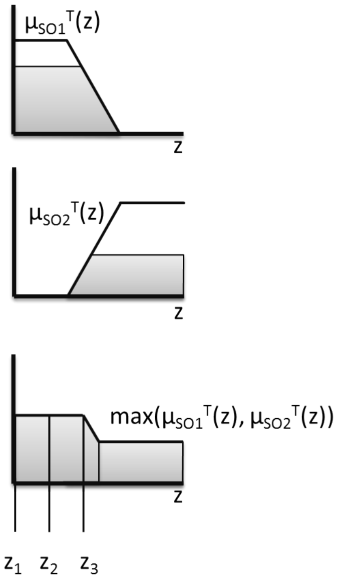


Figure 3.5: Defuzzification process

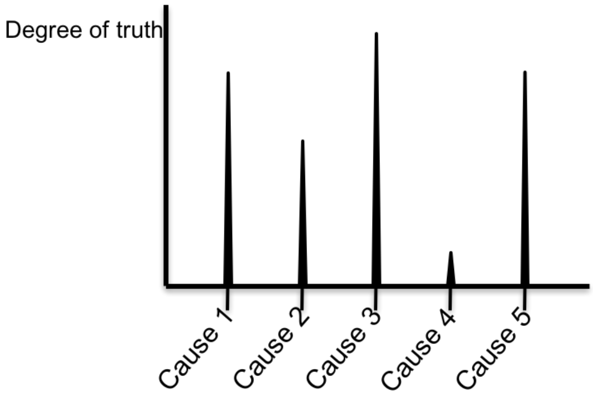


Figure 3.6: Aggregated membership function of the diagnosis output variable

- By interviewing the experts. This option involves a *knowledge engineer* [29] who has the knowledge about the DSS operation and translates the knowledge that the expert provides into the appropriate format. It is both a time consuming and intrusive process for the expert and it requires an additional specialist.
- By teaching the expert the particularities of the DSS so that the knowledge is directly coded by him. It is even more intrusive than the previous option and involves a steep learning curve.
- By using forms that simplify the insertion of the KB parameters. This is a semi-automated process that substitutes the knowledge engineer with a computer program that has a friendly interface. This approach is used in [31], where a Knowledge Acquisition Tool is described. This tool will ask troubleshooting experts for the conditioned probabilities that define a BN, without the need of them knowing the details of how these values must be used in the underlying system.

The CommonKADS [73] method contains a full methodology for creating complex KBs, involving the manual creation of models of different types of knowledge. CommonKADS is the reference for traditional KA, since it has a full description of all the involved steps.

All these methods of KA rely heavily on the expert's involvement. Unfortunately, in the industry of mobile communication network management, the time of the experts is a scarce resource. Therefore, the attempt of creating a RB by traditional KA is bound to failure.

The *knowledge modelling* approach used by CommonKADS can be extended with Data Mining (DM). In this case, the KA process consists of extracting and modelling the knowledge implicitly contained in the byproducts of the work of experts with DM. The DM process takes as inputs the observations done by experts and their results, and creates a model of the knowledge that can be used as a KB. This approach requires a much lesser involvement from experts, since they are only required to provide data, instead of explicitly asking them for the knowledge. [74] explores the information that can be extracted from a CBR database in order to model relations among PIs, alarms and root causes for 3G networks. [75] proposes a hybrid between CBR and rule based system for diagnosing problems in web services. The CBR part covers the cases for where rules are not available. New rules are then generated as new cases accumulate in the CBR database; having the same effect as performing KDD over a set of solved cases for KA.

3.3 Knowledge Discovery and Data Mining (KDD)

As stated in the previous section, DM can be used to extract a KB as a model of a certain database of inputs and expected outputs of the KBS. More generally, DM is part of the Knowledge Discovery and Data Mining (KDD) field. The purpose of KDD is making sense of raw data through data analytics; therefore, it covers all the phases starting from reading raw, untreated data, to a representation of the knowledge that can be used by human operators or a KBS. The applications of KDD are found in a broad variety of fields where data is generated and knowledge required, such as in electronic commerce [76][77] and banking [78], sensor networks [79][80] and, in the case of this thesis and some other studies such as [38] and [39], mobile communication networks. For KDD to be applied, it is required that a stable collection system gathers enough data so that the target knowledge is completely contained.

A KDD system includes models of the formats that data takes during the knowledge extraction process and the techniques that transform one format into the next one. There are five main sequential steps in the process of knowledge extraction:

- Selection: the set of data where the target information is contained must be provided to the system, so only the relevant information is processed. The input of this process is all the available raw data, and its output contains only the relevant raw data. In troubleshooting, two dimensions must be considered: time and place where the problem occurred. For detection, the time frame is usually only the last few measurement intervals (each measuring normally 15 minutes), whereas diagnosis may need to look more into the past for the last hours/days. The place where the problem occurs is usually given by the sector or the eNodeB. Additionally, the data sources (location where troubleshooting data is stored) must be given to the system. This is usually done in the installation process of the KDD system, where the connections and the access permissions are configured.
- Preprocessing: once the selected data is extracted, it must be treated in order to improve its quality, by removing noise and filling missing data that may produce bad results. The output of this process is clean data that has no missing points or outliers. In the large and complex data gathering systems of cellular networks, missing data due to connectivity problems or bad configurations is a common problem.
- Transformation: since the data comes in a large variety of formats, the process may need to unify or change it in order to adapt it for the DM process. Also, in this step, a data reduction algorithm may be needed. Data reduction eliminates superfluous information and therefore the processing power required by the DM process. The output of this step is simplified, normalized and properly formatted data.
- Data Mining: DM is the process of "making sense" of the data, that is, the step that extracts the target knowledge. The nature and implementation of the specific algorithm, as well as the formats of its inputs and outputs depends heavily on the purpose of the algorithm. The output of this step is a model of the input data. It is no longer raw data, but the core information contained in it, a manageable model of the knowledge.
- Interpretation/Evaluation: the model of the knowledge is used to extract concrete insights on the input data. In the case of network troubleshooting, the results of the previous stage will directly be used on a KBS to perform diagnosis.

The KDD process can be applied to automate KA; in this case, the input is the information that is available from the experts and the output is a KB that can be used in a KBS. Since expert's time is a scarce resource, one of the main focuses of the KA process is to reduce the time and effort required from them. Therefore, some restrictions must be applied when designing the process. These restrictions will be discussed in detail in Chapter 4.

The input to the KA system are therefore instances of solved troubleshooting problems (*solved cases*). Each instance must contain values of PM, CM and FM for the duration in time where the problem was observed, and a label representing the diagnosis given by the expert for that particular case. Since DM finds common patterns and discards the rest of the information, many varied instances of problems caused by each root cause must be available.

3.4 Big Data

As a consequence of the decrease in the prices of storage hardware, as well as the increase in bandwidth in mobile networks and the growth in the number of connected electronic devices, the volume of data generated by our society is exponentially increasing. All these data contain information about a wide spectrum of aspects that may be interesting for all types of businesses. For instance, the analysis of web searches or Twitter hashtags is commonly used for determining social trends, traffic information is used to plan the fastest route between two places and email exchanges among experts are used to extract knowledge.

As produced data grow, the information contained in them increases in two ways:

- Quantity: the number of samples taken for measuring a specific aspect is larger and easier to take. Topics that previously required a big effort and specific data gathering actions can now be easily studied and with a low cost. For instance, the measurement of TV audience of a program is traditionally done via interviews or special measurement equipment distributed to a randomly selected sample of users. The advent of social media has enabled new ways of measuring the viewership of TV programs on a wider audience in real time without any need of interaction with the subjects.
- Detail: the range of topics covered by the information is wider; aspects that used to remain hidden are now revealed. Continuing with the example of TV audiences, social media contains real time information not only of the number of viewers, but also of their reactions.

These new possibilities are very interesting for businesses in order to better understand and operate in the market. But all of these advantages come at a cost. The amount of available data is often too big and unstructured to be treated with traditional statistic methods to achieve the results and speed to cater for the needs of the market, so new techniques must be used. *Big Data* is the new paradigm that encompasses the principles and techniques for making sense of data in this new scenario. A data set is considered Big Data compliant when it follows a set of principles known as the *3 V's* [40] of Big Data:

- Big Volume: the quantity of data that must be processed is large, either because there are many individual small information units or because each information unit is large. The exact boundary for the volume of data to be considered Big Data is largely dependent on the application, the time constraints and the available hardware.
- High Velocity: the information is produced at a rate that requires special techniques in order to process it before new data is produced. Again, there is no exact boundary to consider a data source as fast; it depends on the hardware resources that are available.
- High Variety: the data sources have varied formats (that require a preprocessing for homogenization or altogether separate processing pipes), often contain unstructured data (that requires a preprocessing in order to structure it so it can be processed) and are extracted from different physical/logical interfaces (requiring special equipment or hardware drivers). Also, the data units may contain information about different entities that may or may not be needed for a specific application.

Data that complies with these features is complex and difficult to process. Big Data makes use of the latest software and hardware developments to achieve a high data processing power in order to extract the useful information in a reasonable time. Data sets that are voluminous are divided into smaller chunks that can be processed and stored by a network (or *cloud*) of interconnected CPUs instead of a single processor. Big Data makes heavy use of distributed computing for two main purposes:

- Cloud Computing: the processing is done in a remote location, that is “in the Cloud”, which contains the hardware resources. These resources are dynamically allocated depending on the requirements of the client. For small requests, a small amount of physical processors is allocated, whereas for large processes, a grid of processors working in parallel is dedicated. The whole process is transparent for the clients. When the processing power is large (as is the case in Big Data), the algorithms in the cloud must be parallelized. It is important to note that all the KDD steps described in Section 3.3 are Big Data problems in mobile network troubleshooting, and they must therefore have a parallelizable implementation. Some *de-facto* standards are the Lambda system architecture [81] for the high level design of Big Data systems and the MapReduce [82] programming architecture for low level algorithm design.
- Data Warehousing: data storage is distributed “in the Cloud” instead of in a single server. Distributed storage has many advantages over centralized storage, such as a higher reliability and faster storage and retrieval times. Several technologies improve the storage performance even more: new formats that are optimized for the unstructured data (NoSQL [83] databases) or in-memory databases that greatly speed up the storage and retrieval of data.

In Chapter 4, the automation of KA for LTE Self-Healing will be reformulated as a Big Data problem, and the requirements that this introduces in the design process will be considered.

3.5 Conclusions

This Chapter has introduced the AI concepts involved in the Self-Healing process (specifically diagnosis). Firstly, the need for DSS was introduced to assist the experts in the process ofdiagnosis. The role of expert knowledge was described and KBS were introduced as the type of systems that use it to perform actions such as diagnosis. The application of KBS to diagnosis was shown in full detail through the description of FLCs, which is the system that will be further used for diagnosis in this thesis. The concept of KA was described to transform expert knowledge into a specific format that may be used by KBS. In this thesis, KDD is proposed as a method for KA, as an alternative to traditional manual KA. Finally, the Big Data paradigm was introduced, showing the requirements of a problem to be considered a Big Data problem. In the next Chapter, the data used in diagnosis will be described in full detail, showing how it complies with the described Big Data principles.

Chapter 4

Knowledge Acquisition for Diagnosis systems in LTE networks

In Chapter 3 the concept of KA was introduced as the process of generating a KB that contains the knowledge of field experts. Two different approaches to KA were described; the *transfer* approach, where the knowledge was elicited from the experts and manually introduced into a KB, and the *modelling* approach, where a KB was extracted from a large data set where the knowledge was contained. The second approach, which has the advantage of requiring a small degree of involvement from the field experts, can be implemented using KDD systems.

For automating diagnosis in LTE, KBS can be used. Specifically, the use of FLCs is described in Section 3.1, along with the format of the KB that needs to be generated. The KA process based on KDD has two inputs: the network data and the experts analysis of the data. In this Chapter, in Section 4.1, the type of data available in LTE for the KDD process that will extract the knowledge and generate the KB will be analyzed and characterized as Big Data (Section 3.4). Section 4.2 will then propose a tool that will let experts insert their analysis of the data in the KA process (in this case, their diagnosis of the problem).

4.1 Formulation of Self-Healing data as Big Data

This Section will describe in detail the data that is available from LTE networks. These data are used by troubleshooting experts for detecting and diagnosing problems. In Chapter 5, the data extracted from real networks will be used to create a model that can later be used to emulate problems and use them in DM algorithms.

These data contain the *information* that reveals the presence of a problem. In order to find that information, KBS use the *knowledge* that is contained in their KB, which is generated based on the experts knowledge. The KA process will therefore collect the data that experts observe and, along with their diagnosis, use it to extract the expert knowledge.

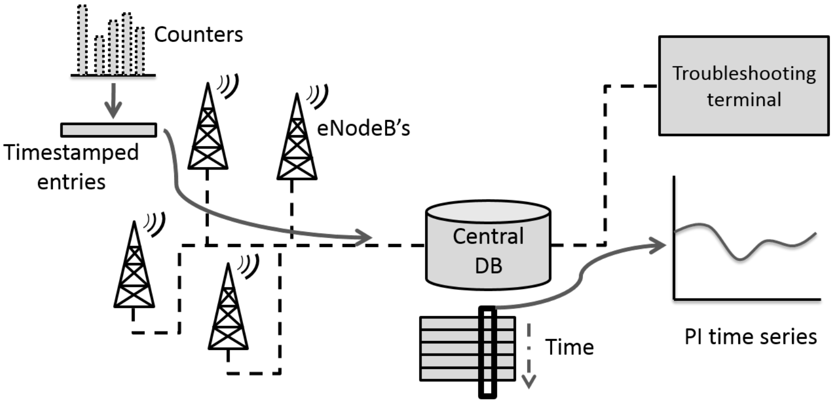


Figure 4.1: Architecture of the PI data collection system

4.1.1 Data Sources

The troubleshooting process uses data that indicate the state and behaviour of the network. These data are usually recorded on-site in the eNodeB, which is connected to a data collection subsystem (depicted in Figure 4.1) that regularly gathers all the information in a centralized database.

Several types of data sources are present in the network. They are collected and stored at different network elements and accessed by the data collection subsystem. The most commonly used data sources for troubleshooting are:

- Performance Management (PM) metrics: each eNodeB keeps an array of counters that increase with specific events, such as established or dropped connections. These counters are accumulated over a variable time period known as Report Output Period (ROP), that is usually 15 minutes. Other measurements are also taken and averaged during this time interval, such as the *received power* or the *instantaneous CPU load* of the eNodeB.
- Fault Management (FM) alarms: along with the counters, eNodeBs monitor specific problematic events. The occurrence of these events is registered in a binary indicator (i.e. the alarm). The nature of alarms is varied, such as software errors or hardware integrity problems.
- Configuration Management (CM) parameters: the configuration parameters of each eNodeB are adjusted by the engineers or SON functions. These parameters regulate the network operation, so they are important information sources for better understanding how the events are affecting the network performance.
- Call Traces: measurements taken in the time interval and channel where a communication takes place. Each call trace contains registers (such as counters and alarms) and measurements related to a specific connection (session) between a UE and the network.
- Others: other information sources that are sometimes used in the troubleshooting process are trouble tickets (previously known problems of the affected sector or neighboring sectors), engineer actions (time-stamped actions taken toward solving a previous problem or optimizing the performance), drive tests (on-site measurements of the radio signals received by the UEs), customer complaints, etc.

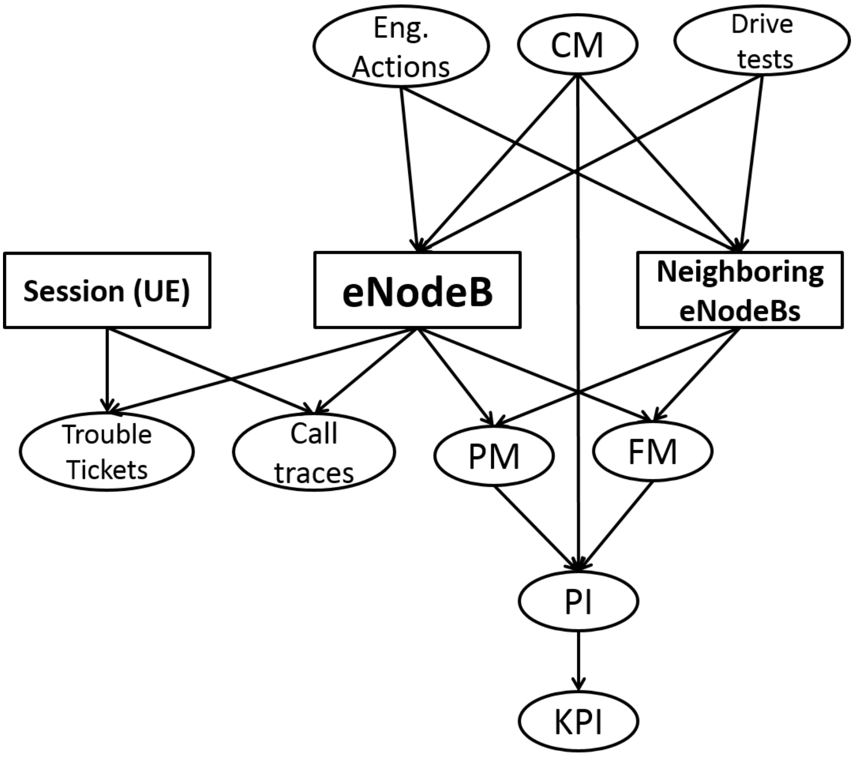


Figure 4.2: Data types and their relations

Figure 4.2 shows the data types and their relations. The PM, CM and FM data are reported to a centralized location each ROP, where the values are stored in a time-stamped database. The aggregation method of PM counters will depend on the measurement that is being taken. For event counters, the number of occurrences is added, whereas for counters that monitor a magnitude (such as the received signal strength), the average is taken. CM parameters, on the other hand are usually fixed for each ROP, and FM alarms change only once if they are triggered. Therefore, for each ROP, a time-stamped vector is saved for each eNodeB in the centralized database. These values are then used for calculating Performance Indicators (PIs), that are composite variables that may contain information from several counters, configuration parameters and alarms. PIs can then be aggregated on several time levels (ROP, hourly or daily). They measure high level magnitudes, such as the *connection establishment success rate*. But there are also some PIs that measure simple low-level magnitudes, by reproducing the value of a counter (such as the *number of connection attempts*s).

A reduced subset of the PIs reflects the general behaviour of the node; showing a degraded behaviour when the performance of the network is degraded. These are the Key PIs (KPIs), a set of high level PIs that experts use to detect which cells must be further screened to find a problem. KPIs are also aggregated in space, that is, over a group of eNodeBs, to reflect the general behaviour of the group. This organization lets experts do a top-down approach on problem detection (as described in Section 2.3); they start by observing the KPIs of the whole network and then drill down on specific groups or individual eNodeBs by using a list of worst offenders to find the problematic spots.

	PI 1	PI 2	...	PI N
...				
T - 2				
T - 1				
T				

Figure 4.3: Format of the PI data stored in the central database

In the centralized database, for each eNodeB, a time-stamped table with all these measurements is saved. Therefore, each PI, KPI, PM, CM and FM value is represented as a time series, that is, each entry in the fault database is a matrix with two dimensions (the measurements and time). This is the format of the performance data that experts observe to troubleshoot problems (Figure 4.3). While a graphical representation of a time series is a very appropriate format for a human expert that will study each PI separately, the time dimension adds some complexity when the data is processed on a KBS that analyzes all PIs in a parallel manner. In this scenario, a KBS would need to analyze a large number of time series (one per PI) in order to obtain a diagnosis. This would either increase the processing time or the required hardware, ultimately increasing the costs of operation. Therefore, a method to reduce the dimensionality of the data would be required. Such a method should ideally eliminate the time dependency with the minimum loss of information. Traditionally, troubleshooting experts work with aggregated values to eliminate time dependency. The average of the values over a period is taken (hourly or daily), which can potentially cause the loss of information on degradations since a higher proportion of normal values may mask the abnormal ones. The complexity of the fault database may be much higher if other performance measurements are included, such as call traces or trouble tickets. In that case, an additional layer of data formatting that converts all the information to a single format would be required.

All these data create a detailed picture of the state of the network. Any event is somehow reflected in these data sources. When a problem (the root cause) occurs, it may cause the degradation of one or several of these indicators. An indicator is considered to be *degraded* when its value is either too high or too low compared to what it should be if the operation of the eNodeB (or group of eNodeBs) was normal. To determine if a value is degraded or not, experts use thresholding (Section 2.3), although usually with imprecise thresholds (Section 3.1.3).

Normally, it is a combination of problems and circumstances that cause a visible effect; therefore, some root causes may be present at a specific moment and not cause degradations. In that case, the problem may be undetectable. It is important to take into consideration this fact when working with aggregated values. If an indicator suffers a degradation for a brief interval of time (for instance, 1 hour), and its value is aggregated over a much longer period (for instance 1 day), it is possible that the degradation is masked, that is, the normal values compensate for the degraded value. When observing the aggregated value, no degradation would then be detected at all. The same applies for spatial aggregation; a degraded indicator of a specific eNodeB may not cause a visible effect on the aggregated indicator over the whole group. For this reason, drill down lists (i.e. lists that show the values of KPIs for the individual eNodeBs of a group) are always important, regardless of the global indicator being degraded or not.

4.1.2 Data Dimensionality

The data sources that are used both for troubleshooting (either manual or automatic) and KA were described in Section 4.1.1. In this Section, the dimensionality of that data will be discussed, to demonstrate that automatic diagnosis and KA can be classified as Big Data problems. In Section 3.4, it was stated that a dataset could be considered Big Data if it had three properties: big volume, high velocity and high variety. These three aspects can be found in the data collected from the network:

- Volume: there are many individual small information units (such as PM counters that are very simple and structured, but are generated by a large number of eNodeBs) and also less numerous but large information units (like call traces that contain variable fields, many measurements and the information is potentially referred to several different eNodeBs). In LTE networks, for each ROP (every 15 minutes), each sector of an eNodeB produces and transmits a vector to the centralized database. Each of these vectors has PM counters, CM parameters and FM alarms. The specific number of values depends highly on the manufacturer of the equipment, but the number is in the hundreds or thousands. That number must be multiplied first by the number of sectors per eNodeB (normally 3 or 6) and then by the number of eNodeBs in the network (that ranges in the thousands). With these numbers, the volume of CM, PM and FM is several million of values per ROP. Other measurements add even more volume; for instance, each of the millions of individual connections occurring in the network generates a call trace, that on itself may contain hundreds of variables.
- Velocity: In an LTE network, the data is generated every ROP. Therefore, the whole data must be collected, stored and processed during this time interval. It is crucial to have quick results that help to prevent severe degradations in the performance of the network. In the case of troubleshooting, the whole processes of detection, diagnosis, recovery and compensation must be done before users perceive a severe loss in QoS. The time frames for this range between minutes and hours. However, currently, with manual troubleshooting, it is common that problems take days to be resolved.
- Variety: As shown in Section 4.1.1, there is a variety of data sources, each having its own formats and rules. For instance, PM and CM variables are given as numerical values, whereas FM variables are boolean. Call traces register individual events, and for each event, a different set of measurements is provided. Other variables, such as trouble tickets or user complaints are much more complex, and may contain important information for troubleshooting.

It is therefore necessary to apply specially developed algorithms that take into account the dimensionality of the data.

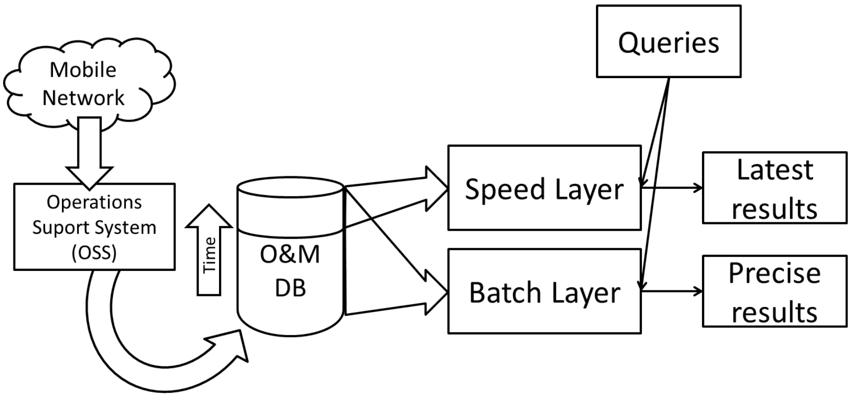


Figure 4.4: Lambda architecture

4.1.3 Big Data techniques applied to Self-Healing

In Section 3.4, the Lambda architecture was introduced as one of the de-facto Big Data technologies for cloud computing. Figure 4.4 shows the Lambda Architecture. This architecture is specially designed for data sources where information is being accumulated over time and live results are requested. The Lambda Architecture has two data pipelines:

- Batch Layer: in this layer, the algorithms are applied over datasets spanning over a long period of time to extract detailed information. This pipeline is slow, since it deals with large volumes of data, so it does not offer immediate results when new data is added. Nevertheless, it must still be fast enough to process the data at the rate that it is produced.
- Speed Layer: it is often necessary to obtain an approximate result that is available immediately after new data is collected. This pipeline processes the latest data along with a data set spanning over a small period of time into the past. It can also use the output of the batch layer for previous periods of time. The results of the Speed Layer tend to be more inaccurate and prone to errors (due to missing data, input errors, etc.), since the focus is always on speed. In environments where a proactive detection of problems is performed, the early approximate results of the Speed Layer are vital.

This architecture can be adapted to the processing of network data for automatic troubleshooting and KA. Normally, troubleshooting is more time sensitive and since it deals only with the most recent data, it can be easily optimized for speed. Therefore, the KBS that performs diagnosis can be integrated in the Speed Layer. On the other hand, KA deals with a much larger dataset, spanning over a long time in order to find examples of each problem. At the same time, precision is more important than speed, since a good KB will result in a high performing KBS for the diagnosis.

In order to be able to deal with the data, its dimensionality must be reduced at some point. In the case of LTE data, the temporal dimension increases the complexity of the data and the required processing power. Therefore, a crucial action is to reduce the time series of each PI (or KPI, CM, PM, FM, etc) to a reduced set of single values that capture the essential information. In Chapter 5, this step will be further discussed and an implementation will be given. Also, redundant or superfluous variables must be eliminated from the processing pipelines.

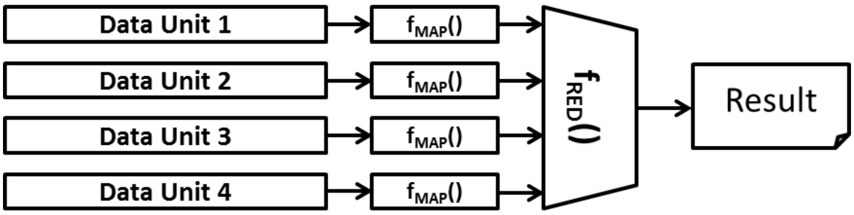


Figure 4.5: MapReduce parallelization

As exposed in Section 3.4, Cloud Computing is a key technology in the Big Data ecosystem. Cloud Computing makes heavy use of parallelization to reduce the processing time, specially when big datasets are involved. The time that a process may take is reduced roughly by the factor of the number of independent processors on the computing cloud where the process is run. A common architecture for parallelization, which can be applied in cellular networks, is MapReduce (Figure 4.5), where the data is processed using two functions. The *map* function is applied over each data unit that is independent (that contains the required information without need of other data units) in a separate process in the computing cloud. The outputs of the map function over each data unit is then aggregated with the *reduce* function.

For an algorithm to be parallelizable, its design must guarantee that the final result is the same when it is run as a single process and when the task is divided among multiple instances. The input is first divided into subsets that keep coherent data. Each subset is then processed independently, without interactions among the processes. Ideally, each process is run on a separate processor, therefore minimizing the execution time. Finally, the results of the independent instances of the algorithm are aggregated to obtain the output.

4.1.4 Use cases

This section presents some Self-Healing examples and how they can be addressed by means of Big Data techniques.

Sleeping cell detection

A common problem in mobile networks is *Cell Outage* or *Sleeping Cells*, that is, cells that should be providing service but are not doing it at all for some reason. In scenarios where the density of cells is high, this problem is specially hard to detect, since the users are redirected to neighboring cells. Sleeping cells produce a low QoS, since the optimal cell for the affected users is not being used. This is usually done using some availability KPIs and alarms that indicate that the cell is down. But in a major outage where the whole eNodeB is down and not responding to status queries (such as a major software fault or power outage) the network operators will not be able to detect the fault quickly. An alternate approach to outage detection is using the neighboring eNodeBs measurements to detect the outage by calculating its impact. Since in a network the number of eNodeBs is normally large, and for each one, all the data from its neighbors is used, it is easily determined that this is a Big Data problem. Moreover, the timeframe to detect and correct an outage is usually low, since the service for the affected users is degraded, leading to a

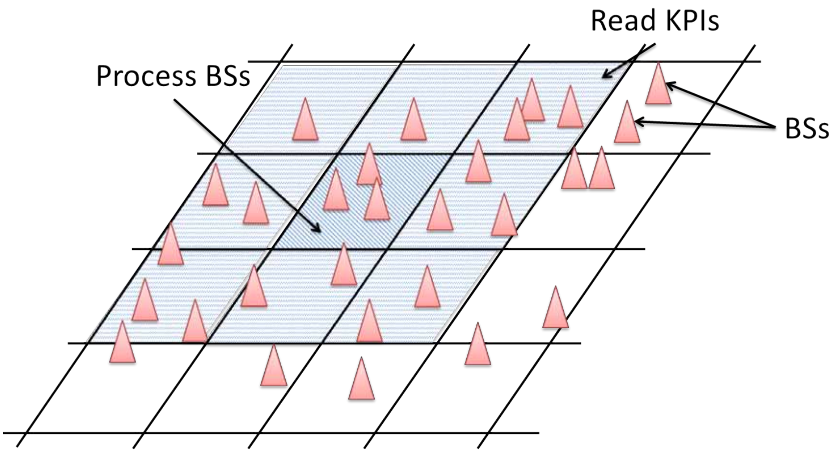
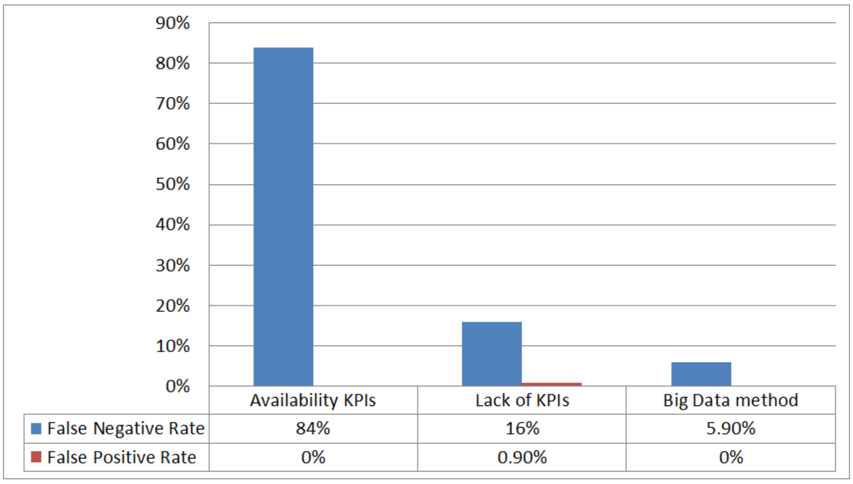


Figure 4.6: Terrain division for the detection of sleeping cells

bad user experience.

In [56], an algorithm for detecting sleeping cells based on the decrease of handovers with neighboring cells is described. To find sleeping cells, for each eNodeB, the number of incoming handovers for the current and previous ROP is aggregated from the neighbor eNodeBs outgoing handovers. If the handovers have suddenly dropped to zero and the readings of other PIs (or the lack of PIs) of the cell indicate a malfunction, the cell is marked as a sleeping cell. To apply this algorithm under the Big Data principles, it should be considered that the full network has to be analyzed over a limited time (one ROP, before new data is received). Thus, it is essential that multiple instances of the algorithm analyze separate parts of the network. In order to achieve this, the terrain can be divided in partitions that are the size of the maximum distance between neighbors, as shown in Figure 4.6. Each instance of the algorithm tests sequentially each of the eNodeBs contained in one partition by looking into the data of its neighbors, that are contained in the adjacent partitions. Therefore, each parallel instance only works with a reduced database containing only the data of the current and adjacent partitions. An alternative partitioning can be done by taking neighboring relations into account; in this case, instead of reading the KPIs of all the eNodeBs in the adjacent partitions, only those that are configured as neighbors are considered. This will increase the initial processing time for creating and delivering the tasks to the different processes, but with the benefit of reducing the potential amount of data that each one must read.

Figure 4.7 shows the results obtained in [56] for the algorithm compared with other methods when applied to a simulated LTE network: Availability PIs (the detection is made by monitoring certain PIs of the cell) and Lack of PIs (a cell is selected as sleeping cell if there are no PIs available). For each method, the results show the False Positive Rate and the False Negative Rate (i.e. the percentage of non detected cases among the total of normal cases simulated). The results show that the proposed method is able to detect most simulated outages, leading to a low percentage of false negatives (5.9%), while Availability PIs and Lack of PIs methods present a high percentage of false negatives. These results show that the increase in the volume of processed data improves the detection capacity.



	Availability KPIs	Lack of KPIs	Big Data method
🟦 False Negative Rate	84%	16%	5.90%
🟥 False Positive Rate	0%	0.90%	0%

Figure 4.7: Comparison between the proposed Big Data method and other common techniques

Diagnosis based on KPI correlation

In diagnosis a very important source of information is to find which PIs correlate the most with the occurrence of the problem. Since the KPIs are an indication of the general behaviour of the eNodeB, a list of the most correlated indicators will give an important clue to the root cause analysis. In [57], a method that performs this analysis is described. This algorithm takes into account the PIs of the affected eNodeB and the neighboring sectors in order to simplify the task of diagnosis. The process of calculating the correlation of two time series is a computationally heavy operation, and the number of correlations that must be done is high (all the PIs of the analyzed sector, plus all the PIs of each neighboring sector), qualifying it as a Big Data problem; but since each PI can be processed independently, the algorithm is easily parallelizable. The correlation process is implemented as a *map* function, and a *reduce* function creates a list of PIs ordered by correlation. In Figure 4.8, a time series of a KPI in the diagnosed eNodeB (the *Number of Radio Resource Control Connections*) is shown, along with a highly correlated PI of a neighboring eNodeB (a counter of *Bad Coverage* reports) and their correlation.

4.2 Troubleshooting Data Collection

In Section 4.1 the data from the network was described. In this Section, the process for collecting that data and creating a fault database for the KA process is described. The automated KA process has two inputs; the data and the experts analysis of the data. The expert's input must contain enough information so that the appropriate time interval and spatial aggregation can be used to collect the data.

The automatic KA process has two main parts regarding the expert's point of view (Figure 4.9):

- Knowledge Capture: collection of the information from the expert and use that information to filter by time and space the network data. On the KDD scheme (Section 3.3), Knowledge Capture corresponds to the Selection step.

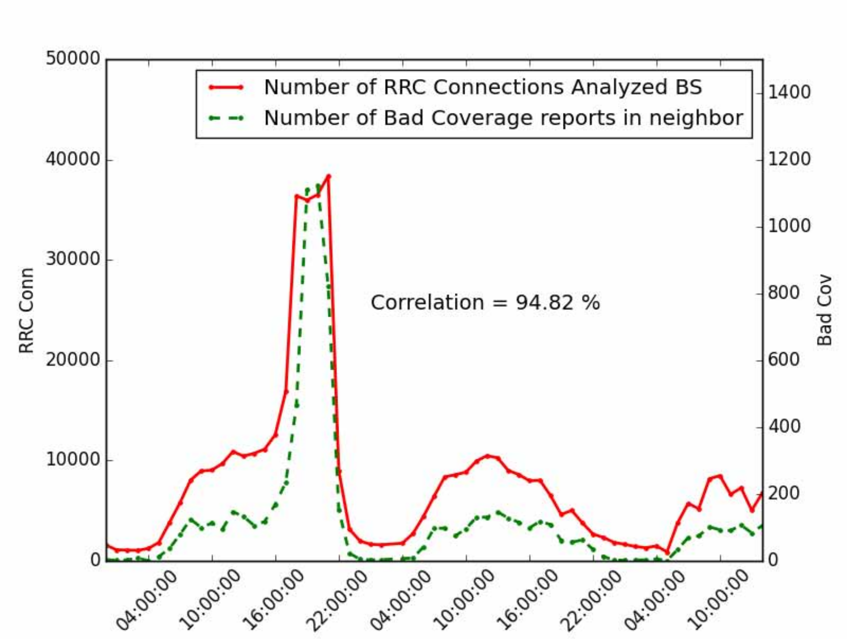


Figure 4.8: Correlation between the KPI and a PI of a neighboring sector

- Knowledge Modelling: creation of a model of the knowledge that will be used to create a KB. Knowledge Modelling encompasses the rest of the KDD steps, that do not require human intervention.

This Section explores the Knowledge Capture step. Specifically, a troubleshooting data collection application will be described. This application will let experts indicate when a diagnosis has been done, and the time frame that they observed for the affected sector in order to find out the root cause. Knowledge Modelling is further studied in Chapters 5 and 6.

The output of the application will be a small dataset containing labeled tables of CM, PM and FM parameters (and optionally more variables). Each table will contain the data of an affected sector (and optionally its neighbors) over the period of time where the problem was observed. This database can then be used for multiple purposes, such as modelling problems, generating a KB or even training new experts.

4.2.1 Requirements of a Troubleshooting Data Collection System

The troubleshooting data collection application has three interfaces: a Graphical User Interface (GUI) for the experts to insert the diagnosis and two database connections: an input interface that will collect the data from the Operations and Support System (OSS) centralized database and an output connection that will save the output of the algorithm in a database to further process the data. On the user side, some requirements must be taken into account:

- A major issue with specialists is time. Therefore, the UI must be easy to use, and straightforward, ideally taking no time out of the experts workflow for learning or operating it.
- The UI must only take information once the work rush of troubleshooting is finished, so the process of reporting a solved problem is not intrusive.

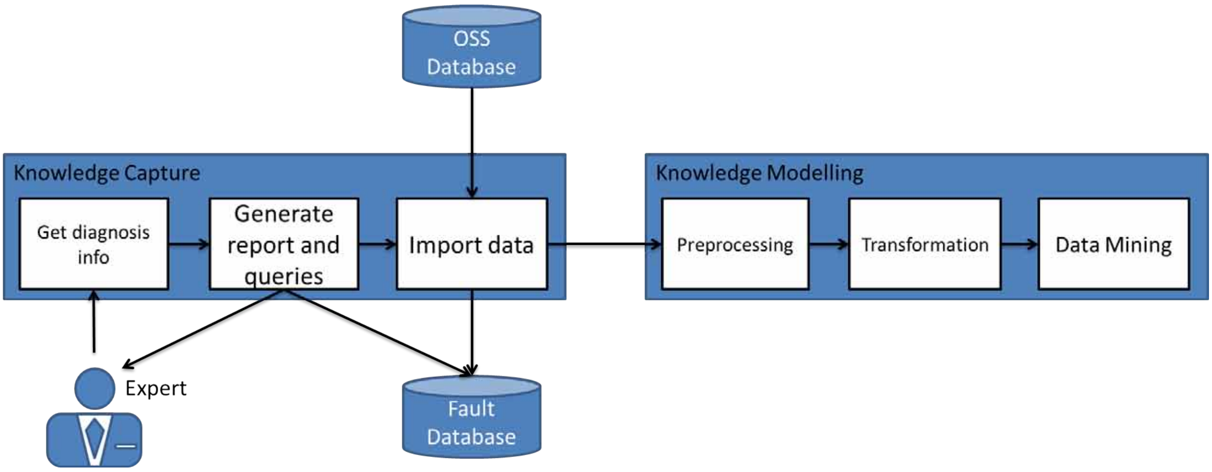


Figure 4.9: Knowledge Acquisition parts from the point of view of the expert

- It would also be important that the UI offers an incentive to its users. Since the strong point of the system is the collection and processing of information, it could offer strategical information to its users about a problem under study.

These requirements were provided by troubleshooting experts when queried about their experience with current software solutions for troubleshooting.

The input database connection is vendor-specific, and it can be implemented usually using the appropriate Application Programming Interface (API) that will take care of all the requirements. Regarding the input data, the requirement for the purpose of creating a model of the problems is that there must not be too many missing or incorrect data. In subsequents steps of KDD, minor errors can be corrected, but large errors (i.e. long periods of time with missing data) may cause malfunctions.

On the output of the application, the data must be varied, that is, for each problem there must be as many different examples as possible, so that the DM process can find the common patterns and discard those that are not related.

4.2.2 Troubleshooting Data Collection Tool

Considering the requirements described in Section 4.2.1, an application has been developed in order to collect troubleshooting data. The application takes two inputs (the experts information inserted through the GUI and the network data retrieved through a database connection) and produces one output (the data of one troubleshooting case). The outputs are accumulated in a database of solved problems that is populated as new cases come in. Figure 4.10 shows the general diagram of the tool. The application has four main components:

- Main module: The part of the application that coordinates the operation and interaction of the other modules.
- GUI: The interface between the user and the system. It must follow the guidelines described in Section 4.2.1. The interaction of the experts with the application is shown in Figure 4.11. Once launched, the application offers a form that the expert must fill. To simplify

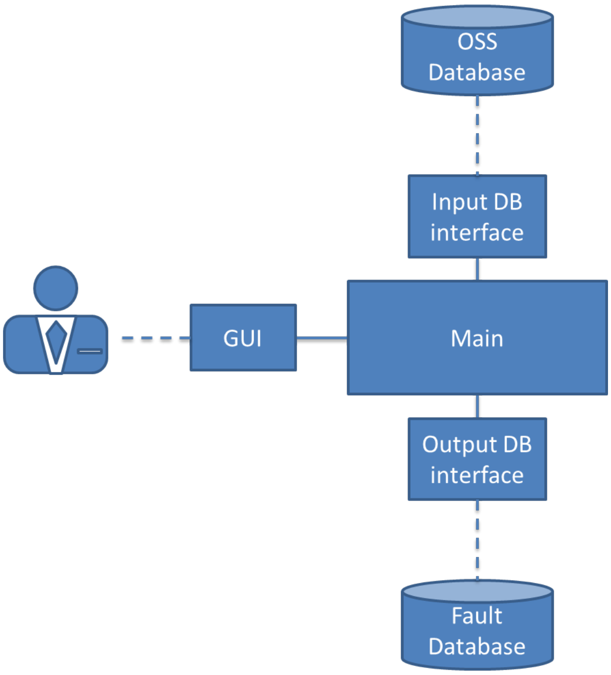


Figure 4.10: Components of the data collection application

and minimize the intervention of the experts, the interface only asks for the date of start and end (the exact hour is optional since degradations are isolated in the transformation step, as shown in Chapter 5), the name of the affected eNodeB or sector and a single tag for the diagnosis. Optionally, the experts can include extra comments, indicate if the problem is solved or ongoing and the solution to the problem. Experts submit the form and the collection of data is performed in the background. Finally, the application offers a summary. In this point, the application can provide with a useful report on the inserted data, such as detection of degradations (this process is described in Chapter 5), or finding similar cases in the database. Figure 4.12 shows a screenshot of the diagnosis submission form.

- Input database interface: interface between the OSS database and the system. This component receives the information collected from the GUI and performs a query on the OSS side to retrieve the data. The date and site information are used as filters in the query.
- Output database interface: interface between the system and the solved problems database. The application joins the data retrieved from the network with the information provided by the expert (mainly the root cause, but also the additional optional information if it was included). The output is saved in a local database or filesystem.

The application has been developed on the Django [84] platform based on Python [85], that offers a web interface for the GUI and lets the user launch processes on the server (using the Celery [86] library). The Django platform uses a Model-View-Controller (MVC) [87] architecture. In this application, the input and output interfaces are defined by the model, the GUI by the view and the main module corresponds to the controller. The input interface is implemented using the SQL database interface provided by the SQLAlchemy [88] library, since the OSS database

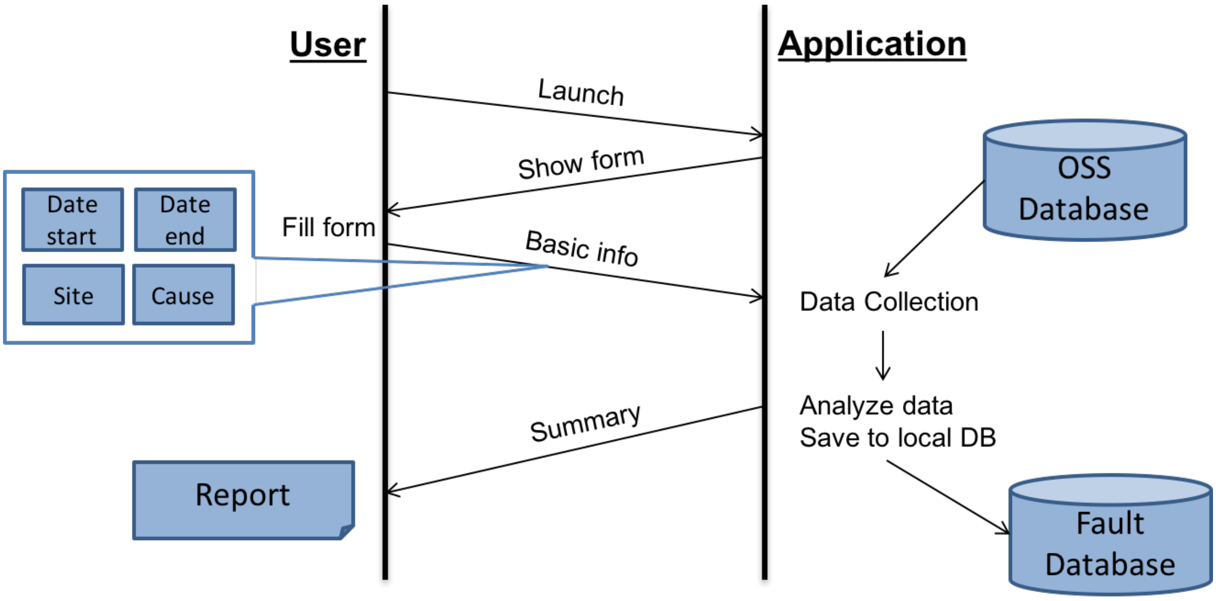


Figure 4.11: Interaction between the user and the application

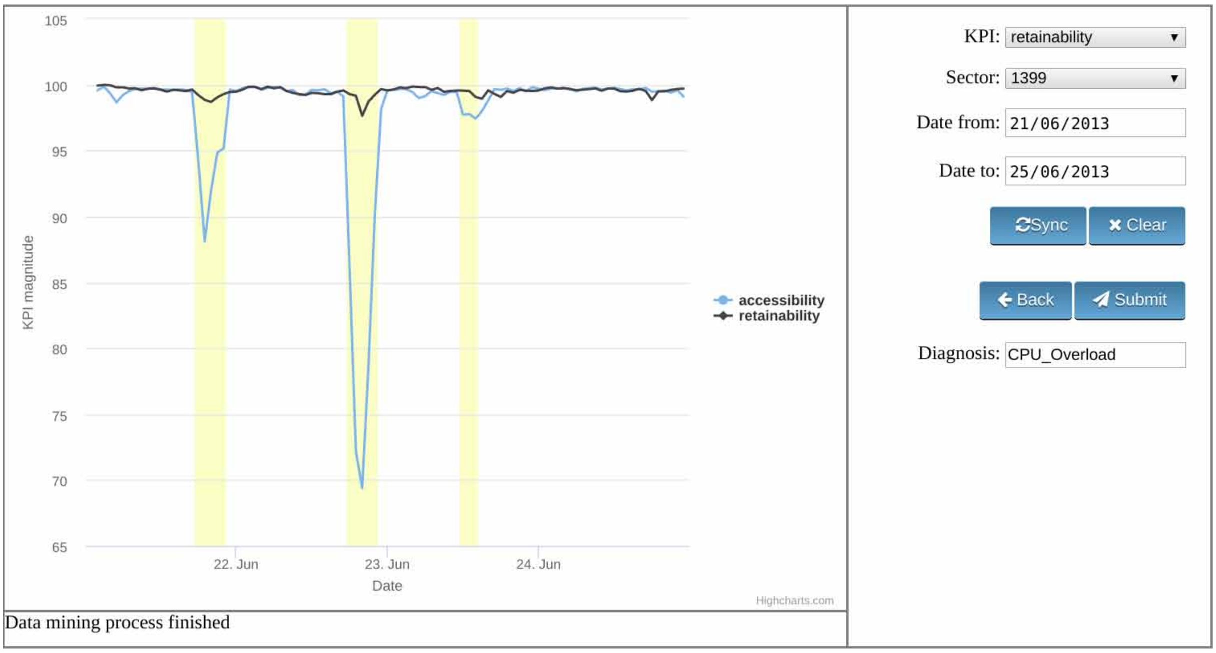
uses this protocol to access the data. The output database interface uses both a separate native Django interface to save data on a local database and the Pandas [89] library in order to perform some analysis on the data and save to CSV files. The rest of the KA processes on the data are launched with a Celery worker.

These design decisions are based on the ease of use and convenience of a web interface, that is in sync with the latest trends in software product development (Software as a Service [90]). Also, the high capabilities of Python for data processing and the availability of libraries for both the web and the database interfaces, drove to its choice for the implementation.

The application described in this Chapter was deployed over a live network. As reports of new problems were received, they were fed into the application, slowly collecting a database of solved problems. In the next Chapter, the problems collected with this application are used to generate a model of the problems in the LTE network. This model will be used to further study and characterize the behaviour of the KA process; but in a full deployment of the automated KA and troubleshooting system (using the Lambda architecture), the modelling step would be optional (although very useful if the number of collected cases is low). In this scenario, once the application has collected a reasonable number of problematic cases, the rest of the KDD functions (preprocessing, transformation and DM) are launched.

4.3 Conclusions

In this Chapter, the KA process for diagnosis systems was described. Firstly, the data available both for diagnosis and for KA was described, specifying the data sources and formats. The dimensionality of these data was also described, showing how it can be qualified as a Big Data problem. The Big Data concept applied to Self-Healing was further explored, describing several processing techniques and scenarios where they can be applied.



KPI: retainability
Sector: 1399
Date from: 21/06/2013
Date to: 25/06/2013
Diagnosis: CPU_Overload

Data mining process finished

Figure 4.12: Screenshot of the form where the expert inserts a diagnosis

To perform KA using KDD techniques, large datasets must be collected. The requirements of an application to collect these data were described, including UI requirements. Finally, an application developed for collecting troubleshooting data from real networks is described. This application only requires a very reduced quantity of information from the experts, namely, the date where the problem occurred, a label describing the problem and the affected eNodeB. In the next Chapter, the data collected from a real network with this application will be fully described.

Chapter 5

LTE Fault Database Modeling

In this Chapter, the database of collected problems will be studied in detail. A modelling process for the database will be proposed and applied to summarize the relations between the indicators and the problems. The full process that is applied on the original raw database is shown in Figure 5.1. Finally, the model will be used to emulate vectors of solved cases for their use in data mining.

5.1 Collected Fault Database

In Chapter 4, a process for collecting a raw database of solved problems was described. An application using that process was developed and deployed. In this Section, the network over which the application was deployed is described, along with the description of the collected database.

At the moment when the database collection started, the network was recently deployed (less than 2 years) by a major United States operator and optimization was still ongoing by Ericsson. The network was growing, so new sectors were added frequently, with optimization problems in the beginning. The network covers a major urban area in the United States and it is part of a larger network owned by the operator. The network has 19879 sectors (5366 eNodeBs), divided among 4 large clusters, each subdivided into smaller groups that cover specific neighborhoods of the urban area. To use the resources more efficiently, two carriers were deployed.

5.1.1 Building of the Fault Database

The troubleshooting of a network of this size is a complex task that requires work division among several troubleshooting experts. Each cluster is assigned to a single expert that oversees its behaviour. If degradation is detected on the high level KPIs of the cluster (such as *Accessibility*, *Retainability*, *Handover Success Rate*, etc.), a top down process is started where a list of worst offenders is constructed for that specific KPI to find which sectors are degrading it. Further

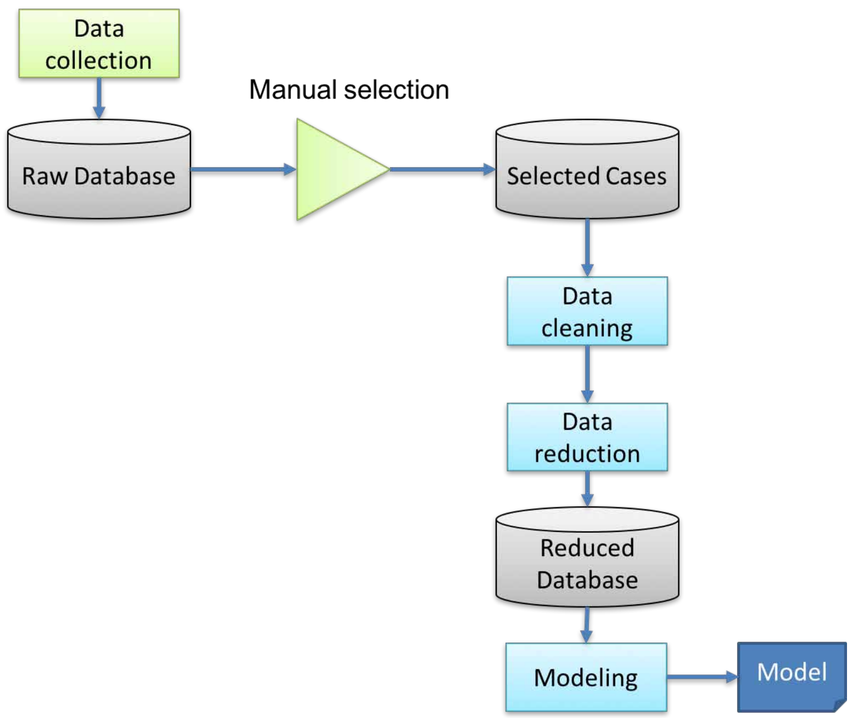


Figure 5.1: Process applied over the original dataset. Actions marked in green show the manually done actions and those in blue show the automated steps.

analysis of the lower level KPIs of the worst offenders is performed in order to diagnose the problem if possible. Each sector may also raise alarms if a failure is observed, and depending on its priority, it may need to be addressed by the troubleshooting engineer without being a worst offender. Additional measurements may need to be taken on-site by a team of engineers in order to have more information for the diagnosis. Some diagnosis are partial, requiring additional information from other sources (such as hardware or software problems that can only be fully diagnosed by the vendors). If there are no visible degradations on the cluster level, experts usually still inspect the worst offender lists in case that a specific sector is having problems although its effect on the overall behaviour of the cluster is not important. Also, incident reports may be raised from other sources (such as user complaints) directly pointing to a failing sector.

In all this process, there is a high volume of information being transmitted among troubleshooting experts. The information of each case and an iteration of suggestions and tests is shared between the responsible engineers of each sector and the troubleshooting teams (that include experts in several fields). Once the problem is diagnosed, the corrective actions are taken. These actions may or may not solve the problem; in a negative case, new solutions are proposed, whereas if the problem is solved, the affected sector is monitored for a time to assure that the solution has worked.

In this scenario, the application described in Section 4.2.2 was used to summarize the information contained in the email exchange of each case. For each one, some important aspects were saved:

- Start and end date: approximate start of the problem, or, if not explicitly stated, the date

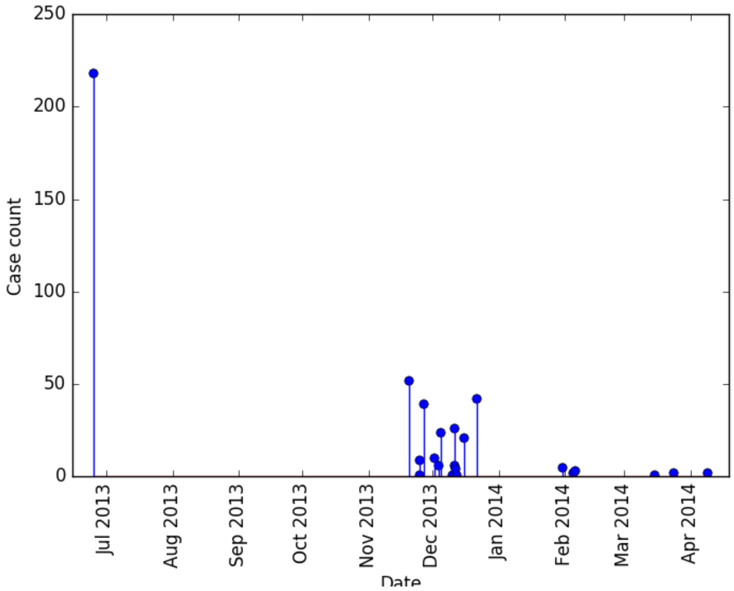


Figure 5.2: Timeline of the collected cases

of submission of the first email identifying the problem. The date of the end is taken as the date where a solution was applied, or the last email refering to the problem was submitted, if the problem remained unresolved. These dates are taken for the records, but actual data is gathered starting two weeks prior to the problem and finishing on the end date; in order to perform a search for trends prior to the visible degradation.

- Sector name: name of the affected sector or sectors.
- Summary of the problem and a label identifying the problem. The label must be the same for similar problems.
- Summary of the solution (if any).

As explained in Section 4.2.2, once the basic information is collected, the application automatically downloads the data of the sector (CM, PM, FM and KPIs) and saves it to a local database.

5.1.2 Collected Database Summary

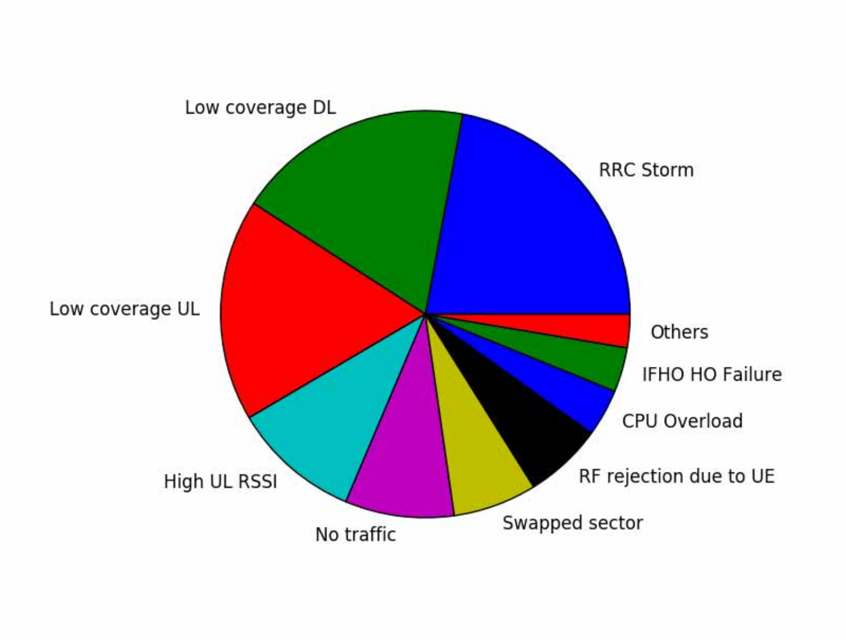
Following the interchange of emails of troubleshooting experts for a period of 10 months (from June 2013 to April 2014) has produced a list of 475 cases.

In the database, each *case* represents a problem and it is composed of a set of collected PIs and a label identifying the problem. The data collection was performed in three different stages (Versions 1 through 3), where the list of collected PIs was successively improved. Version 1 (100 cases) of the PI list included only a closed set of KPIs. Version 2 (360 cases) extended that list to all available PIs, and Version 3 (15 cases) further increased the list with PM counters, FM alarms and CM parameters. Figure 5.2 shows a timeline of the collected cases.

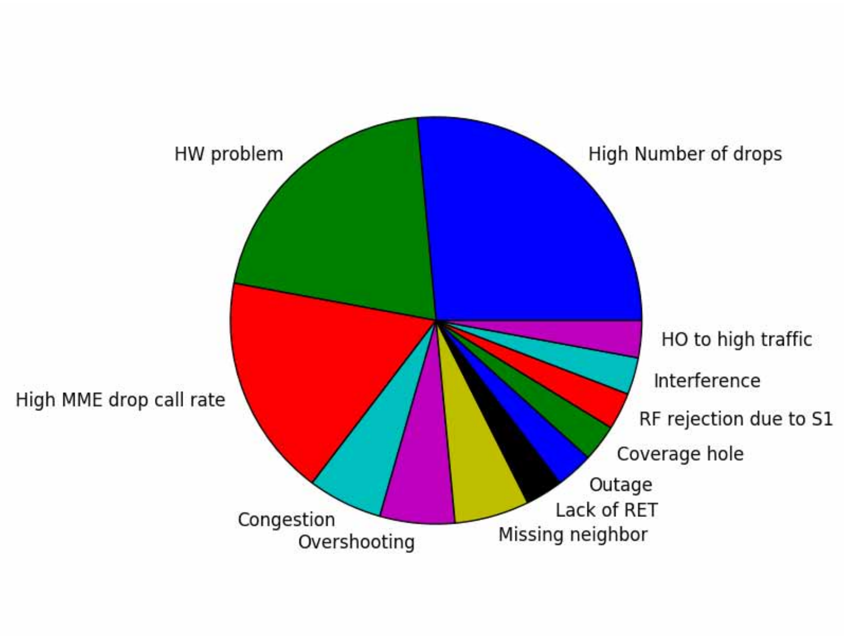
The timeline shows a large amount of cases obtained around June 2013. These cases were collected using the results of a manually-coded rule based diagnosis algorithm. This algorithm was executed as a set of SQL queries generated from a user-friendly MS Excel interface; therefore it is an example of traditional KA, where the experts are asked to manually insert the knowledge in the system. Nevertheless, the whole system was a prototype that could not be easily extended with new PIs and rules. Any minor change required an intensive task of SQL and MS Excel programming. Since the algorithm was executed only once, the results were collected at the instant where the analysis took place, resulting in a large concentration of cases on the same day. The results of this collection campaign had a very high number of false positives. Furthermore, the algorithm only returned the IDs of the affected cells in the last ROP; therefore, this information was fed to the data collection platform described in the previous Chapter. The convention used to automate the data collection was to mark the end of the problem as the date of collection and the start as two weeks prior. The label of the problem was given by the SQL query (each representing one rule, and therefore, one problem) that had retrieved it. No additional details were available on this cases, so the summary fed to the collection system was empty. A large number of cells appeared into more than one category (due to the high False Positive Rate of the algorithm). The rest of the cases were collected feeding the basic information into the data collection platform as new reports were received during the data collection stage between November 2013 and April 2014. These cases were mostly reported through an internal mailing list.

Each of these cases is diagnosed with a summary that may be more or less complex, but in this thesis they have been classified in a set of labels that group cases that have a very similar behaviour according to the troubleshooting reports. These labels are more or less informative; in some cases they describe the root cause, and in others they just indicate that a specific behaviour is common along all the cases, without there being a clear indication that the underlying root cause is the same for all cases. These labels are not mutually exclusive. Specifically, 21 types of problems have been collected. The full list and description is given in Appendix B.

Figure 5.3 shows the proportions of the problems. The number of *RRC Storm* cases is the highest, because this situation was reported automatically by a specially designed algorithm. This algorithm would monitor certain PIs and restart automatically the eNodeB if there was any possibility that the problem was happening. Since it was deployed on a network in service, and the cost of restarting an eNodeB was relatively low, no care was taken for minimizing the false positives of the algorithm. Therefore, this class was unusable for the purpose of modelling because of the high false positive rate and also because the number of collected PIs (Version 1) was insufficient. *Low Coverage* (in the downlink and in the uplink) are the second and third most commonly collected problems since the number of border cells is high. These two classes and the rest of the most common problems (shown in Figure 5.3a) are collected using the results of the previously mentioned manually coded rule-based diagnosis algorithm, that also had a relatively high False Positive Rate. These cases had to be manually analyzed in order to make sure that the diagnosis was correct, and many cases were discarded because it was not clear that there actually was a problem or that the diagnosed problem was correct. This step is required in the case shown in this thesis because cases were collected in an environment which was actively being developed. In a real deployment, this step should be performed at the time of inserting the cases in the system. The classes with the lower proportions (Figure 5.3b) were collected based on direct reports by troubleshooting experts.



(a) Most common problems



(b) Other problems

Figure 5.3: Proportions of collected problems

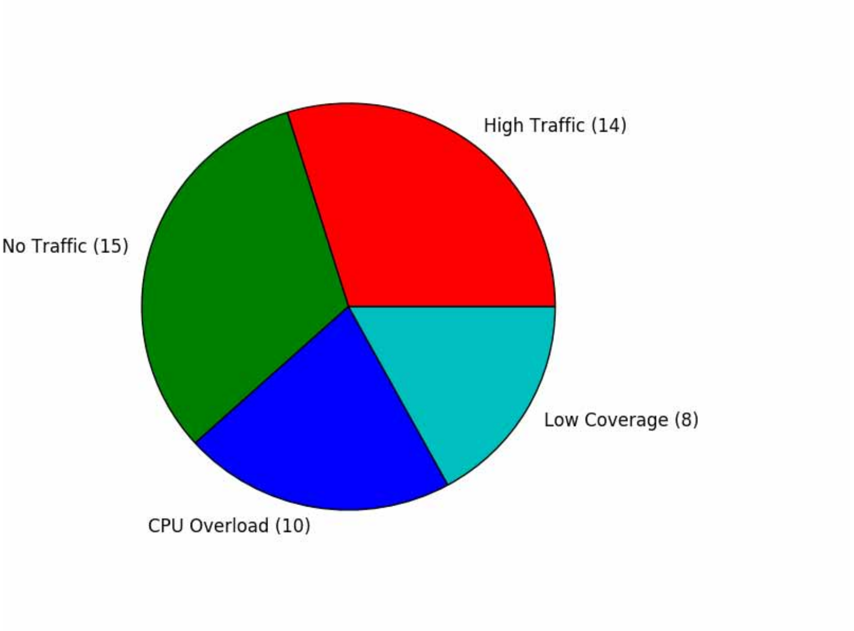


Figure 5.4: Proportions of root causes among selected cases

From these problems, only a reduced set of them is taken for the development of the system. Cases that have a very low number of incidences (such as *Outage*) are discarded because of the low variety of situations. This would cause overfitting in the DM process. For instance, in a case of *Interference*, the *HOSR* may or may not be degraded; but that is not correlated with the main cause of the problem (i.e. the degradation would be caused by another root cause). If the only examples of *Interference* have a degraded *HOSR* and their number is low, the DM algorithm would find a 100% correlation between *HOSR* degradation and *Interference*. Another reason for discarding problem types (such as *RRC Storm*) is that the diagnosis would require the analysis of non collected data (such as PIs from neighbors or field test results).

With these considerations, a reduced list of problems has been selected for the extraction of diagnosis rules. These problems have been analyzed in order to extract the root cause with the help of the experts, and have been classified under four labels: *High Traffic*, *No Traffic*, *High CPU Utilization*, *Low Coverage*. Note that these labels indicate the root cause, whereas the previous labels classified the cases according to their symptoms. Figure 5.4 shows the number of cases for each root cause.

For these cases, a variable number of PIs has been collected, depending on the Version of the data collection mechanism. The list of PIs that have been used in this thesis, along with their descriptions is given in Appendix A.

5.2 Processing LTE performance data

To perform the DM process, a *learning set* composed by labeled vectors of PIs is required. But when the data is collected from the network, their format is totally different. Therefore, a data preprocessing must be done to create these vectors without loss of information. Two steps are required in order to do this transformation: *data cleaning*, where missing values are replaced with a calculated value in order to improve the quality of the results, and *data reduction*, where the originally downloaded data is converted into an appropriate format with a lower dimensionality. This process corresponds with the Preprocessing and Transformation stages of KDD (Section 3.3). The input of the process is composed of the data collected from the network, given as a matrix of time series per PI (or PM/CM/FM values), and the output is a set of vectors with one element per PI.

5.2.1 Data Cleaning

When data is collected from the network, some “dirt” may be collected. Depending on the nature of the collected data, the “dirt” may take different forms:

- Outliers: When measuring a magnitude over a large set of sources an abnormally high or low value will have a high impact on the overall behaviour of the measurement. For instance, when measuring the average *Downlink RSSI* over all the UEs of a sector, one single very high measurement (e.g. due to a user that is located near an external interference source) may drive up the average and standard deviation, giving an inaccurate picture of an otherwise well-behaving sector.
- Missing data: When monitoring a magnitude over time in fixed intervals, an event may happen that prevents the measurement over one period. In the case of CM/PM/FM monitoring, many factors may cause this situation: an outage in the eNodeB where measurements are taken, a failure in the transmission over the monitoring network, a fault in the operator’s database host, human errors, etc ...
- Wrong data: Some measurements may be outright wrong, due to a bad configuration of the measurement mechanism or a fault when transmitting or reformatting data at some point. Unless the error causes illegal values (such as negative values for a counter or proportions higher than 1), these errors are really difficult to detect.

In this thesis, it is considered that there are no outliers or wrong data (i.e. the counters are considered reliable). In fact, in the software used for data gathering in the OSS central database, collected variables are checked and left empty if their values are illegal (e.g. percentages larger than 100%). This leaves the possibility of faults in the data gathering system which cause missing data; which is a fairly common issue. Since the data reduction technique described in Section 5.2.2 needs time series without gaps, an algorithm must be used to deal with these missing data. Traditionally, several simple techniques are used:

- Default value: when a missing value is encountered, it is filled with a default value; for instance, 0 for a counter, or 0 % for a proportion. The issue with this technique is that it creates “fake” values that may affect the preprocessing results. But since these values are still valid, they are difficult to discard.
- Variable removal: the variable is completely discarded. Normally this is used in combination with other techniques for a certain threshold in the proportion of missing values. If the number of missing values is above the threshold, the variable is discarded. Otherwise, missing values are filled. If the discarded variable is required in some process further down the preprocessing line, this may cause a failure. Therefore, if this technique is used, a

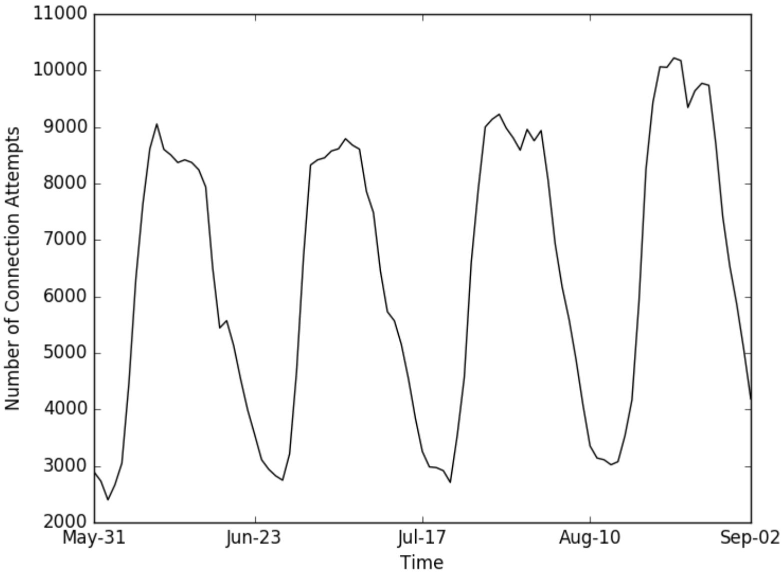


Figure 5.5: The *Number of ERAB Connections*, like many other PIs, has a daily periodicity.

recovery strategy must be defined, in the dependent processes.

- Reconstruction: a value is artificially created in order to fill the gap. It may be a simple interpolation of the non-missing values that surround the gap. Although this technique may create fake values, in small proportions it may increase the quality of a time series so it can be processed.

The technique used in this study is a reconstruction algorithm that uses statistical properties of the mobile network data. Specifically, it relies on the 24 hour periodicity that is usually observed in the values of PIs. Figure 5.5 shows an example of a PI over consecutive days. It can be observed how the general behaviour repeats on a daily basis.

Missing data is filled using the hourly average, that is, the average of the PI for the sector under study and the hour of the day when the missing value is located. This value is adjusted to the behaviour of the PI for the last 48 hours. The formula for a missing value x_m is

$$x_m = \frac{\overline{x_{48}}}{\overline{x}} \overline{x_H}(h) \tag{5.1}$$

where $\overline{x_{48}}$ is the average of the PI for the last 48 hours prior to the missing value, $\overline{x}$ is the historic average for the affected sector and $\overline{x_H}(h)$ is the historic average at hour h (the hour of the day when the missing value occurs).

This method has some limitations; it may introduce inaccuracies in the data if used for many consecutive missing values, since reconstructed values are always an estimation. Also if the behaviour of the sector changes in the missing time interval, or in the last 48 hours, the resulting estimation may be skewed towards the old behaviour. Outliers in the values used for calculating the three averages (especially in the last 48 hours) may increase the inaccuracy. Finally, in order to correctly fill the data, the algorithm needs at least 48 hours of prior data.

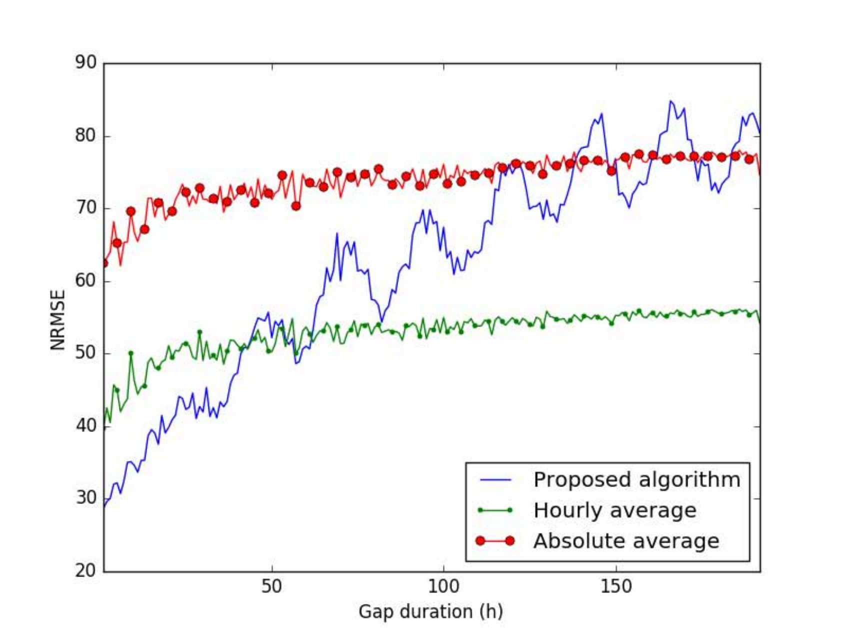


Figure 5.6: NRMSE of the data filling method compared with filling with a constant (the absolute average) and the hourly average.

The algorithm has been tested over real cases, simulating missing data by hiding known values and estimating them. To test the accuracy, the Normalized Root Mean Square Error (NRMSE) of the estimation is calculated for each case, given by:

$$E_e = \frac{1}{R}\sqrt{\frac{\sum\limits_{i=n_1}^{n_T}(V_r(i)-V_e(i))^2}{n_T}} \tag{5.2}$$

where R is the range of the PI, $V_e(i)$ is the estimated value at instant i, $V_r(i)$ is the real value at instant i, n_1 is the time of the first missing data and n_T the time of the last one. Figure 5.6 shows the averaged NRMSE of the results applied over 100 different time series of the same KPI (*Number of Connection Attempts*) for increasingly long periods of missing data. The NRMSE of the proposed data filling method is compared with using the absolute average of the time series and the hourly average. It can be seen that the proposed method improves the NRMSE, although for increasing amount of times, it grows. Nevertheless, this method should not be used for very long periods of time, since it relies on the data being predictable; therefore, it will not be able to fill correctly values that are abnormal.

The output of this algorithm is a matrix of time series that do not have missing values, enabling the application of algorithms that do a sequential processing over the data, such as the data reduction algorithm presented in Section 5.2.2.

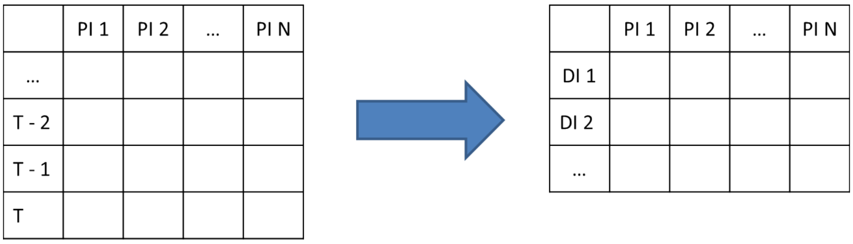


Figure 5.7: Input and output formats of the Data Reduction algorithm

5.2.2 Data Reduction

As shown in Section 4.1.1 (specifically, in Figure 4.3), the data collected from each eNodeB is given as a time dependent matrix, where each column is a time series representing a PI or CM/PM/FM value. Each troubleshooting solved case has one of these matrices and a label indicating the problem. But this format is complex for being processed by automatic algorithms and usually has superfluous information. The Data Mining algorithms described in this work (Chapter 6), require vectors of information where each element represents the value of a PI, labeled with the problem (without the temporal dependency). Also, a major issue with the original data format is its volume. Normally the number of indicators is high, and there is one entry per indicator per time unit. The high volume of data (and the proportion of superfluous information) will slow down the Data Mining process. Therefore, a data reduction process that reduces both the dimensionality and the volume of data is required. Figure 5.7 shows the input and output formats of the process.

Traditionally, there are several techniques for data reduction, and their use depends on the features of the data and the requirements of the Data Mining algorithm:

- Aggregation: data is aggregated (summed or averaged) over one dimension.
- Variable selection: detect and remove irrelevant, weakly relevant or redundant variables. For instance, PM counters that are used for calculating a PI but have no meaning on their own.
- Dimensionality reduction: encode the data set to reduce the size (e.g. Wavelets, DFT, Principal Component Analysis) and remove one dimension.
- Numerosity reduction: replace data by creating models. Models may be either parametric (a set of parameters for a known function) or non parametric (such as a set of samples or a histogram).

In this study, several of these techniques are used. Variable selection is performed based on a manually compiled list; although this can be improved by automating the creation of the list using criteria of correlation with the occurrence of degradations. In order to find such relations, the problem database must be studied. The list of manually selected PIs, along with their descriptions is given in Appendix A.

A dimensionality reduction algorithm is proposed and used in combination with an aggregation method. The dimensionality reduction algorithm will detect the intervals of time on a downloaded data matrix where the behaviour of the eNodeB is degraded (hereby named as *Degraded Intervals* or DIs). Once determined, the values of the PIs are aggregated using their

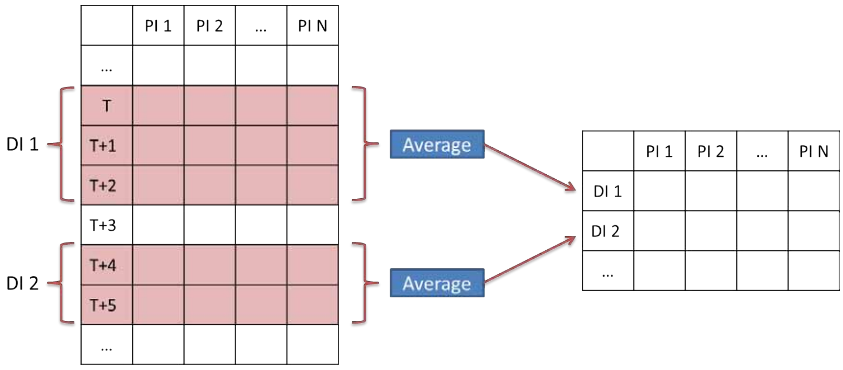


Figure 5.8: Transformation of time dependent matrix into a vector of averages

average in that period of time. Figure 5.8 shows this process.

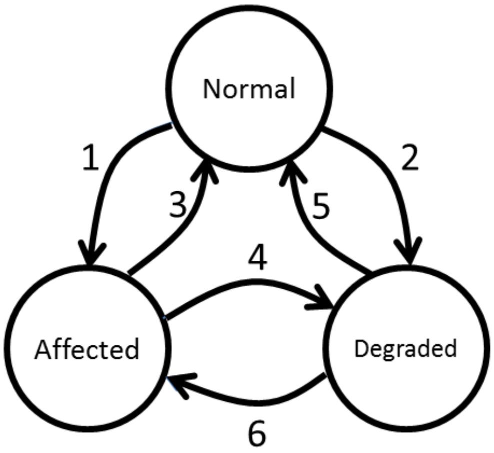
To detect the DIs, one or more driving PIs are used. A driving PI is an indicator that highly correlates with the general behaviour of the sector. Whenever a degradation occurs in the eNodeB, at least one of the PIs in the set of the chosen driving PIs must be degraded. For each driving PI, two thresholds are defined: a *good* threshold, above (or below) which the PI is considered as having a *normal behaviour*, and a *bad* threshold, below (or above) which the PI is showing a *degraded behaviour*. Values between these thresholds are considered to have an *affected behaviour* that would normally indicate that there might be a problem in the sector that is slighlty affecting the PI. These thresholds are equivalent to the parameters of the membership functions described in Section 3.1.4.

To define if the overall behaviour of a sector is degraded, a state machine (Figure 5.9) will be used. Three states are defined:

- *Normal*: All the KPIs are in the *normal behaviour* state.
- *Affected*: At least one KPI is in the *affected behaviour* state, but there is none in the *degraded behaviour* state.
- *Degraded*: At least one KPI is in the *degraded behaviour* state.

The algorithm will consider a DI any sequence of states that starts with a *degraded* overall state and contains only *degraded* or *affected* states. The DI will then start with the first *degraded* state of the sequence and finish when the last *degraded* state ends. Figure 5.10 shows a full example of a detection using two KPIs: *Accessibility* (proportion of successful connection attempts) and *Retainability* (proportion of calls ended without a drop). Both of this KPIs are percentages, and are considered degraded if they are below 98% and affected if they are between 98% and 99%. At 10:00 Retainability falls below 98% (*degraded*), initiating the DI. At 11:00 it climbs above 98%, but it doesn't reach 99% (*affected*), so the DI is not finished. At 13:00 Accessibility also becomes *degraded*, forcing the total state to be *degraded* too. At 16:00 Retainability becomes *normal* (above 99%) and Accessibility *affected*. At 18:00 Accessibility returns to *normal* values. This also causes the overall state to be *normal*, retroactively marking 16:00 (the end of the last *degraded* state) as the end of the DI.

Once the DIs have been determined, the values of the PIs are averaged in the time interval to obtain a vector.



Normal: all KPIs are normal.
Affected: at least one KPI affected. None degraded.
Degraded: at least one KPI degraded.

1: at least one KPI changes from normal to affected.
None degraded.
2: at least one KPI changes from normal to degraded.
3: all affected KPIs change to normal.
4: at least one KPI changes to degraded.
5: all non-normal KPIs change to normal.
6: all degraded KPIs change to affected or normal.
At least one KPI remains affected.

Figure 5.9: State machine of the detection algorithm

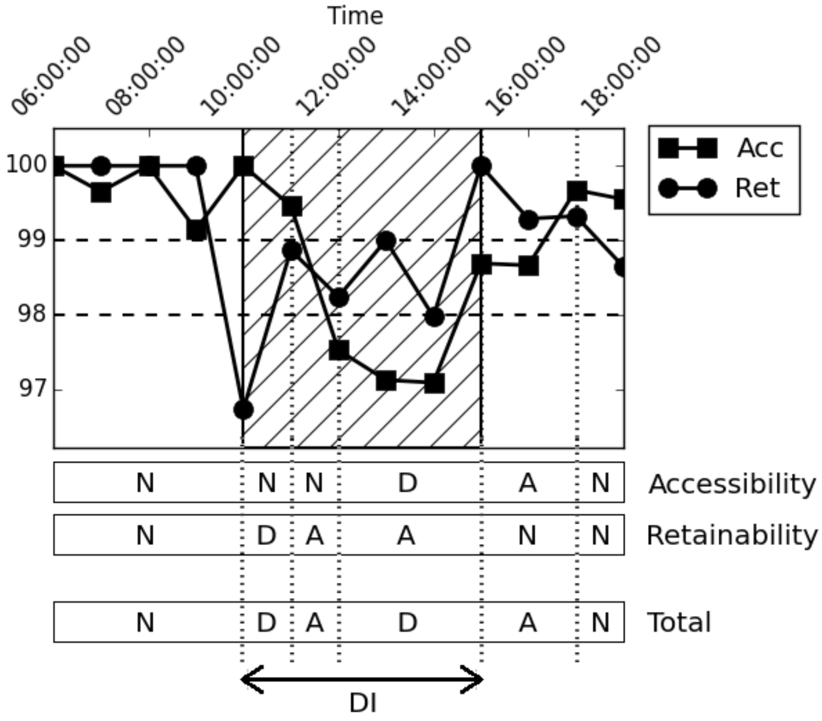


Figure 5.10: Example of a DI detection using two KPIs

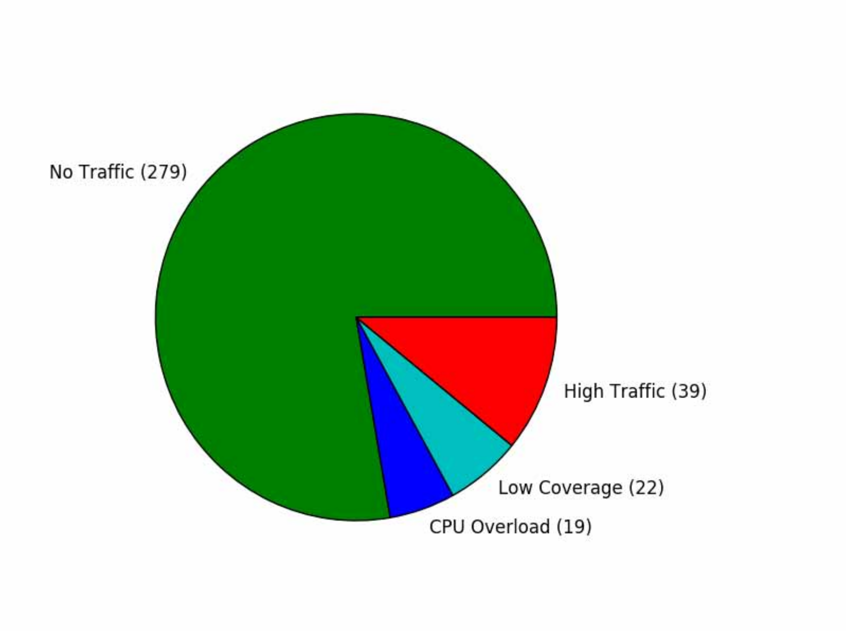


Figure 5.11: Proportions of problems in the reduced database

5.2.3 Processed Fault Database

In Section 5.1.2, the original database was described. Once the data preprocessing has been applied, a new reduced database is obtained. In this case, each entry (DI) will be accounted for as a separate occurrence of a problem, although each case in the original database may produce several entries.

The reduced database has 359 entries, distributed according to the proportions shown in Figure 5.11.

The number of detected DIs greatly modifies the proportions (as compared to Figure 5.4). The *No Traffic* problem causes 279 degradations out of 15 cases. The *Low Coverage* problem, that originally had the least cases (8) now has 22 DIs, 3 more than *CPU Overload*.

The total size of the database in elements is also reduced. The size in elements in the original database (S_O) is given by:

$$S_O = \sum_{i=0}^{N_C} N_K(i) \cdot N_H(i) \tag{5.3}$$

where N_C is the number of cases, $N_K(i)$ is the number of PIs and CM/PM/FM variables collected for the case i, and $N_H(i)$ the number of hours collected for case i. With this formula, the original database has 10929065 elements. The size of the reduced database is given by:

$$S_R = N_K \cdot N_D \tag{5.4}$$

where N_D is the number of detected DIs and N_K is the size of the set of PIs collected for all the cases (585). Not all of these PIs will be used in the model, but all of them will be considered for the reduced database. With this formula, the number of entries in the reduced database is

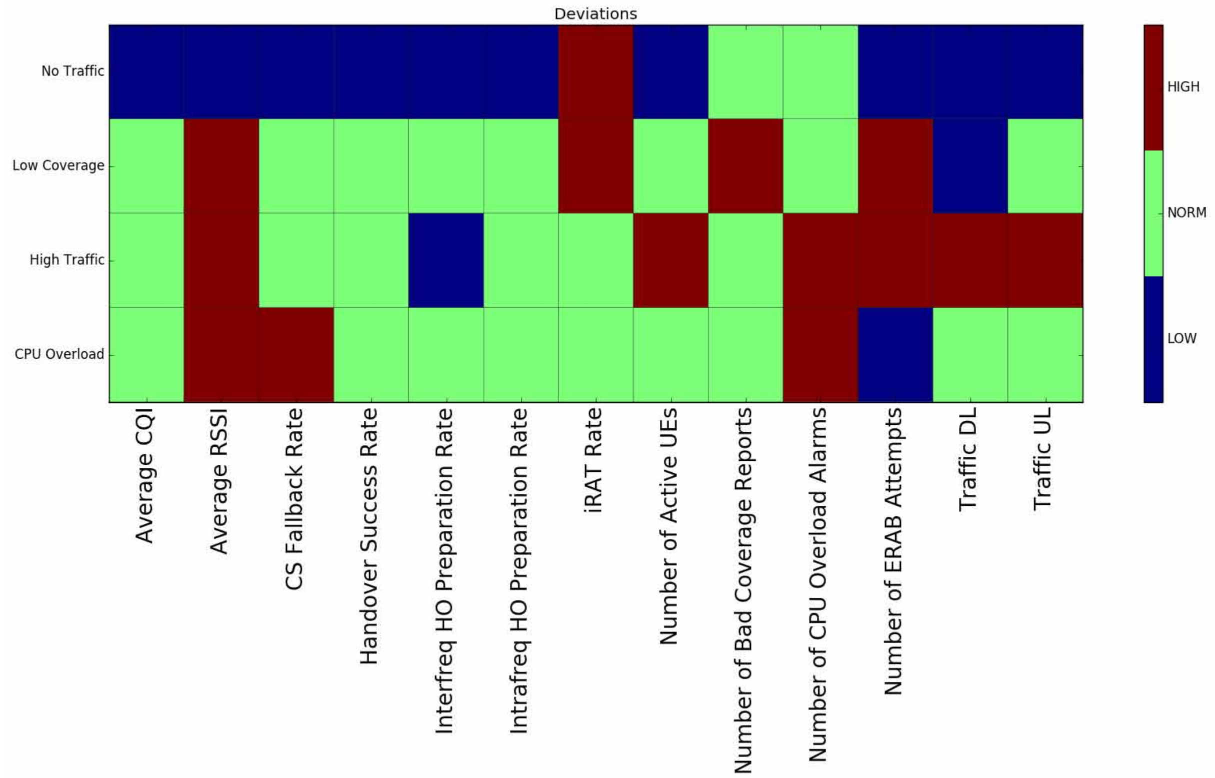


Figure 5.12: PIs with conditioned averages more than one standard deviation above or below the normal average.

210015, which represents a reduction of 98.08% on the original volume.

The data in the reduced database is much simpler; there is one value per PI for each occurrence. This makes possible the measurement of the behaviour of the PIs conditioned to the occurrence of each root cause. An in-depth study and model extraction based on this premise is shown in Section 5.3. A more general overview is shown in Figure 5.12; this figure shows which PIs have an average that is more than one standard deviation above (red) or below (blue) the normal average, conditioned to the occurrence of each problem. This gives an idea of whether a PI will have a *"high"* or *"low"* value for a specific problem.

5.3 Modeling Fault Databases

After data reduction, each solved case is composed of one or more labelled vectors. Each vector can be considered an independent entry in a training set for a data mining algorithm. In order to better understand the relations among the PIs and the root causes, a modelling process has been developed. This modelling process extracts the Probability Density Function (PDF) of the average value of each PI conditioned to the occurrence of a problem.

The creation of such a model will also permit the emulation of vectors, which is useful for obtaining additional training vectors for data mining algorithms. Emulated training vectors may be used for increasing the quality of the data mining process; for instance, by increasing the

Table 5.1: PDFs used in the fitting process

Distribution	PDF
Unbounded	
Normal	$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Laplace	$\frac{1}{2b}e^{\left(-\frac{\lvert x-\mu\rvert}{b}\right)}$
Gumbel-R	$\frac{1}{\beta}e^{-\left(z+e^{-z}\right)}$ where $z=\frac{x-\mu}{\beta}$
Gumbel-L	$\frac{1}{\beta}e^{z-e^{z}}$ where $z=\frac{x-\mu}{\beta}$
Semibounded	
Gamma	$\frac{\beta^{\alpha}}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$
Exponential	$\lambda e^{-\lambda x}$
Log-normal	$\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(lnx-\mu)^2}{2\sigma^2}}$
Bounded	
Beta	$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$
Arcsine	$\frac{1}{\pi\sqrt{x(1-x)}}$
Johnson SB	$\frac{\delta}{\lambda\sqrt{2\pi}z(1-z)}e^{-0.5(\gamma+\delta ln(z/(1-z)))^2}$ where $z=\frac{x-\chi}{\lambda}$

number of cases belonging to a class that has a small number of real examples and equalizing the proportions of each class in the training vector.

5.3.1 Modeling Process

In order to estimate a Probability Distribution Function (PDF) for a PI conditioned to a fault cause, the zeroes are removed from the list of values in the PIs that are defined above zero (such as counters) and modelled as a delta. This process is also done for ones in PIs that are proportions. The proportion of zeroes and ones in the samples is given as two parameters: P_0 and P_1. With the remaining values, a set of known PDFs are fitted to those values. Table 5.1 shows the possible distributions used for each type of PI, depending on them being unbounded (i.e. PIs that theoretically do not have a maximum or minimum value), semibounded (such as counters that are defined above zero) or bounded (PIs that have a maximum and a minimum such as proportions between 0 and 1). The list of tested distributions has been selected because of their resemblance with the normalized histograms that are usually observed for the data.

For each fitted PDF, a Kolmogorov-Smirnov (K-S) test [91] is performed to obtain a goodness of fit indicator (the D-Value that indicates the distance between the empirical distribution function of the PI values and the Cumulative Distribution Function of each tested distribution). The PDF with the lowest K-S statistic is selected for the model of the conditioned PI. An example of some tested PDFs is shown in Figure 5.13. With P_0, P_1 and the fitted distribution $p(x)$ with the highest D-Value, the PDF for the model of a PI $m(x)$ is given by:

$$m(x) = P_0\delta(x) + (1 - P_0 - P_1)p(x) + P_1\delta(x) \tag{5.5}$$

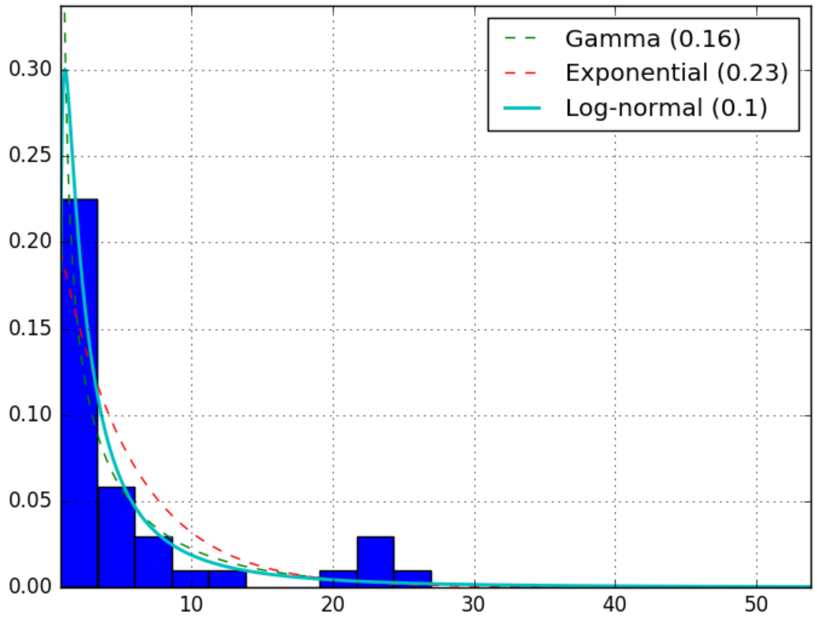


Figure 5.13: Histogram of *Average Number of Active UEs* conditioned to *High Traffic* with the fitted PDFs superposed.

5.3.2 Resulting Model

Appendix C shows the parameters of the estimated PDFs. The PDF selection algorithm is implemented using the Scipy library [92] under Python [85], and the fitted parameters are defined in [93]. Appendix B depicts all the standard PDFs fitted to each PI/problem pair, along with their D-Values.

As representative examples, Figure 5.14, shows the PDFs of some PIs. These PDFs show that for each PI, there is a high level of overlapping for different causes, i.e. the PI might have similar values under different causes. This characteristic complicates the design of diagnosis systems, causing a low performance in traditional rule-based algorithms. As expected, the PDFs for the *Number of ERAB Attempts* show great differences between the *No Traffic* and *High Traffic* classes, which are the two extreme cases for this PI. The *No Traffic* has a delta on 0, showing that in 42% of the cases ($P_0 = 0.42$), the value for this PI is 0. The *Low Coverage* and *CPU Overload* classes sit in a middleground, showing a behaviour that is close to the *Normal* class, since in these two classes the traffic is not a relevant feature. For the *Number of Bad Coverage Reports* PI, it can be observed that the *Low Coverage* class stands out as having a specially high values. In fact, once a problem is detected, this PI is what troubleshooting experts would consult to see if there is a high number of terminals with bad coverage. The next class with a higher number of reports is *High Traffic*, which may be explained because when the number of terminals increases, so does the number of potential bad coverage reports. Also, border cells may often be tagged as having a high traffic, since they cover a larger area containing terminals with a low signal reception. Another representative PI is *Number of CPU Overload Alarms*, where the *High*

CPU Load cases have the highest values. It is the only case not having a large delta on 0. The *High Traffic* class is the second with a high number of alerts. When the number of connections that an eNodeB processes is high, it is expected that the CPU load increases and ocassionally triggers alarms. In fact, the eNodeB software has the functionality to deal with these situations, so service to the already connected users suffers the least possible degradation. On the other hand, in the *High CPU Load* class, the overload alarms are triggered despite there being a low (or at least not high) number of connections. That is, the high CPU load is not traceable to a high number of managed connections. For the *No Traffic* problem there is actually no case with CPU overload alarms. Figure 5.13 shows the histogram of the *Average Number of Active UEs* conditioned to the occurrence of *High Traffic* along with the distributions fitted using the algorithm. Since it is a semibounded PI, Gamma, Exponential and Log-normal distributions are tested. In can be seen that the Log-normal distribution is the best for approximating the histogram, since its D-Value is the minimum.

5.3.3 Emulating Troubleshooting Cases

The model described in Section 5.3.2 on its own provides information about the behaviour of the PIs, and it can also be used to generate emulated vectors that represent DIs. For each emulated problem, a random number generator can be programmed with the conditional PDF of each PI, creating a plausible vector.

These vectors can be used for the training and testing of DM algorithms and diagnosis systems. Since in these cases, a large number of cases is required, emulation provides the possibility of multiplying the number of available cases starting from a reduced set of available cases; which is often the case in mobile networks.

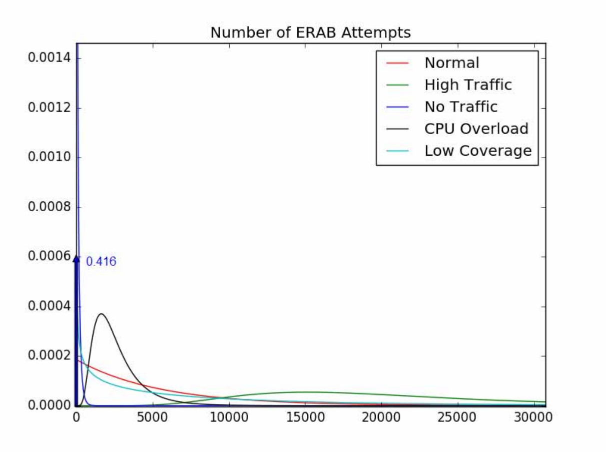
5.4 Conclusions

This Chapter has described the collected fault database and the process for adapting these data for DM.

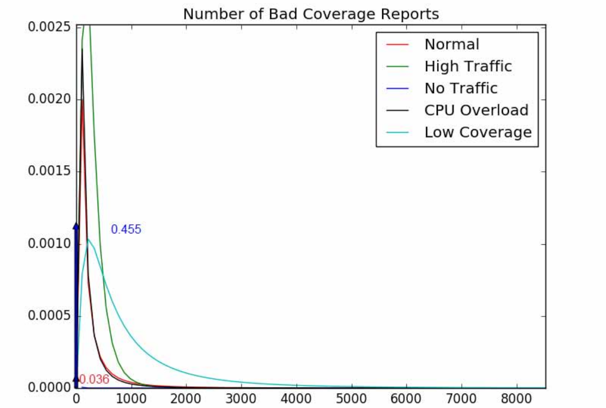
Firstly, the collected fault database has been described in full detail, specifying the collected problems and which cases are selected for the DM process.

Once the subset of the fault database used for KA has been selected, the data preparation stage is applied. Two processes have been described in this section: data cleaning, to fill missing values, and data reduction, to transform time-dependent matrices into single vectors representing occurrences of problems. The result is a dataset of vectors that can easily be used for DM.

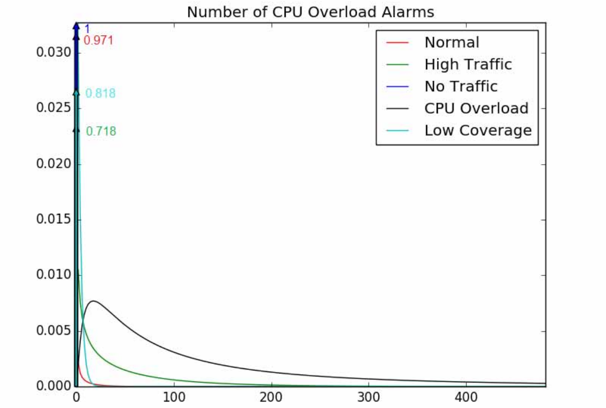
Finally, in this Chapter, a process for modelling a dataset of problems is described and applied over the collected problems. This model determines the PDFs of the PIs conditioned to the occurrence of each problem, and creates the possibility of generating new emulated vectors representing problems.



(a) Conditioned PDF of the *Number of ERAB Attempts* PI



(b) Conditioned PDF of the *Number of Bad Coverage Reports* PI



(c) Conditioned PDF of the *Number of CPU Overload Alarms* PI

Figure 5.14: Examples of PDFs obtained in the modelling process

Chapter 6

DATA MINING IN LTE

In KDD (Section 3.3), once data has been prepared (collected, cleaned and reduced as shown in Section 5.2), it is ready to be processed for the extraction of the target information in the Data Mining (DM) step. In the case of this study, the target information are diagnosis rules for LTE networks. These rules will be extracted from a set of CM/PM/FM vectors labeled with the problem. This Chapter starts with a brief introduction to DM and the types of algorithms that have been proposed. Next, the algorithms designed for this study are described in full detail and the tests that validate and compare them are shown.

6.1 Data Mining

To begin, the role of DM is framed in the overall process of KDD. Afterwards, the types of available DM algorithms are described, according to several taxonomies. The selection of the type of algorithm among these taxonomies is highly dependent on where and how is the DM algorithm used.

6.1.1 Introduction

As shown in Chapter 3 (specifically Section 3.3), KDD has five stages (selection, preprocessing, transformation, DM and interpretation/evaluation). The purpose of DM is to extract a model out of a data set by exploring the underlying patterns. Since a preprocessing has been previously done, the problems of data quality, heterogeneity and format are abstracted in this stage. In the case of applying KDD for KA, the model resulting from DM represents the expert knowledge on diagnosis.

The inputs of the DM stage in the KA process for network diagnosis are mainly the CM/PM/FM data (as described in Section 4.1.1) and other information on the behaviour of the eNodeBs that can be extracted in the preprocessing stage (as shown in Section 5.2). Each individual entry in the training set is a case that represents the occurrence of a problem, and it is

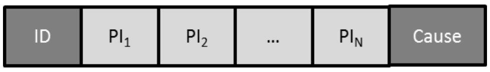


Figure 6.1: Format of a case

a tuple formed by an n-dimensional attribute vector (in this case, the values of PIs) and a class label that identifies the class to which the case belongs (i.e. the diagnosis), as shown in Figure 6.1. These vectors must be complete (i.e. no missing data) in order for the DM algorithms to work, and correct (i.e. as few diagnosis errors as possible and no wrong variable values) in order for the resulting model to be as accurate as possible.

The previous steps in KDD assure that these requirements are satisfied. Specifically, for mobile networks, a significant effort has been done in order to fill missing data and perform a format change in order to transform time dependent CM/PM/FM values into time independent vectors representing DIs. The only requirement that is out of the reach of the designed KA process is the correctness of the problem labels, which relies on the expertise of the troubleshooting experts.

One of the most important factors for designing the DM algorithm is the desired format for the output. Ultimately this depends on the nature of the patterns that are targeted and where the extracted information will be used. Very often, the extracted knowledge is meant to be presented to experts that are analyzing the data. In this case, the final step consists of formatting the data in a user-friendly manner, with sophisticated visualization tools (graphs, plots, tables, etc ...). The output of the KDD system may also be used in another automated system that is somehow related with the input data. In these cases, the last step will give the output of the DM process a specific format (coded instructions, executable files, etc ...) dependent on the requirements of the second system.

In the case of the KA system in this thesis, the latter is the main scenario; since the output of the KDD process is bounded to an FLC controller that performs diagnosis. The output of the DM process will therefore take the format described in Section 3.1.4, with a Data Base (DB) containing thresholds for the input values of the FLC and a Rule Base (RB) containing the diagnosis rules.

Once the input and output formats are settled, the DM design process must concentrate on obtaining the best possible results; but this is normally a difficult step that must be adjusted iteratively (as shown in Section 6.3). The design process will also take into account additional external constraints. These requirements depend strongly on the scenario where the KDD process is deployed. In the case of mobile network troubleshooting, some factors that must be taken into account are:

- Human factor: experts sometimes make mistakes when diagnosing. This is a very difficult issue and there really is no solution that can totally solve it. A wrong input will always produce a wrong output. Nevertheless, assuming that experts are experts for a reason, it can be considered that the input data has a very low proportion of wrong diagnoses. With some diagnosis error tolerance built in the DM process, the impact of a small proportion of problems can be reduced, with the possible cost of not covering very unsual situations.
- Economic factor: processing time costs money. Not only the occupation of resources is

costly, but also the results not being ready on time cause an opportunity cost. As it was stated in Section 4.1.2, the data sets obtained from a mobile network require a special treatment using Big Data techniques. This problem has already been partially dealt with in the data preparation step, but in the DM stage, it is also important to take it into account. From the point of view of the design, this factor calls for a parallelizable algorithm that can be executed on a cloud computing platform.

The human factor has some additional ramifications on the requirements for the output. For instance, the fact that experts will always trust (and therefore use) more a system that can be easily understood (such as a simple FLC with heuristic rules as opposed to a Neural Network with hidden layers) guides the choice for the output format. In this study, this was a major concern and the reason why FLCs were chosen.

The need for parallelization poses a practical requirement on the designed DM algorithm: the aggregated results of several instances of the algorithm must be equal to the results of a single instance. Also, in order to have a net benefit by parallelizing, the input variables must be processed independently (i.e. each single input value must be processed only once by any of the instances).

6.1.2 Types of Data Mining algorithms

Since KDD covers a broad range of problems, there are also many different algorithms for DM. There are several taxonomies that can help when choosing a DM algorithm for a problem. For instance, according to the type of problem approach:

- Classification: the DM creates a *classifier*, that is, a function that, given several variables as input, computes one value on the output that corresponds to a class in a set of finite classes. The FLC that is used for diagnosis is a type of classifier (considering that only the most likely diagnosis is always used). Classification problems are found everywhere: object recognition in images, insurance fraud detection, etc ...
- Regression: the output of the DM is a function that approximates the behaviour of an unknown function. Regression is used very often for finantial or weather forecasting.
- Clustering: The DM finds *clusters* in the data, that is, groups of individual data points that have something in common. This helps to understand the data better, since clustering can be done usually with a minimal prior knowledge of the data. Clustering is used in marketing applications for finding new trends; and it can also be used in mobile communications to find new, previously unknown, root causes.
- Summarization: The objective of the DM process in this case is obtaining a simplified *model* of the input data. An example of this type of DM is the modelling method shown in Section 5.3.1 (in fact, in this thesis there are really two parallel KDD processes that share the stages prior to DM: one for extracting a model of the fault database, and another one for creating an FLC based on the data). Usually the output of summarization is a human-readable report that provides insight of the input data.
- Dependency modelling: The DM finds the *dependency relations* among the input variables. Classification trees are an example of the output of these types of DM algorithms.
- Change and deviation detection: In these types of problems the focus is on finding *anoma-*

lies in the data. The DI detection algorithm described in Section 5.2.2 could be used in this phase as a DM algorithm in a problem of this type. Some examples of these type of problem are intrusion detection systems or extrasolar planet detection.

A certain problem can be classified in more than one type. For instance, to some degree, a classification problem can be considered a dependency modelling problem, since the relations among variables are ultimately used to create the classifier. The selection of the problem type will depend on how it is approached, and the choice will lead to a set of DM algorithms that are more appropriate for that approach.

DM algorithms (mainly those used for classification) can also be classified according to the information available at the input in two families:

- Supervised learning: when the input contains both information on the input and the output of the DM's target model. In the case of classifiers, the training set has both the values of the observed data and the class that each data point belongs to.
- Unsupervised learning: when there is no information on the desired output, either because it is unavailable or unknown. These cases usually call for clustering.

In this thesis, since it is assumed that the input contains the root cause of each case, a supervised learning algorithm can be used.

The implementation of the DM algorithm also creates a taxonomy, since some families of computing techniques can be used over a varied set of problems. This would be a very long list of techniques that include Neural Networks, Evolutionary Computing, Swarm Intelligence, Data Driven methods, etc...

The moment when the DM process takes place is also a relevant factor in the overall design of the KDD system. If the data is processed as it is received (i.e. when a new data point is received, the KDD updates its output), the DM is an *online* process, whereas if data is accumulated and only processed at a certain time or size of the input, it is an *offline* process. Usually online implementation tends to be used when speed is more important than accuracy, whereas offline processing is usually slower but more robust. The Lambda architecture described in Section 4.1.2 uses the advantages of both approaches for Big Data systems; using online DM in the speed layer and offline DM in the batch layer.

6.2 Designed Data Mining algorithms

This section describes in detail the two DM algorithms designed in this thesis and used for the extraction of the RB. After the description, a set of sensitivity tests will be done on the designed algorithms to study the effects of the parameters. The dataset used for the sensitivity tests will be also used with a commercially available DM algorithm that extracts a Bayesian Network for diagnosis in order to compare results. Next, manually processed real cases from the network will be used to compare the behaviour of the algorithms with real data. Finally, one of the algorithms is integrated in the full KDD process (together with another data mining algorithm for the DB, that is, the thresholds that determine when each PI is high or low), using the cases generated with the model exposed in Chapter 5 in order to extract rules that apply to the collected database.

The reason that there are two DM algorithms is that first, a genetic algorithm was developed and tested, and some shortcomings were observed. Specifically, the execution time was long, the algorithm was difficult to debug and the parallelization was not possible since the output of each thread could not be aggregated and produce the same output as an individual thread. This drove to the development of a second algorithm that was more dependent on the structure of the data, but also faster, simpler and parallelizable. Due to its improved capabilities, this data driven algorithm was chosen for inclusion in the KDD process.

6.2.1 Genetic Algorithm

Genetic algorithms [94] [95] imitate the process of natural selection, that is, they seek the solution of problems by a trial and error method, repeated generation after generation. All genetic algorithms have three common elements:

- A *population of individuals*, each being a possible solution to the problem. Each individual is a collection (*genome*) of traits (*gens*) that determine the behaviour of the solution and are subject to variation when the individual reproduces.
- *Operators* that define the birth of new individuals. Specifically, there are two big groups of operators:
  - Crossing: when the traits of two individuals are combined to produce a new improved individual.
  - Mutation: when a new individual is created by copying another individual and introducing minor (usually random) changes.
- A *fitness function* that assigns a score to each solution based on its quality.

A typical genetic algorithm has three processes:

- Reproduction: new individuals are created either by crossing two chosen rules, by mutation, or a combination of both. Usually, a high mutation rate produces a high exploration ratio (i.e. proportion of random search for new solutions as opposed to improvement upon already found solutions), since random changes are introduced. This will prevent the algorithm of finding only a subset of the possible solutions. In some algorithms, the parent rules are eliminated once the reproduction is done.
- Evaluation: the fitness of each individual is obtained using the fitness function. This value will determine the probability of the individual to survive and reproduce in future generations.
- Selection: This process decides which rules survive to pass on to the next generation. The rules with a higher fitness have higher chances of surviving.

The training data is a set of cases following the format shown in Figure 6.1. The values of PIs in the vector contained in the case are crisp.

The algorithm proposed in this work follows the flowchart shown in Figure 6.2 and implements the typical functions of genetic algorithms as follows:

- Individuals: each individual represents a rule. The genome of an individual has two components:
  - Antecedent (the "*if* ..." part): a vector, $\{a_1, a_2, ..., a_L\}$, of length L equal to the number of inputs of the diagnosis system. Each entry a_i in this vector represents

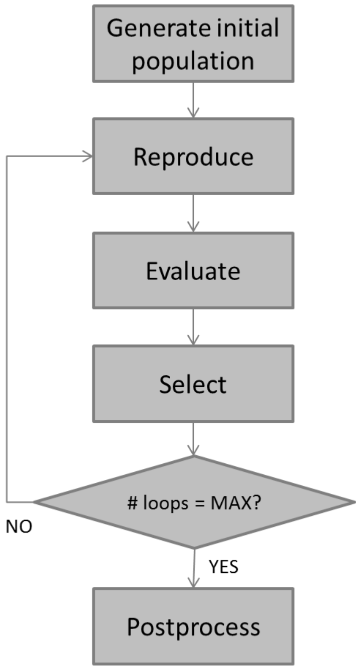


Figure 6.2: Flowchart of the algorithm

the fuzzy value that the i-th PI (PI_i) should take for the rule to be activated. If PI_i should be *"low"*, then $a_i = 1$; if it should be *"high"*, then $a_i = 2$, and if the rule does not require any particular value for PI_i, then $a_i = 0$. The last value of a_i allows the existence of incomplete rules. Incomplete rules are calculated at the end of the algorithm, and are also used in the process of reproduction, but there are no individuals representing incomplete rules during the execution of the algorithm.

  - Consequent: (the "*then* ..." part): an integer, d, representing the problem diagnosed by the rule for the fuzzy values of the antecedent.

Each individual has an associated score W that represents its fitness. The initial population is generated with random valid values in the antecedent vector and the consequent.

- Reproduction: the rules are chosen randomly to reproduce. Crossing and mutation are done separately. For each individual, the probability of being chosen for reproduction by crossing is given by $P_{cross} = W \cdot R$, where W is the score of the rule, and R is the reproduction ratio. Only the best rules are chosen for crossing, so the best traits are passed on. For mutation, the probability is $P_{mut} = 1 - P_{cross}$, so the worst rules (that have not been killed) are given a chance to improve.

  Crossing (Figure 6.3) happens between parent rules that have the same consequent. A random cutting point $c_p \in [0, L-1]$ is chosen. Each parent rule antecedent is divided into two fragments $[a_0 ... a_{c_p}]$ (first fragment) and $[a_{c_p+1} ... a_L]$ (second fragment). A score is given to each of the four fragments (two per parent). For this purpose, each fragment is extended with zeros into a valid antecedent of its own, and a partial rule is created with this antecedent. The partial rules are then evaluated with the training cases to see which of them are activated more often by calculating the average *degree of activation* ($\overline{DoA}$).

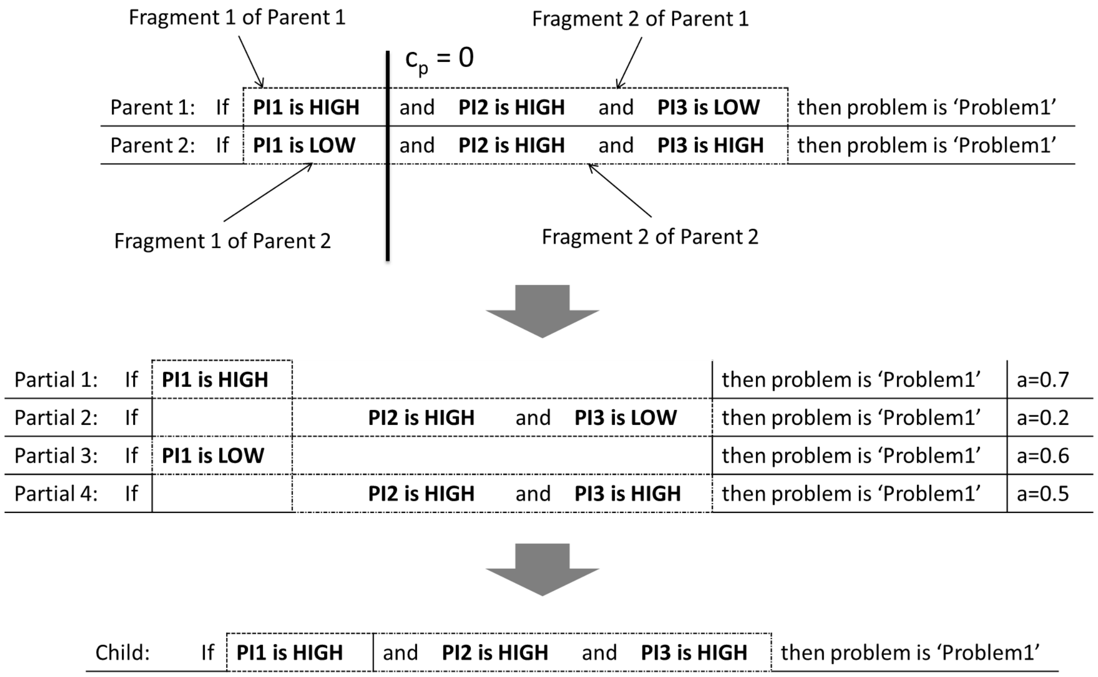

Fragment 1 of Parent 1
Fragment 2 of Parent 1
$c_p = 0$
Parent 1: If PI1 is HIGH and PI2 is HIGH and PI3 is LOW then problem is 'Problem1'
Parent 2: If PI1 is LOW and PI2 is HIGH and PI3 is HIGH then problem is 'Problem1'
Fragment 1 of Parent 2
Fragment 2 of Parent 2
Partial 1: If PI1 is HIGH then problem is 'Problem1' a=0.7
Partial 2: If PI2 is HIGH and PI3 is LOW then problem is 'Problem1' a=0.2
Partial 3: If PI1 is LOW then problem is 'Problem1' a=0.6
Partial 4: If PI2 is HIGH and PI3 is HIGH then problem is 'Problem1' a=0.5
Child: If PI1 is HIGH and PI2 is HIGH and PI3 is HIGH then problem is 'Problem1'

Figure 6.3: Example of crossing two rules to obtain a child.

The *DoA* of a rule for an input vector is the degree of truth of the antecedent. A partial score (a) is assigned to each fragment:

$$a = \overline{DoA}_{fragment} \cdot W_{parent} \tag{6.1}$$

where W_{parent} is the score of the parent rule. The child is then created with an antecedent that is the combination of the best (highest partial score) first fragment with the best second fragment. The parents and the child are kept in the population.

The mutation process creates several copies of an individual, each with one gene randomly altered. Either one of the components of the antecedent vector or the consequent are given a random value between the valid values it can take.

- Evaluation: the score W of the rules is calculated as the product of two separate terms:

$$W = B \cdot S \tag{6.2}$$

The *Base (B)* represents the statistic relevance of the rule and the *Success Rate (S)* represents the accuracy of the rule, that is the percentage of covered cases fulfilling the rule. The Base is defined as:

$$B = 1 - \frac{1}{1 + \alpha \frac{c}{N}} \tag{6.3}$$

Where α is a parameter that adjusts the sensitivity of the Base to uncommon cases (the higher is α, the easier it is to obtain a high Base even with a low number of covered cases), c is the number of covered cases and N is the total number of cases in the training set. A case is considered covered when the *DoA* of the antecedent for that case is greater than a

	Rule 1:	If	PI1 is HIGH	and	PI2 is HIGH	then problem is ‘Problem 1’
+	Rule 2:	If	PI1 is LOW	and	PI2 is HIGH	then problem is ‘Problem 1’
	Rule 3:	If			PI2 is HIGH	then problem is ‘Problem 1’

Figure 6.4: Example of fusion of two rules.

predefined threshold *Minimum Degree of Activation (MDA)*.

- Selection: the algorithm finds the rules that have a score equal to 0 and eliminates them.
- Postprocessing: Rules are grouped according to their consequents, and they are fused into incomplete rules. Two rules are fused if they are both correct. A rule is considered to be correct if its score is higher than the *Minimum Inclusion Degree (MID)*. In that case, the contradictions between them are resolved by ignoring the non-common parts of the antecedent. Rules whose score is lower than MID are removed. An example of rule fusion is shown in Figure 6.4. Since Rule 1 and Rule 2 have different restrictions over *PI1*, and if both rules are correct, then it will be considered that the value of *PI1* is not relevant for detecting the cause *'Problem 1'*. *PI2* has the same restrictions in both rules, so the observation of a *HIGH* value in this PI leads to the detection of *'Problem 1'*.

All these elements and processes describe the operation of the algorithm. Since the operation of many of the individual processes depend on specific values, a set of 7 configurable parameters is provided to adjust the behaviour of the algorithm and fine tune the results:

- *Uncommon case sensitivity* $(\alpha) \in (0, +\infty)$: adjusts the sensitivity of the algorithm to uncommon cases. A too low value may disregard cases that are important although they are uncommon. A too high value may give relevance to cases that are wrongly diagnosed.
- *Minimum inclusion degree (MID)* $\in (0, 1)$: minimum score a rule must have to be part of the final rule set.
- *Minimum degree of activation (MDA)* $\in (0, 1)$: indicates the minimum *DoA* of a rule on a case to consider that the rule covers it.
- *Initial rules* $\in [1, +\infty)$: Number of random initial rules. If none of the initial rules survives the first iteration of the algorithm, an *extinction* of the population occurs and the algorithm stops.
- *Number of loops* $\in [1, +\infty)$: Number of iterations of the genetic algorithm. The number of loops must be high enough for the algorithm to converge to a correct solution.
- *Reproduction ratio (R)* $\in (0, 1]$: the proportion of cases that will reproduce (supposing all the cases have $W = 1$). This parameter is a population growth control.
- *Mutation multiply factor (MMF)* $\in [1, +\infty)$: Number of children that a rule produces when mutation takes place. A small value will produce a low exploration rate, whereas a too high value will increase the execution time.

6.2.2 Data Driven Algorithm

The second developed algorithm is a faster data driven DM method. The input and output formats are the same as for the genetic algorithm (i.e. both algorithms are interchangeable), although the results do not necessarily match, especially since the genetic algorithm has a random underlying nature. The development of this algorithm is also focused from the beginning on parallelization in order to cope with Big Data issues (Section 3.4). The new data driven learning algorithm is based on the WM [96] method, introducing some modifications (i.e. the formula for assigning a score to each generated rule taking into account parallelization and the procedure for fusing multiple rules into one per diagnosis) in order to better adapt to the obtention of the rules used in troubleshooting of LTE networks. The WM method is a well known algorithm that can easily be parallelized since the creation of new rules from the data is independent from the creation of previous rules. Therefore, the data can be divided among several *worker processes* arbitrarily without loss of information. Each worker process is an independent instance of the algorithm running as a separate execution thread. A final step can fuse the results of different worker processes, allowing them to run in parallel over different processors or cores. Another advantage of the WM method is that it is deterministic, that is, the results of two equal training sets are always equal, independently from the order that the data is provided or how it is divided among parallel processes.

The algorithm obtains the RB of an FLC from the training set composed of labeled cases. Given each case as the tuple $C = (\overline{k}, d)$ composed of the PI vector $\overline{k} = \{k_1, k_2...k_N\}$ representing the values of PIs $\overline{PI} = \{PI_1, PI_2...PI_N\}$ and a label d as the class label among possible root causes $\overline{RC} = \{RC_1, RC_2...RC_M\}$. The algorithm has three consecutive steps:

1 Generate fuzzy rules from training tuples: the variable values in $\overline{k}$ are assigned the fuzzy sets where their truth degree is the highest, creating a vector $\overline{k_F}$; that is, for each $k_n \in \overline{k}$ a new fuzzy value $k_{F_n} = T_n \mid \mu_{T_n}(k_n) = max(\mu_{T_{1n}}(k_n), \mu_{T_{2n}}(k_n))$ is defined, where T_{1n} and T_{2n} are two fuzzy sets identifying opposing qualitative states (such as *high* or *low*) of PI_n, $\mu_{T_{1n}}(k_n)$ and $\mu_{T_{2n}}(k_n)$ are the membership functions of T_{1n} and T_{2n} defined over the domain of PI_n and evaluated for k_n and T_n is the chosen state. In the unlikely case that both membership functions have the same value for k_n, two different antecedents will be created, in order to cover both cases. The label d is also assigned the set $RC \mid \mu_{RC}(d) = max(\mu_{RC_m}(d)) \forall RC_m \in \overline{RC}$ representing the root cause. This process is depicted with an example in Figure 6.5. A training set with two variables (PI_1 and PI_2) and a class label (d) is depicted. The variables take values k_1 and k_2 respectively. The truth degree for each fuzzy set is calculated for these values, and the variables and the label are assigned the set with the highest truth degree. In classification, the consequent will always have a membership degree of 1 in the set representing the class and 0 in the others.

With the fuzzy linguistic labels $R = (\overline{k_F}, RC)$ established in this step, an *AND* rule is created: *"if* PI_1 *is* k_{F1} *and* $PI2$ *is* k_{F2} *... and* PI_N *is* k_{FN} *then root cause is* RC*"*. Once the rule is created, the training set is explored for cases that are covered by the same rule (that is, cases that are identical once fuzzified) with a certain degree of activation $a = min(\mu_{T_n}(k_n)) \forall k_n \in \overline{k}$ where T_n is the fuzzy set assigned by the rule for PI_n). A list keeps track of the cases that have been covered by a rule. The marked cases are not

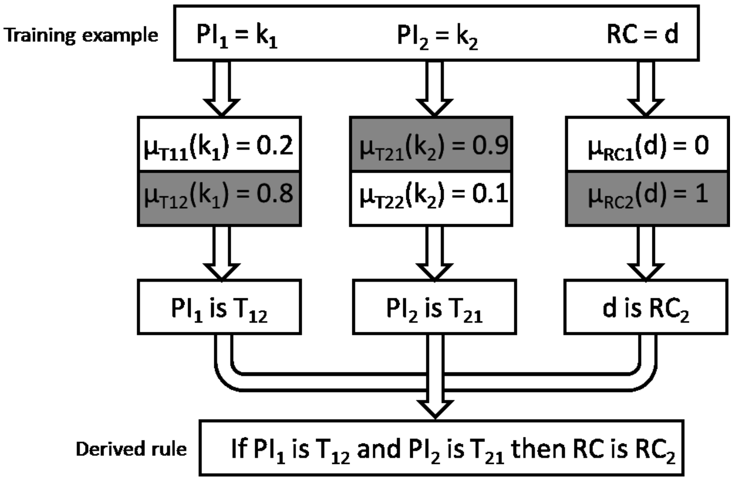


Figure 6.5: Rule creation process. A training set with two input variables (PI_1 and PI_2) and one output variable (RC) is depicted. These variables take values k_1, k_2 and d respectively. The truth degree for each fuzzy set is calculated for these values, and the input variables are assigned the set with the highest truth degree. With these sets, a rule is created

used for generating rules in future iterations, so the same rule is not generated more than once and also to reduce the computation time and memory footprint. The list is a boolean indexed list, so there is one entry for each rule. Each entry is started with a *False* value, since no case is covered in the beginning. Once a case is used for generating a rule, or is found to be covered by a rule, the corresponding entry in the list is changed to *True*. This step can be parallelized by dividing the learning set. This may lead to repeated rules among different worker processes, but this will be solved in the next step.

2 Assign a score to each rule representing its confidence. The score of each rule is composed of two terms: the *Base* and the *Success Rate*. Just like in the genetic algorithm, the Base represents the statistical significance of the antecedent of the rule. Rules that cover few cases get a lower score, so spurious cases (or human errors) are filtered out. The Base is given by $B = 1-(1/(1+\alpha c/N))$, where c is the number of cases covered by the antecedent, N is the total number of cases and α is an adjustable parameter that can take values in the $(0, +\infty)$ interval. The Success Rate is the number of successfully diagnosed cases n_s over the number of covered cases in the training set $S = n_s/c$. In the first pass through the training set, $S < 1$ only if there are contradicting cases (same fuzzified KPI vector, different diagnosis). The score is given by $W = B \cdot S$. Thus, when the number of covered cases is small, the score of a rule is limited by the Base term; and if it gets statistical significance, the score is determined mostly by the Success Rate. To compute the score of each rule, it is required that the equal rules found by different workers have their bases aggregated with $B = \sum_{w=1}^{N_w} B_w$ where N_w is the total number of workers and B_w is the Base found for worker w. Their success rates must conversely be aggregated by $S = \sum_{w=1}^{W} \frac{B_w}{B} S_w$, where S_w is the Success Rate found by worker w. The rules, along with their total scores, are stored in a common rule base prepared for the next step.

3 Reduce the number and complexity of the rules: The rules obtained so far are *complete rules*. To obtain incomplete rules, several complete rules are *fused* together. The rules to be fused are required to have the same consequent and a score higher than a minimal threshold, to avoid the inclusion of spurious or incorrect rules. Given two rules R_1 and R_2 with the same consequent RC_C, a new rule $R_{1+2} = (\overline{k_{F1+2}}, RC_C)$ is defined where $\overline{k_{F1+2}} = \overline{k_{F1}} \cap \overline{k_{F2}}$. Once two rules meet the requirements and are fused, the score is calculated again by testing over the training set. This step is the same as in the genetic algorithm, depicted in Figure 6.4, and in this case it can also be parallelized if each worker takes a subset of the original rules and a subset of the data to calculate the score of the fused rules that it calculates. Once all the rules that comply with the conditions for rule fusion are fused, the scores are aggregated over all the workers as explained in Step 2, and a new reduced rule base is created. This step is iteratively repeated until there are no more possible rule fusions.

In the case that successive fusions generate an empty antecedent, it is considered that the particular root cause is not diagnosable with the current set of PIs, since there are occurrences of the problem with every possible combination of fuzzy values of the PIs.

The rules that do not meet a minimum score requirement are also removed from the RB in this step. The possible conflicts between rules (same antecedent, different consequents) are solved by selecting the rule with the highest score.

In Figure 6.6 the flowchart of the algorithm is depicted.

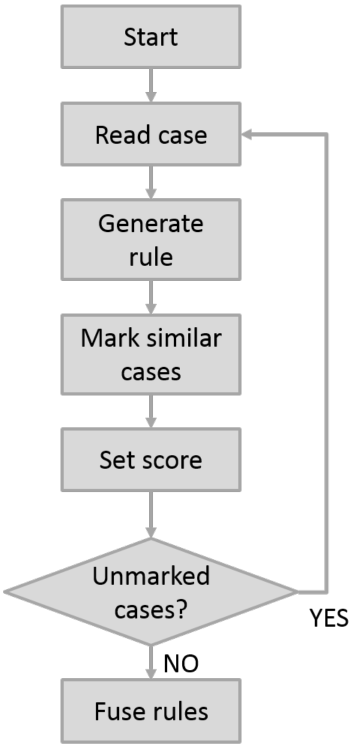


Figure 6.6: Flowchart of the algorithm.

This algorithm has three parameters which are common with the genetic algorithm described in 6.2.1:

- *Uncommon case sensitivity (α)* $\in (0, +\infty)$: sensitivity of the algorithm to uncommon cases, used in the scores assigned in step 2 to the generated rules.
- *Minimum Inclusion Degree (MID)* $\in (0, 1)$: minimum score a rule must have to be used in the process of fusion with other rules of the same consequent. Effectively, it is the minimum threshold for the score of a rule to be included in the final rule base.
- *Minimum Degree of Activation (MDA)* $\in (0, 1)$: minimum DoA for a case to consider it as part of the Base of a rule in step 2.

6.3 Tests

In this Section, the tests performed over the algorithms are shown. These tests are divided into two parts for each algorithm: a sensitivity test to assess the behaviour of the configuration parameters and tests with the real cases extracted from the collected database described in Chapter 5.

6.3.1 Description of the tests

Both algorithms have numerous configuration parameters. In order to show the effect of each one, tests with a specific data set are done sweeping values on each parameter one by one. The dataset used for this purpose is generated by an emulator using a theoretical model. The generated dataset has 5 PIs:

- *Accessibility*: it reflects the ability to establish a connection in the network. It is the inverse of the *Blocking Rate*. Its values range between 0 and 100%, and it is considered *normal* above 99% and *low* below 98%. These values are a commercial requirement.
- *Retainability*: it reflects the ability to end a call correctly. It is the inverse of the *Dropped Call Rate*. Its values range between 0 and 100%, and it is considered *normal* above 99% and *low* below 98%. These values are also a commercial requirement.
- *Handover Success Rate (HOSR)*: percentage of initiated handovers that end successfully. Its values range between 0 and 100%. It is considered *normal* above 98.5% and *low* below 95%. These values are also a commercial requirement.
- *95 percentile of $RSRP$*: value of $RSRP$ under which 95 percent of the samples fall. Experts consider it *low* if it is below -100 dBm and *high* if it is above -80 dBm.
- *95 percentile of $RSRQ$*: value of $RSRQ$ under which 95 percent of the samples fall. Experts consider it *low* if it is below -23 dB and *high* if it is above -12 dBm.

The model used by the emulator is not the same as the one described in Section 5.3 because at the time of the design of the algorithms, the number of collected cases was too low to create a model. Each PI is modelled in a similar way as described in Section 5.3, but with different PDFs. Instead of extracting these PDFs from data, they were provided by experts. Specifically, Table 6.1 describes the used PDFs. With this model, 2000 training cases (containing 600 problems and 1400 normal cases) and 5000 testing cases (1500 problems and 3500 normal) have been generated. Both sets contain the proportions for each problem defined in Table 6.2. The algorithms are first trained with the training cases, and afterwards, they are tested with the testing cases. This

Table 6.1: PI modelling. PDF of each PI conditioned to an existing problem

PI	Type		Parameters/Cause				
			Nor	SW	Cov	Qual	Mob
Acc.	beta	α	2	12	1.391	450.3	2
		β	0.1	3	0.028	23. 7	0.5
Ret.	beta	α	17	11.756	10	11	9
		β	0.5	1.306	1.5	1.9	2
HOSR	beta	α	4	4.62	3	5	42.5
		β	0.02	0.024	0.02	0.04	7.5
RSRP	norm	avg	-70	-75	-107	-72	-80
		σ	3	6	5	7	10
RSRQ	norm	avg	-6.5	-6	-10	-13	-11
		σ	1.1	2	5	2	3

Table 6.2: Proportions of problems

Fault category	Proportion (%)
SW Problem	13
Coverage	25
Quality	34
Mobility	28

process is repeated 100 times and the average values are calculated for the genetic algorithm. This repetition of experiments is not necessary for the data driven algorithm due to its deterministic nature.

6.3.2 Genetic Algorithm sensitivity tests

These tests evaluate the influence of the most important parameters of the genetic algorithm.

Initially, a random population of 5 rules is generated to seed the algorithm. The genetic process is repeated for 50 generations; that is, 50 cycles of reproduction, evaluation, selection and death. Table 6.3 shows the tested values for each parameter.

Test 1: α

This experiment finds the influence of α. This variable regulates the sensitivity of the algorithm to rare cases. A small value of α gives a low score to rules that cover uncommon cases. A higher value lets the score of a rule grow rapidly as its Base increases.

The Diagnosis Error Rate, Undetected Rate and False Positive Rate are depicted in Figure 6.7. The Diagnosis Error Rate is not very sensitive to the value of α. The False Positive Rate increases with α, whereas the Undetected Rate decreases with α. A value of α between 12 and 18 provides a good compromise in the values of both the Undetected Rate and the False Positive Rate. However, if minimizing the Undetected Rate is considered more important than minimizing the False Positive Rate, then a high value of α should be selected.

Test 2: MID

Table 6.3: Parameter values for the genetic algorithm

Test	Variable	Default value	Tested values
1	α	24	6, 12, 18, 24, 30, 36, 42, 48, 54, 60
2	MID	0.2	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
3	MDA	0.4	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
4	Reproduction ratio	0.5	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
5	MMF	5	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
6	Training cases	Only problematic	Only problematic, problematic + normal

The MID parameter determines the minimum score that a rule should have to be fused with other rules to produce more general rules. The results are shown in Figure 6.8.

The Diagnosis Error Rate and False Positive Rate decrease as MID increases. For $MID = 0.9$, both error rates are zero. Nevertheless, the Undetected Rate is 1, which means that absolutely no problem is diagnosed; there is no actual diagnosis system, as shown in Figure 6.9, where the average number of rules is depicted. As MID increases, the number of rules decreases. A high MID is more restrictive, thus including less rules in the final rule set. For $MID = 0.9$ there is no rule with the required score to be included in the final rule set, so nothing is diagnosed. There is a direct relation between the number of rules and the Undetected Rate. Likewise, as the number of rules decreases, the False Positive Rate also decreases, as less rules produce a smaller chance of a wrong diagnosis. Given the values $P_n = 0.7$, $P_p = 0.3$, and taking the values of E_{fp} and E_u for the default value of $MID = 0.2$ (0.3179 and 0.0075, respectively), according to Equation 3.6, $P_{fp} = 0.4277$. That is, 42.77% of the times, when the system indicates positive, it is a false alarm. This is for a network where 30% of the sectors have problems. For a better network, that number would increase. This stresses the need of a detection phase that filters normal cases, avoiding their analysis by the diagnosis fuzzy logic controller. Supposing that an ideal detection phase is used, $P_n = 0$ and $P_p = 1$ by definition. Applying Equation 3.6, $P_{fp} = 0$.

Anyway, a detection phase might still have a small error rate that lets some false positives pass, so it still makes sense to try to keep a low False Positive Rate, even though this criterion is not the priority. Since the Diagnosis Error Rate is relatively insensitive to the variation of MID, the main criterion will be to minimize the Undetected Rate. As a conclusion, to keep a low Undetected Rate, MID should be low (around 0.2).

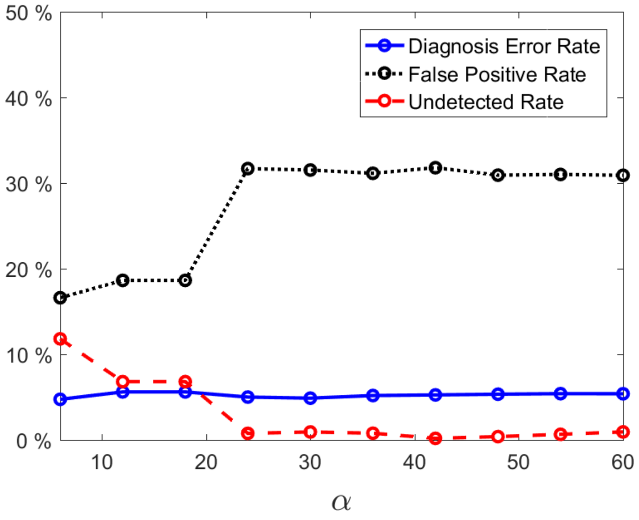


Figure 6.7: Error rates for variable α.

Test 3: MDA

This experiment evaluates the influence of the MDA parameter over the error rate. This parameter regulates the minimum degree of truth of an antecedent for a case to consider it covered. This modifies the Base of the rules, and consequently their scores. Figure 6.10 depicts the results.

Again, the trade-off between Undetected Rate and False Positive Rate is observed. Since a higher MDA is more restrictive, the effect in the number of rules is similar to the case of MID, as Figure 6.11 shows.

The best value of MDA depends on the priorities when tuning the algorithm. To obtain a minimum Undetected Rate, the value for MDA should be lower or equal to 0.3. Nevertheless, for values 0.1 and 0.2, both the Diagnosis Rate and False Positive Rate are at their maximum. For 0.3, the Diagnosis Error Rate decreases to almost a third of its maximum value. Therefore, 0.3 is a good choice that has a low Undetected Rate and a relatively low Diagnosis Error Rate. To obtain a low False Positive Rate a good choice would be a value of 0.5 for MDA, which has a slightly worse Undetected Rate. Since the addition of a detection stage would eliminate (or at least reduce) the number of normal cases in the input of the FLC, the False Positive Rate would no longer be an important factor. The Undetected Rate would be transformed into the proportion of cases that are known to be problematic, but cannot be diagnosed. Therefore, the minimization of the Undetected Rate should be prioritized over the minimization of the False Positive Rate.

Test 4: Reproduction ratio

The reproduction ratio adjusts the probability of reproduction for each rule. A high value of this parameter increases the number of combinations tested in the algorithm, at the cost of an increased execution time. In Figure 6.12 the measured errors are represented. The execution time with different reproduction ratios relative to the execution time taken with the default configuration is shown in Figure 6.13.

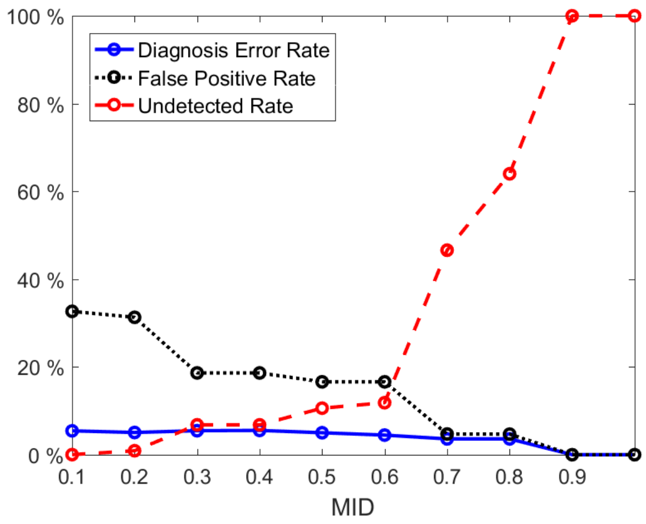


Figure 6.8: Error rates for variable MID.

The results show that for a low reproduction ratio (lower than 0.5), the Undetected Rate decreases as the reproduction ratio increases. At the same time, the False Positive Rate increases. For values higher than 0.5, all the rates are stagnant. On the other hand, the relative execution time increases as the reproduction ratio grows. Therefore, a value of 0.5 for the Reproduction Rate is the optimal, since it offers results as good as those obtained for higher ratios, without increasing the execution time.

Test 5: MMF

The MMF adjusts the number of rules that are created each time that a rule reproduces by mutation. A high value of MMF provides an increased exploration capacity, but also increases the execution time as it adds rules to the system. Figure 6.14 depicts the error rates as a function of the MMF and Figure 6.15 shows the relative execution time as compared to the configuration with a default value for MMF ($MMF = 5$).

As the number of mutated offspring grows, the algorithm produces better results. The Diagnosis Error Rate is more or less stagnant all along the executions. The Undetected Rate shows a large decrease as the number of mutated offspring grows, and the False Positive Rate grows at the same time. The increased exploration provided by this method greatly influences the results.

For values of MMF higher than 6, the Undetected Rate reaches its minimum. Therefore, for values lower than 6, the higher MMF is the better the solution, at the cost of computing time. For values higher than 6, the quality of the solution does not improve, despite the increase of execution time.

Test 6: Training using normal cases

In all the previous tests, the training was done using only cases that had problems. In this experiment, the training will be done with the default values of all parameters (Table 6.3), but using normal cases in the training phase together with the problematic cases. Also, to isolate

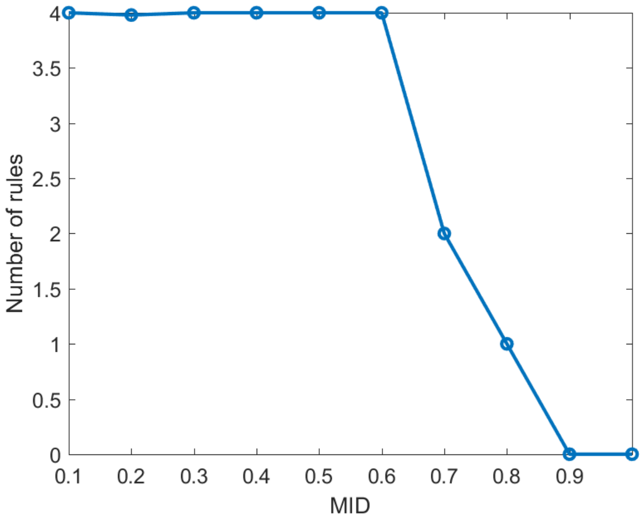


Figure 6.9: Number of output rules for variable MID.

the effects of α, the default value is used when not using normal cases, and $\alpha \cdot \frac{N_p+N_n}{N_p}$ when using normal and problematic cases in the training. When using this modified value of α in Equation 6.3, the absolute number of cases c that a rule must cover in order to obtain a certain Base B is the same. Therefore, the variations in the results are only dependent on the fact of using normal cases in the training. The results are depicted in Figure 6.16.

The Diagnosis Error Rate and Undetected Rate suffer a slight degradation, whereas the False Positive Rate is clearly improved. This is due to the the fact that the algorithm evolves the rules with knowledge about normal cases. Therefore, it will be less likely that rules that classify normal cases as problematic are considered valid. Thus, the rules are more evolved towards avoiding false positives. The improvement of the False Positive Rate comes at the cost of an increased execution time, due to the increase in the number of individual applications of rules over cases when the evaluation in the genetic loop is done (appr. 3.5 times higher execution time when including normal cases). When no normal cases are used in training, each individual rule is evaluated on 30 percent of 2000 cases (600), and when normal cases are used, each rule is evaluated over all the cases. The decision of using normal cases in training or not will depend mostly on whether there will or will not be a preliminary detection phase. In case there is, since false positives are not a problem, the training should be done without normal cases, because of the reduced execution time and slightly better error rates. On the other hand, if no detection phase is used, reducing the False Positive Rate will result in a more usable diagnosis algorithm.

6.3.3 Data Driven Algorithm sensitivity tests

These tests evaluate the behaviour of the parameters of the data driven algorithm. The simulated dataset is the same as the one used for the genetic algorithm sensitivity tests. Table 6.4 shows the tested parameter values.

Test 1: α

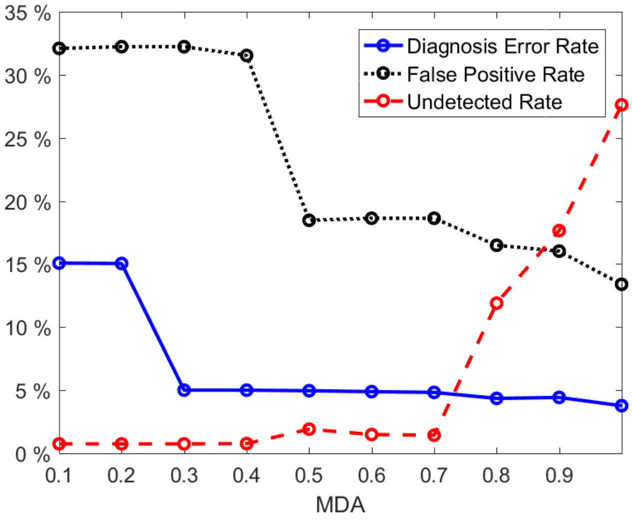


Figure 6.10: Error rates for parameter MDA.

Table 6.4: Parameter values for the data driven algorithm

Test	Variable	Default value	Tested values
1	α	48	6, 12, 18, 24, 30, 36, 42, 48, 54, 60
2	MDA	0.4	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
3	MID	0.1	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

This experiment finds the influence of α. This variable regulates the sensitivity of the algorithm to rare cases. Just as in the case of the genetic algorithm small value of α gives a low score to rules that cover uncommon cases and higher value lets the score of a rule grow rapidly as its Base increases.

The Diagnosis Error Rate, Undetected Rate and False Positive Rate are depicted in Figure 6.17.

The Diagnosis Error Rate and Undetected Rate decrease as α increases. Both errors are maximum for $\alpha = 6$, whereas the False Positive Rate is minimum for this value. The False Positive Rate grows with α. The increase in α means that the score increases rapidly as the Base of a rule grows. This increases the diversity of rules that achieve the minimum score to be integrated in the final RB; resulting in a RB that covers more cases (lower Undetected Rate). This comes at a cost, because the larger number of covered cases (especially rare cases) drives to an increase in false positives up to 34.5%. Therefore, there is a trade-off between the Undetected Rate (reliability) and the False Positive Rate, which has a great influence on the

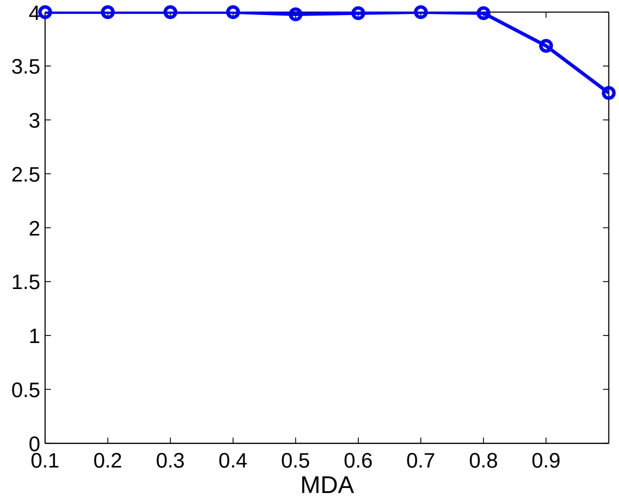


Figure 6.11: Average number of rules for MDA parameter.

performance of the resulting FLC. To illustrate the relation between the False Positive Rate and the performance, by using Equation 3.6 the complementary of the Positive Predictive Value can be obtained, reflecting the probability that a certain diagnosis is a false positive. With a proportion of normal and problematic cases of 70% and 30% respectively, and the results for the default values (E_{fp}=0.345, E_u=0) the probability results in P_{fp}=0.446 (44.6%). This remarks the importance of using a detection stage prior to the diagnosis. This stage should separate the abnormal cases from the normal cases. Supposing an ideal detection stage, the False Positive Rate would have no meaning anymore. The Undetected Rate would reflect the proportion of cases that the diagnosis system cannot classify; but at least those cases are detected and can be diagnosed manually.

Summarizing, the optimal value of α would be $\alpha > 6$, since for these values, the Undetected Rate is almost 0. Between 12 and 48, the Diagnosis Error Rate is around 5%. For $\alpha \geq 54$, the Undetected Rate is 0, but the Diagnosis Error Rate rises to 6%. The default selected value is 48%, because for that value both Undetected Rate and Diagnosis Error Rate are low.

Test 2: MID

The MID parameter determines the minimum score that a rule must have to be fused with other rules to produce more general rules. The results are shown in Figure 6.18.

The Diagnosis Error Rate shows a stagnant behaviour as MID increases. For $MID = 1$, the Diagnosis Error Rate is 0, but as observed in the Undetected Rate, there are no diagnosis in that case. MID determines the minimum score that a rule must have to be included in the final RB, so a high value of MID reduces the diversity of the RB. Since no rule reaches a score of 1, there is no output RB for $MID = 1$. This is expected, since an increase in MID means an increase in restrictions for including rules that cover rare cases (also for rules that have a low Success Rate), the Undetected Rate grows when MID increases, and the False Positive Rate decreases. The best interval for this parameter is $0.1 \leq MID \leq 0.3$, because the Undetected Rate is minimum.

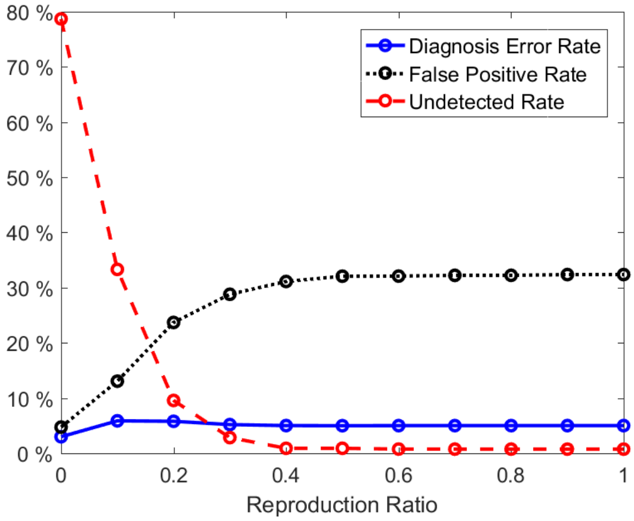


Figure 6.12: Error rates for the reproduction ratio.

Test 3: MDA

This experiment tests the influence of the MDA parameter over the error rate. This parameter regulates the minimum degree of truth of an antecedent for a case to consider it covered. This modifies the Base (that is, the term of the score of the rule that depends on the number of covered cases) of the rules, and consequently their scores. Figure 6.19 depicts the results.

It is observed that the Undetected Rate and the False Positive Rate remain almost stagnant for all values of MDA. The Undetected Rate grows slightly as MDA increases. The Diagnosis Error Rate is high for low values of MDA (27.67% for $MDA = 0$), and decreases until it reaches a minimum between 0.4 and 0.9. For $MDA = 1$, the Diagnosis Error Rate increases slightly. Since the Diagnosis Error Rate shows the classification errors, its increase shows that the rule that identifies a problem A is actually also covering certain instances of another problem B. Since a high value of MDA is restrictive for cases that have not very clearly classified PIs, it means that some rules that should have appeared have not been created, and therefore, the cases covered by them are confused with a different cause. On the other hand, for low values of MDA, the restrictions of loosely covered cases are lower, so some rules that cause confusion (i.e. they cover cases that should not be covered) are created and included in the final rule set, therefore increasing the classification error.

Summarizing, the recommended values for this parameter are $0.4 \leq MDA \leq 0.5$, which minimize the Undetected Rate and Diagnosis Error Rate.

Test 4: Training using normal cases

The tests with the default parameters have been repeated using the normal cases in the training set. The result with the three measurements is shown in Figure 6.20.

When using the normal cases in the training, the Diagnosis Error Rate and the Undetected Rate grow slightly, whereas the False Positive Rate decreases significantly. To better visualize the meaning of this change, the calculation described in Equation 3.6 is used, and results in P_{fp}=0.347 (34.7%, against 44.6%). Although there is a slight improvement in P_{fp}, the gain is

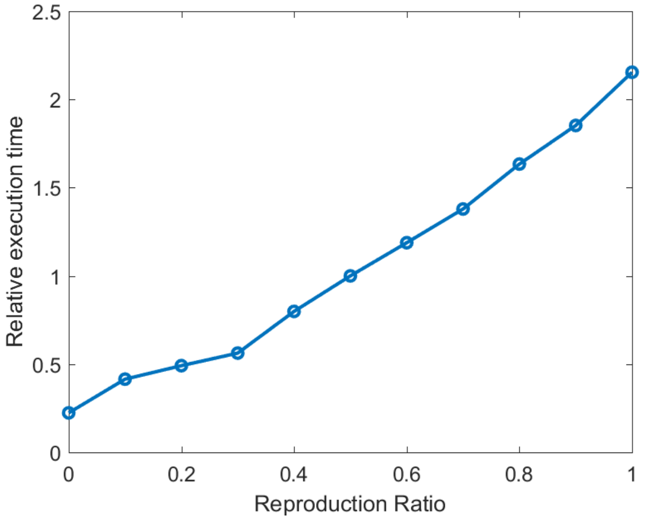


Figure 6.13: Relative execution times for different values of the reproduction ratio.

insignificant, since the system still needs a detection stage. This small gain comes at the cost of an increase in execution time due to an increase in the number of operations when including normal cases. The execution time increases by a factor of 4.95. This increase may not have a great impact in the diagnosis system, since the training phase is done offline. Nevertheless, since a prior detection stage is needed anyway, the primary objective is to minimize the Diagnosis Error Rate and the Undetected Rate, and therefore it is recommended not to use normal cases in training.

6.3.4 Comparison between expert elicited and learned rules

Some experts were requested to manually define the rules relating the selected causes and symptoms (Table 6.5). In this experiment, the three error measurements were compared for the expert elicited rules versus the learned rules (Tables 6.6 and 6.7) using the proposed algorithms (with default parameters of each one, training without normal cases). Results are shown in Figure 6.21.

Table 6.5: Expert elicited rules

Acc.	*Ret.*	*HOSR*	*RSRP*	*RSRQ*	Cause
LOW	LOW	HIGH	HIGH	HIGH	*SW Problem*
	LOW		LOW	HIGH	*Coverage*
LOW	LOW	HIGH	HIGH	LOW	*Quality*
	LOW	LOW			*Mobility*

It can be seen that there are significant reductions in the Undetected Rate when using learned rules. On the other hand, False Positives are much greater when using learned rules, while Diagnosis Error Rate is similar. It can be concluded that the parameters used in these tests favor the reduction of the Undetected Rate. This might be an advantage depending on the

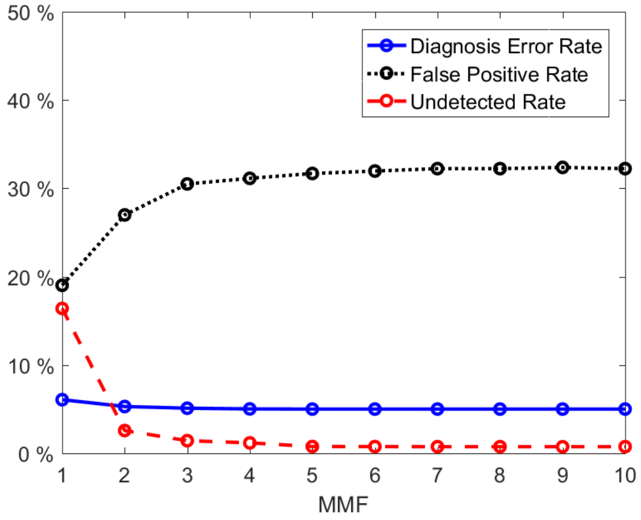


Figure 6.14: Error rates for MMF.

Table 6.6: Rules learned with the genetic algorithm

Acc.	*Ret.*	*HOSR*	*RSRP*	*RSRQ*	Cause
LOW		HIGH	HIGH	HIGH	*SW Problem*
	LOW	HIGH	LOW		*Coverage*
LOW	LOW	HIGH	HIGH	LOW	*Quality*
	LOW	LOW			*Mobility*

Table 6.7: Rules learned with the data driven algorithm

Acc.	*Ret.*	*HOSR*	*RSRP*	*RSRQ*	Cause
LOW		HIGH	HIGH	HIGH	*SW Problem*
		HIGH	LOW		*Coverage*
LOW		HIGH	HIGH	LOW	*Quality*
	LOW	LOW			*Mobility*

requirements of the final system.

6.3.5 Comparison with state-of-the-art algorithm

Bayesian Networks (BNs) have also been previously proposed for troubleshooting in mobile networks [13]. In this Section, the simulated cases used in the sensitivity tests will be used for training a BN. Figure 6.22 depicts the BN equivalent to the fuzzy system trained by the proposed DM algorithms.

The BN is designed, trained and validated in the GeNIe software package [97]. Each PI is discretized with a single threshold. This threshold will be the middle point between the high and low values previously used in the fuzzy set membership functions described in Section 3.1.4. The discretized cases are then used for supervised learning.

The results are shown in Figure 6.23. Two different tests are done, one excluding normal

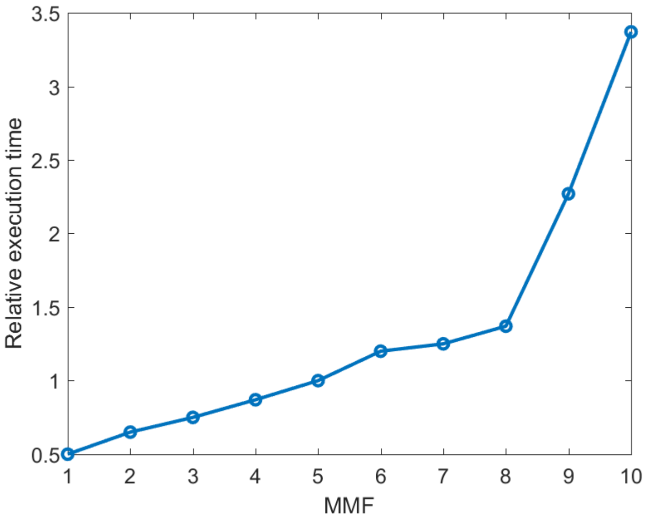
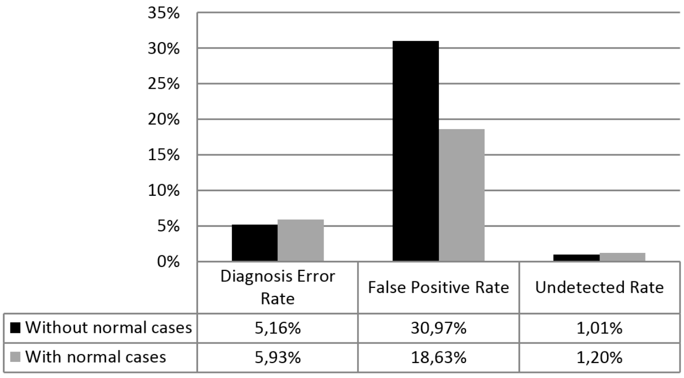


Figure 6.15: Relative execution times for different configurations of MMF.



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
■ Without normal cases	5,16%	30,97%	1,01%
▨ With normal cases	5,93%	18,63%	1,20%

Figure 6.16: Results comparing the presence and lack of normal cases in the training set for the genetic algorithm.

cases from the training set and one including them.

When normal cases are not used, the BN is not trained to recognize them. Therefore, since the BN always provides a diagnosis, it will obtain a high probability for one of the problems, producing a False Positive Rate of 1 and an Undetected Rate of 0. This effect vanishes when using normal cases in the training, so the False Positive Rate is reduced to 14.5%. On the other hand, the Undetected Rate increases from 0 to 17.4%. The Diagnosis Error Rate also increases slightly when using normal cases from 21.9% to 25.5%.

The first conclusion from these results is that, although using normal cases reduces the False Positive Rate of the BN at the cost of reducing the reliability and the accuracy, it is still too high. Therefore, a detection stage is still required.

Secondly, it can be concluded that, under the same conditions, the proposed methods produce a more accurate diagnosis system, specifically, for the data driven method a Diagnosis Error Rate of 5.1% versus 21.9% of the BN is achieved. In addition, the use of fuzzy logic has other side advantages over BN, such as producing understandable rules and simplifying the process of

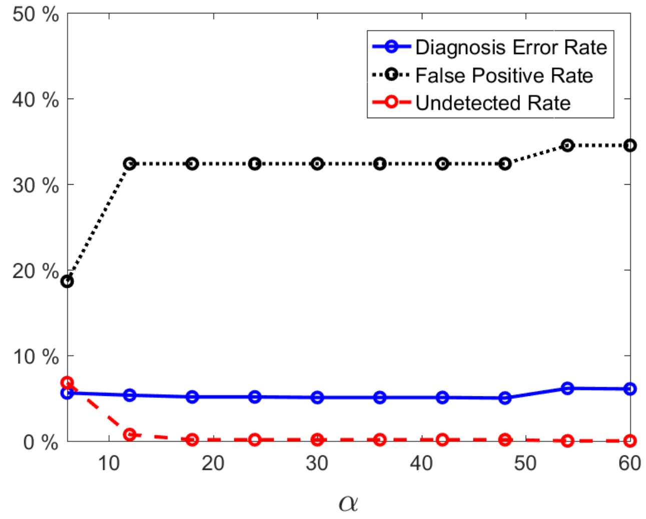


Figure 6.17: Error rates for variable α

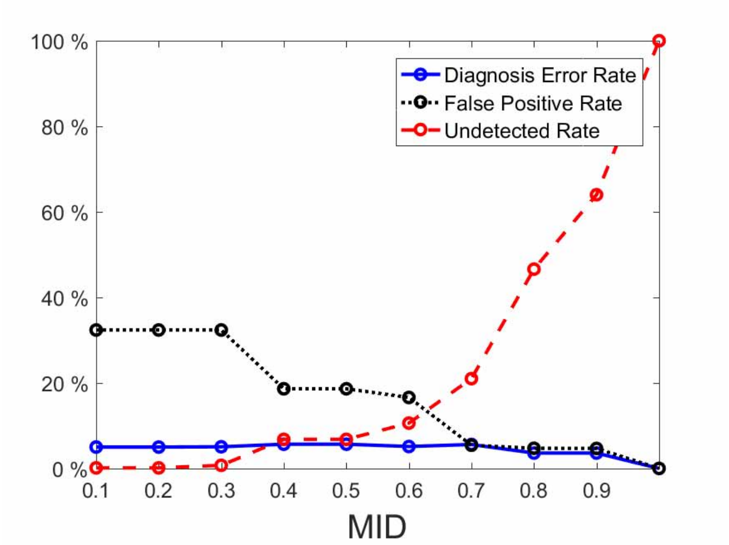


Figure 6.18: Results for variable MID

integration of learned and manually elicited rules.

6.3.6 Tests with manually selected real cases

In order to have more information when choosing among the genetic or the data driven algorithms, a test comparing them head to head on cases extracted from a real network was carried out. This test was done before the full data preparation steps were available, so the test cases were selected and processed manually.

There are 72 available cases, belonging to four possible categories (*missing neighbor*, *interference*, *high CPU usage* and *normal*), each one having 18 cases. Note that in a real network, the proportion of normal cases will be much higher. In this test, the proportions have been mod-

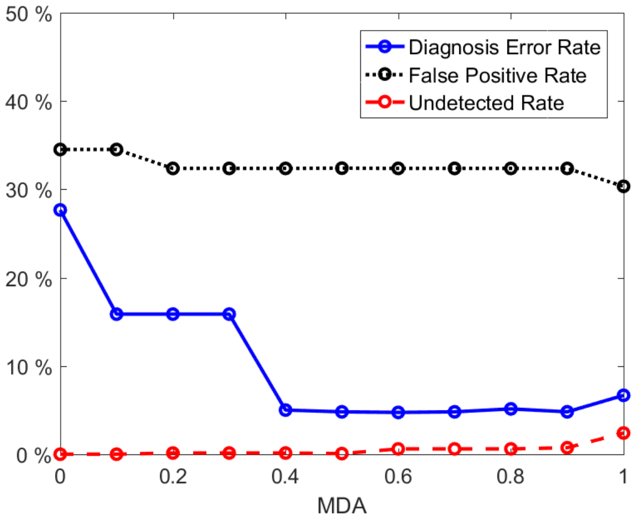


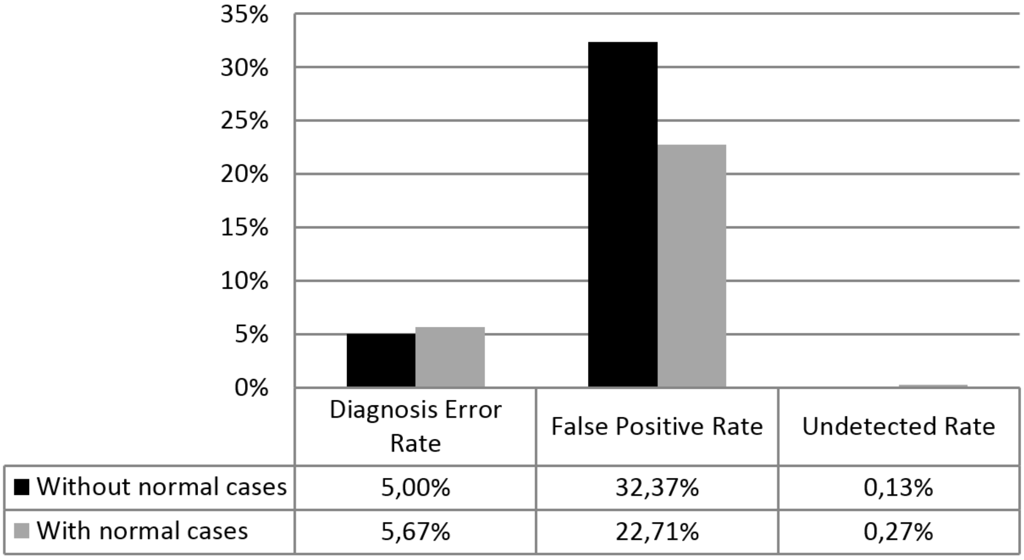
Figure 6.19: Error rates for parameter MDA

ified in order to better understand the behaviour of the DM algorithm when using problematic cases. Each case has values for 5 PIs (*accessibility*, *retainability*, *HOSR*, *Average Received Signal Strength Indicator (RSSI)* and a *CPU overload* indicator).

Due to the reduced available number of cases, a cross validation technique has been applied. The set of cases is divided in two partitions, each containing 9 random instances of each problem. One partition is used for training and the other for validating the results. Since the normal cases are not used in the training process, they are all included in the validation set. This process is repeated 100 times with different training and validation sets, and the errors are averaged.

The results for both algorithms are shown in Figure 6.24 For the genetic algorithm, using the default values for the parameters, the obtained results are very poor. This is mainly due to the small number of available cases. This has a complex effect on the evolution of the population. Since the number of rules is low, a rule with a low Success Rate has less probabilities of surviving. Therefore, bad rules die earlier. This reduces the initial population rapidly, and, therefore, the gene pool, that is, the reserve of *"not-so-good"* rules that may produce very successful rules via a small mutation. In this population bottleneck, only the best rules survive, and they may not cover all the cases. Since the best rules have a lower probability of mutating, there is a lower chance that new rules covering the gaps in the training sets appear, and therefore, the Undetected Rate increases. To fix this, it is important that the algorithm has the opportunity of exploring many possibilities. Therefore an immediate action that can be taken is increasing the MMF to increase the exploration. Also, in this situation of a small gene pool, it will help to increase the chances of exploring rules that would otherwise be ignored. Therefore, the number of initial rules is also increased. A new set of enhanced parameters is created where the number of initial rules is 20 and the MMF is 10. The results are clearly improved, although still the Undetected Rate is higher than in the tests due to the very small number of training cases.

With the data driven algorithm parameters adjusted to the default values ($\alpha = 48$, MDA=0.4, $MID = 0.1$), the Undetected Rate is 14.3%, which is better than the improved results obtained by the genetic algorithm. The Diagnosis Error Rate shows that the system is very accurate when



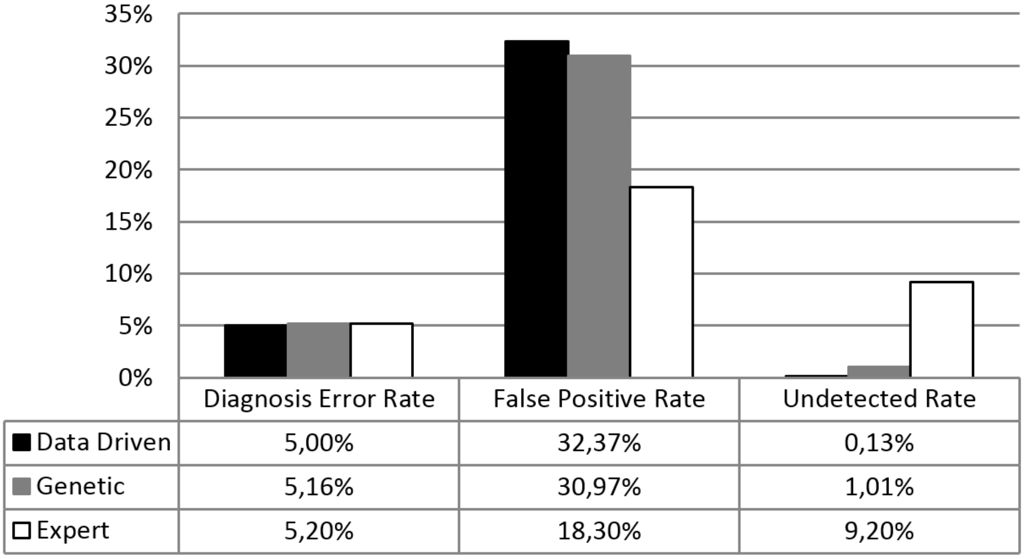
	Diagnosis Error Rate	False Positive Rate	Undetected Rate
■ Without normal cases	5,00%	32,37%	0,13%
▨ With normal cases	5,67%	22,71%	0,27%

Figure 6.20: Results comparing the presence and lack of normal cases in the training set for the data driven algorithm

a diagnosis is produced, given that the case is not a normal case. Again, the parameters can be manipulated to adapt them to the low number of training cases. Since the algorithm does not have enough information, the aggregated rules are too restrictive, that is, they impose a value on a PI that may be irrelevant. For example, *missing neighbor* cases are identified by the experts when the *retainability* and *HOSR* are low, regardless of other values in the set of PIs chosen here. In one of the executions of the algorithm, the rule obtained for diagnosing these problems is *"if (accessibility is high) and (RSSI is low) and (CPU overload is false) and (HOSR is low) then (problem is missing neighbor)"*. Therefore, a case that behaves as the experts expect, but has a low *accessibility* (for instance due to a high traffic at the same time as the *missing neighbor* problem), will not be classified as a *missing neighbor* problem, and will contribute to the Undetected Rate. The only way that the algorithm can overcome this problem is if that case (or several similar cases) is present in the training set. In this scenario, the only action that can be taken to improve the performance of the resulting FLC is to loosen the parameters so even one occurrence in the training set produces a rule with a score high enough to be part of the final RB. With this objective, the MID setting can be changed to 0, so the score does not influence the validity of a rule for its inclusion in the RB. Repeating the same experiment with the new MID settings, the Diagnosis Error Rate is still 0, the Undetected Rate is slightly reduced to 10% and the False Positive Rate remains 0.

These results show that the data driven algorithm can perform better also on real cases an in scenarios with small training sets. This adds up to the slightly better results observed on the sensitivity tests.

As previously pointed out, the reasons behind creating a new DM algorithm were varied. The main reason was to create a parallelizable algorithm that could be used in Big Data systems. Other resons were the long execution times of the genetic algorithm and the randomness of the results (that could be overcome with a high number of loops, i.e. a higher execution time). After



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
■ Data Driven	5,00%	32,37%	0,13%
▨ Genetic	5,16%	30,97%	1,01%
□ Expert	5,20%	18,30%	9,20%

Figure 6.21: Results comparing expert elicited and learned rules.

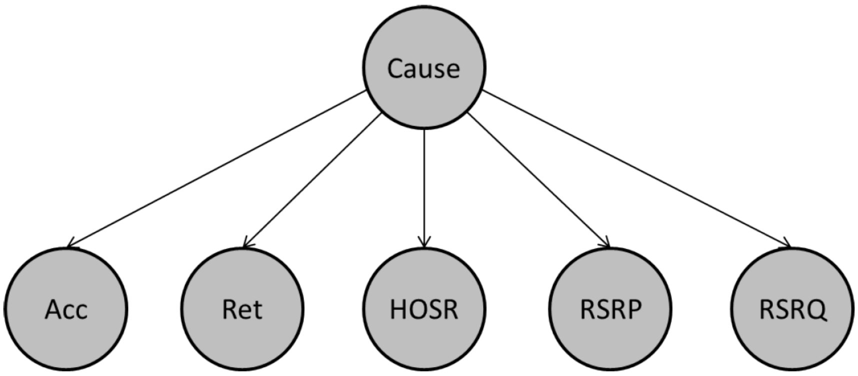


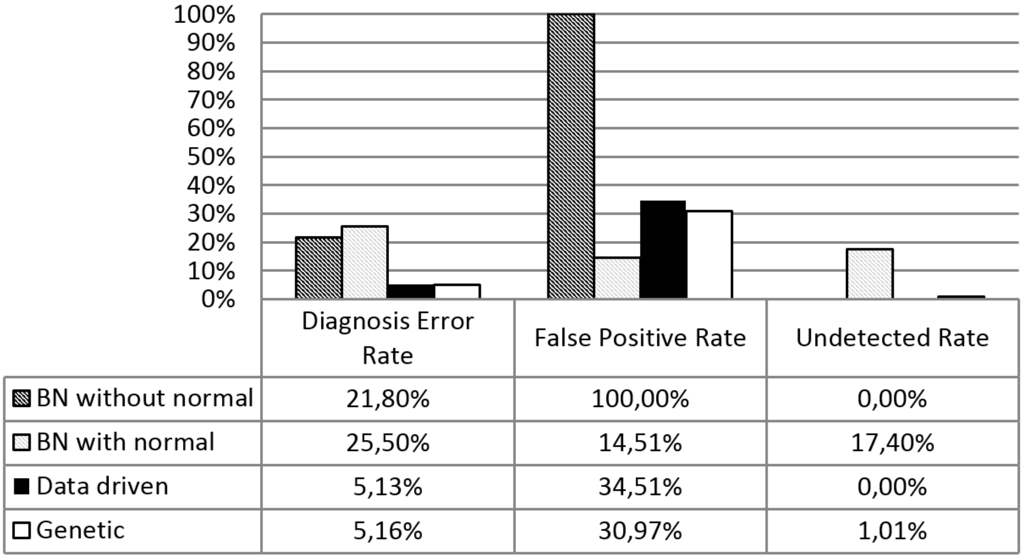
Figure 6.22: BN used for troubleshooting.

the tests, the decision of creating the data driven algorithm is further justified. Given these advantages, the data driven algorithm was chosen over the genetic algorithm for its inclusion in the full KDD process.

6.4 Inclusion in KDD process

In this Section, the designed algorithm will be integrated in the overall KDD system. In Chapter 5 a model of a real fault database was created in order to better understand the behaviour of the PIs conditioned to the occurrence of each problem. In this Section, the model will be used to create a training and a testing set in order to extract an FLC for diagnosis. This FLC will then be tested again over the original cases for validation.

As previously stated, FLCs have two sets of parameters that must be extracted in KDD



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
▨ BN without normal	21,80%	100,00%	0,00%
☐ BN with normal	25,50%	14,51%	17,40%
■ Data driven	5,13%	34,51%	0,00%
☐ Genetic	5,16%	30,97%	1,01%

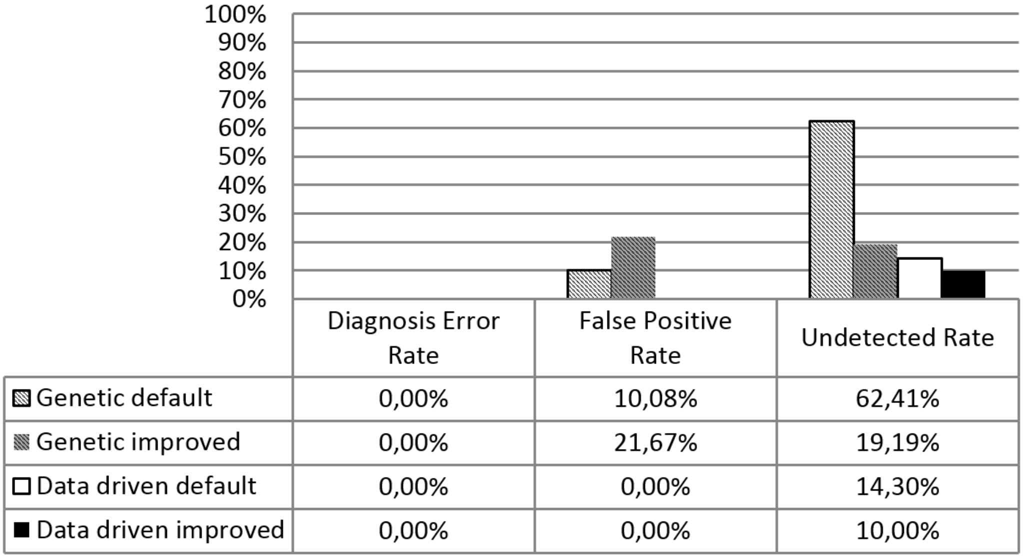
Figure 6.23: Results for BNs trained with supervised learning compared to FLCs trained with the proposed methods.

process: the Data Base (DB) and the Rule Base (RB). In Section 6.2 the data driven algorithm used for the extraction of the RB was described. The tests were all performed with a previously known DB. Nevertheless, in the final system, the DB must be extracted from the data. For this purpose, two parameters (x_{LOW} and x_{HIGH}) are required in order to define the *low* ($\mu_{LOW}(x)$) and *high* ($\mu_{HIGH}(x)$) fuzzy sets for each PI. To obtain these parameters, an algorithm based on Entropy Minimization Discretization (EMD) [98] is used. The EMD algorithm returns one threshold that divides the domain of a PI in two intervals. Since two thresholds are required, in this study, the EMD algorithm is used twice; first, to divide the domain in two intervals, and a second time for each interval to obtain a threshold to divide it into two sub-intervals. The thresholds obtained for each sub-interval are used as the *low* and *high* thresholds.

Figure 6.25 shows the overall diagram of the DM algorithms integrated in the full KDD system. The output of the DB learning algorithm is used in the RB learning algorithm along with the training set.

The DM stage is then used on a training set generated by a problem emulator using the model extracted from the fault database. The resulting KB (a rule set) will then be used in an FLC to diagnose a testing set generated with the same model and find the error rates. The FLC will also be used on the original set to test how the rules that were obtained from the model behave with its original source. Figure 6.26 depicts this process. Specifically, 5000 training vectors will be created, 1250 for each type of problem. The reason behind using a model instead of the original cases is twofold; as shown in Section 5.1.2, the number of original cases is low (359 training vectors) and the distribution among the different classes is unbalanced (i.e. there are great differences in the number of vectors in each class).

With these considerations, Table 6.8 shows the fuzzy sets extracted by the DB learning algorithm. Using the parameters described in Table 6.10, the data driven algorithm is used over the generated training set, resulting in the rules described in Table 6.9.



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
▧ Genetic default	0,00%	10,08%	62,41%
▩ Genetic improved	0,00%	21,67%	19,19%
☐ Data driven default	0,00%	0,00%	14,30%
■ Data driven improved	0,00%	0,00%	10,00%

Figure 6.24: Results for both algorithms using manually processed real cases

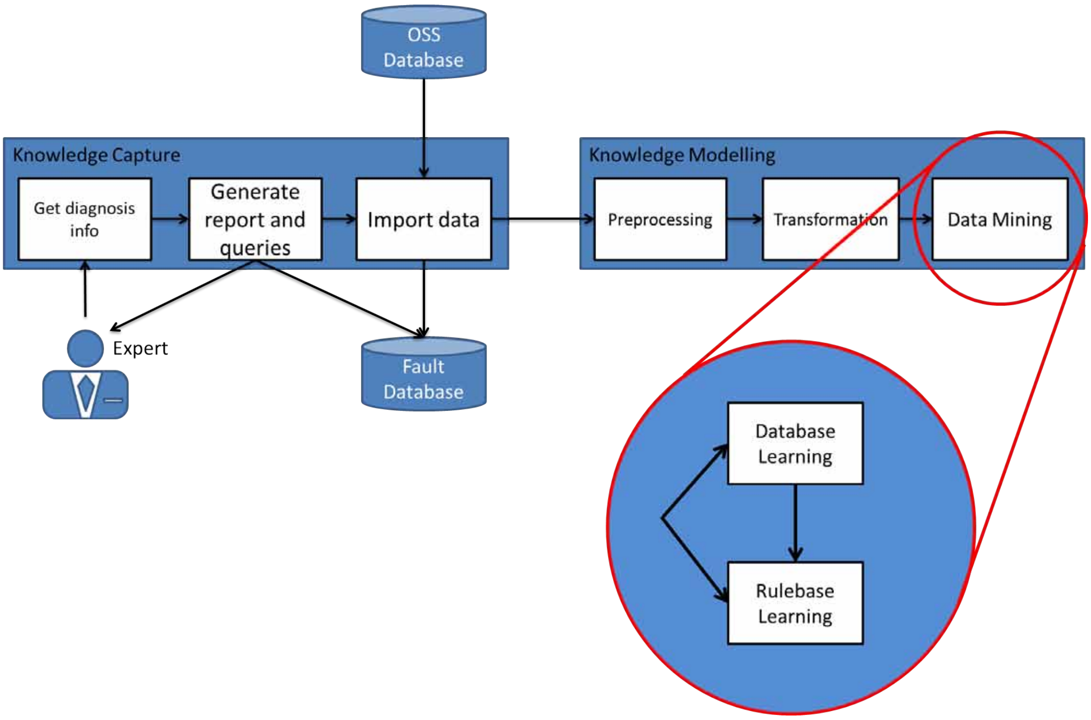


Figure 6.25: Diagram of the DM process in the overall KDD system.

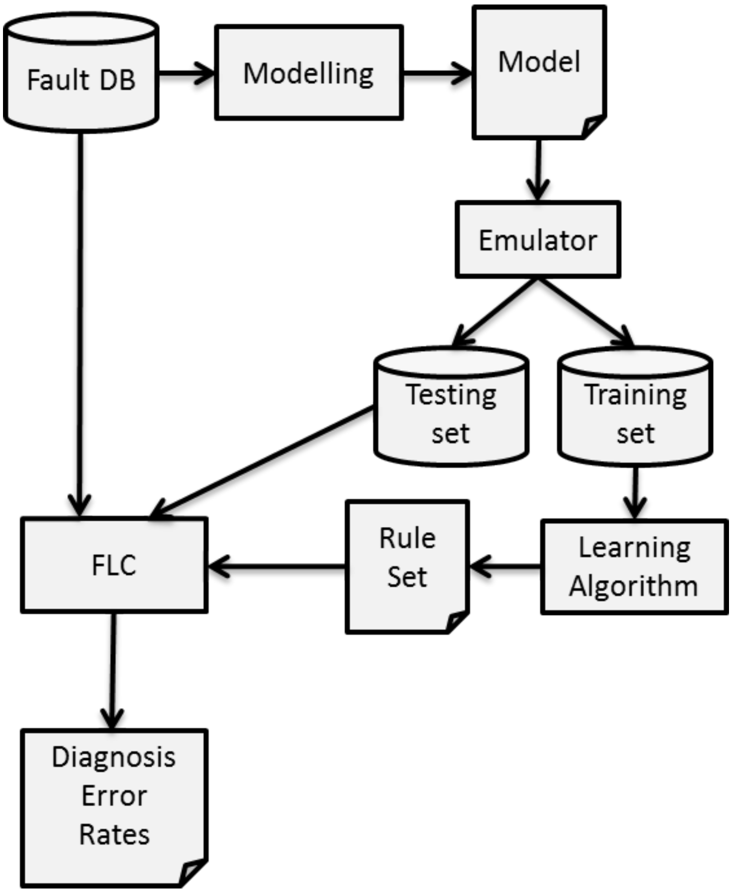


Figure 6.26: Experimental procedure

Table 6.8: Fuzzy sets

PI	x_{LOW}	x_{HIGH}
Average CQI	5.99	10.78
Average Number of Active UEs	0.01	0.7
Average RSSI	-118.8	-109.14
CS Fallback Rate	0.04	4.03
Handover Success Rate	40.69	99.9
Interfreq HO Preaparation Rate	0.17	72.22
Intrafreq HO Preaparation Rate	43.18	100.0
iRAT rate	0.0	2.05
Number of Bad Coverage Reports	1.75	367.0
Number of CPU Overload Alarms	0.0	6.63
Number of ERAB Attempts	20.5	8480.83
Traffic Volume (DL)	0.05	1.14
Traffic Volume (UL)	0.01	0.16

Table 6.9: Rules learned from the generated cases

CQI	Num. UE	RSSI	CS Fallback	HOSR	Interfreq. HO	Intrafreq. HO	iRAT	Bad. Cov	CPU Load	Num. ERAB Att.	Traffic DL	Traffic UL	Cause
	H			H		H				H	H	H	High Traffic
	L							L	L	L	L	L	No Traffic
				H		H			H				CPU Overload
				H		H	H	H	L				Low Coverage

Table 6.10: Learning algorithm parameters

α	MDA	MID
50	0.3	0.1

The obtained fuzzy sets and rules are then included in an FLC that is used for diagnosis over the testing set, measuring the errors described in Section 3.1.3. The results are shown in Table 6.11.

To further understand the origin of the errors, the diagnosis results will be represented in a confusion matrix (Table 6.12). The confusion matrix represents the accuracy of a classifier broken down into each class, and enabling the distinction on where most of the errors occur. In the case of the diagnosis system, the confusion matrix represents for each real root cause (row), the distribution of diagnosis given by the system. For instance, In Table 6.12, the *High Traffic* root cause (second row), is diagnosed 87% of the time correctly, 12% as *CPU Overload* and 1% as *Low Coverage* (which sums up a total Diagnosis Error Rate of 13% for *High Traffic* cases). There is no confusion of *High Traffic* cases with *No Traffic* and there are no cases where the diagnosis system does not detect a *High Traffic* problem (Undetected Rate is 0%). The False Positive Rate can be calculated by inspecting the *Normal* row; since only 1% of *Normal* cases are marked as *Normal*, the False Positive Rate is 99%. This problem with the normal cases can be also observed in Table 6.11. Nevertheless, as already seen in the sensitivity tests, a detection phase is normally required, which will filter out the normal cases. For example, assuming a detection phase with an 80% accuracy (which is not very high), the False Positive Rate will be 19.74%.

It can also be observed that *CPU Overload* is the most accurately diagnosed problem, whereas *Low Coverage* is only diagnosed correctly 74% of the time. Most of the errors for this class are confused with *High Traffic* (14 %). This is due to the fact that in the original set of cases, many problems marked as *Low Coverage* were due to border sectors, which usually also have a high traffic. Table 6.13 shows the confusion matrix over the original data set. In Table 6.11 it can be observed that the results are similar to those obtained on the modelled testing set.

Table 6.11: Results of the learning tests

Set	Diagnosis Error Rate	Undetected Rate	False Positive Rate
Testing set	3.64 %	9.48 %	98.72 %
Original DB	0 %	3.48 %	100 %

Table 6.12: Confusion matrix

	Normal	High Traffic	No Traffic	CPU Overload	Low Coverage
Normal	0.01	0.33	0.23	0.01	0.41
High Traffic	0.0	0.87	0.0	0.12	0.01
No Traffic	0.13	0.0	0.87	0.0	0.0
CPU Overload	0.0	0.0	0.0	1.0	0.0
Low Coverage	0.02	0.14	0.01	0.1	0.74

Table 6.13: Confusion matrix over the original cases

	Normal	High Traffic	No Traffic	CPU Overload	Low Coverage
Normal	0.0	0.41	0.47	0.01	0.11
High Traffic	0.0	0.88	0.0	0.09	0.03
No Traffic	0.0	0.0	1.0	0.0	0.0
CPU Overload	0.0	0.0	0.0	1.0	0.0
Low Coverage	0.0	0.18	0.09	0.09	0.64

6.5 Conclusions

The rules obtained in Section 6.4 are the final target of the overall KDD system. These rules have been extracted from a set of cases accumulated with only a minor human intervention: indicating the sector where the problem occurred, the day when it happened and a label identifying the problem. In this Chapter the final and most important step, DM, has been described. It is in this stage where the expert knowledge is effectively extracted and transformed into an FLC that imitates a troubleshooting expert. Nevertheless, this stage would not have been possible without the intervention of all the previous KDD steps. The full design process for the DM stage has been described and its behaviour has been tested first on simulated and controlled cases, and then on the collected fault database.

Chapter 7

Conclusions

In this Chapter, the conclusions of this thesis will be reviewed. The results, contributions and future work will be summarized.

For each contribution, its role in the thesis are described and the results of its tests are shown. Design decisions on the overall process are analyzed based on the results of each part of the thesis.

7.1 Results

In this Section, the results obtained in this thesis are reviewed. Specifically, the results obtained in the network under study are analyzed. These results can be easily obtained for any other network by using all the stages from the beginning (data collection) to the end (data mining). Therefore, although the results may not always reflect the behaviour of all networks, understanding them in the context of a known scenario is a very valuable resource for future deployments. In addittion, the proposed data analytics methodology is applicable to all networks.

7.1.1 LTE fault database

The LTE fault database collected in Chapter 5 is the first step of the data analytics process. It is the output of the selection phase (described in Chapter 4), therefore, it is the first subproduct that contains expert knowledge.

The fault database must meet some requirements, such as having enough coverage of each problem (both in time and selected PIs), variety in the types of collected problems and accuracy in the diagnosis. Some of these requirements depend on the quality of the work of the experts, whereas others depend on the design of the system. For instance, in order to have a good coverage in time of each problem, the data collection process was designed to collect data for two weeks prior to the diagnosis. The selection of this number was done as a compromise between the advantages of using a short period (less data volume, lower probabilities of covering two independent problems under the same label) and a long period (covering possible hidden trends that could predict the occurrence of a problem even before experts would notice it). Two weeks was considered a good trade-off based on experience, since it covered the detection, diagnosis and recovery stages (more or less one week for long standing problems) plus an additional previous margin.

The collection system software platform was designed with a set of criteria oriented towards minimizing the impact on experts workflow.

The database extracted with this method covered a period of 10 months (from June 2013 to April 2014). In this time interval, a total of 475 cases were reported and collected in the database. A total of 21 types of problems were covered.

The collection system went through several redesign stages to improve in each step both the ease of use and the quality of the collected data. As a consequence, three versions of the PI list were produced, so cases collected at different times had different levels of detail.

Some difficulties were found on the collection stage that had to be solved before moving on to data analytics. Firstly, a high imbalance among the number of instances of each problem was observed, since some problems were much more common than others. Secondly, for some problems, the false positives were very high, especially in the cases collected by a manually coded rule engine used as an early prototype for automatic diagnosis. Thirdly, some of the diagnosis inserted in the system were changed later as new evidence (either in the same case or in a similar one) was found, creating some invalid entries. Finally, there was some reticence from the troubleshooting experts for using the system, especially in the earlier versions, that were more complicated to use. The solution for this problem was to integrate the system into the tool that they used for inspecting problems, but this feature was still not implemented at the time of this writing.

This factor drove to the decision of using only a reduced subset of the collected database for further analysis. To select this subset, another followed criterion was that the class must have been collected at a time where enough PIs where considered, and all classes must have the same PIs. This drove to a reduced database of 47 cases divided among 4 problem types for the data mining process.

7.1.2 Data preparation for LTE faults

Although the core process for extracting knowledge is DM, data preparation is an indispensable step. Without data preparation, the patterns that the DM process searches for would be hidden among noise. In the case of the LTE fault database, data preparation plays the role of cleaning the data (that is, filling missing values), reducing the dimensionality and translating to a viable format (by eliminating the time dependence).

The result of the data preparation stage must be a training set for the DM composed of individual vectors containing PI values for a degraded time interval and a label indicating the diagnosis of the problem causing the degradation. There must be one vector per degradation, although each problem in the database usually registers more than one degradation.

The data preparation process used in this thesis has two main stages: data cleaning (filling missing values) and data reduction (finding the degraded intervals and averaging the PIs for said intervals).

The data cleaning stage was tested by artificially creating missing data, showing that the error introduced with respect to the original data was lower than other state-of-the-art methods, and that it increased for growing time intervals of missing data.

The data reduction algorithm used a state machine to determine degraded time intervals. Each time interval was translated into a single instance of a problem. Due to the nature of the data reduction algorithm, it was applied after the data cleaning stage. The result was a database of 359 vectors.

The major difficulties found in this stage were mainly due to missing data. Some periods of missing data were far too long to be filled with the algorithm without significantly increasing the error. Moreover, these periods were frequently present in the data near the occurrence of problems, since the affected sectors were subject to inspection, reconfigurations, restarts and to be shut down. These periods were easy to detect and discard (although they caused loss of information). In cases where the eNodeBs were still on, but had the PI collection subsystem disabled, the reported PI values were adjusted to a constant value (normally 0). This was a big issue, since these values were still valid and therefore could not be considered missing values. The solution for this problem was to mark long intervals where the KPIs were 0 as missing values, although this could cause some errors in extreme cases.

7.1.3 LTE fault modelling

The next step in the study of the fault database was the creation of a model of the relations between the selected problems and a subset of the PIs selected by the experts. Such a process is in fact a DM stage. Although this step is not part of the overall KA system, since it does not directly create a ruleset out of the input database, in this thesis it has been used to improve the results due to the low number of problem occurrences and the imbalance among root causes.

The result of the modelling is a set of PDFs for each PI conditioned to the occurrence of each problem. With this objective, the vectors extracted from the data preparation stage were analyzed. For each pair of PI/problem, a set of predefined PDFs was adjusted and the PDF that minimized the error (using the D-Value of the K-S test) was chosen as the model for that pair. The resulting model was shown in Appendix C and explained in Section 5.3. It was subsequently used for generating a large emulated database with equal number of samples for each problem. This database is a better input for the DM algorithm used for rule extraction, since the number of samples is larger and the problems are balanced (i.e. there are equal samples for each class).

The difficulties described in earlier stages (namely, wrong diagnosis and wrong values caused by missing data) had an observable effect in this stage. Once these problems were detected and studied, new procedures were added in the appropriate stage and the whole process was repeated.

7.1.4 Datamining of LTE troubleshooting rules

The DM stage is the main part of the KA algorithm. With the cleaned and reduced data, the DM stage extracts an FLC that will be used for diagnosis.

Since FLCs have two parts (DB containing the fuzzy sets and RB containing the fuzzy rules), two DM algorithms were used over the training data. The resulting FLC contains the boundaries and rules adapted to the target network and based on the knowledge of the troubleshooting experts. First, the DB was extracted using an algorithm that maximized the entropy of the sets defined by the boundaries. The extracted fuzzy sets were then used along with the training set to extract the RB using a specifically designed algorithm. Two RB supervised learning algorithms were designed and tested with another emulated dataset. The first developed algorithm (a genetic algorithm) was discarded because of execution speed issues and its inability to scale up for Big Data scenarios. The second algorithm (data driven) was chosen and used on the cases from the network under study.

The results of the DM algorithm were then tested over the original set of vectors, showing a high success rate.

In this stage, the unavailability of real cases was a major difficulty in the beginning. Due to this reason, a model based on experience was developed and used for training and validation. This situation was later compensated by using the cases in the collected database, showing the behaviour of the system from data collection to generation of diagnosis rules.

7.2 Contributions

In this Section, the contributions of this thesis will be reviewed. The overall contribution of the thesis was the development of a methodology for KA in LTE troubleshooting (**Objective 5**) based on data analytics. The product of this methodology is an automatic diagnosis system based on fuzzy logic. This system will save experts time when dealing with repetitive issues, and has the capacity to improve as new problems arise and become common. Therefore, experts usually will be able to focus on new and more challenging problems. Also, the problems that are diagnosed by the automatic system can be solved earlier, driving to reduced downtimes and therefore a better user experience.

This thesis also has individual subproducts in each stage of the processing pipeline.

7.2.1 LTE fault database

Another contribution of this thesis was the LTE fault database (**Objective 1**) that contains real troubleshooting cases as opposed to data from simulations, which most studies use. This database is a tool that can be used for many applications related to troubleshooting. The range of applications includes training of new experts, planning for improvement and optimization works, analysis of the progress of a network over time, etc. Another scenario where a fault database can be (and has been) used is in the development of SON functions. In this thesis, specifically, a Self-Healing system has been developed with the fault database. A fault database also reflects the state of the network and how optimization efforts have affected it over time.

The lack of such databases is often a problem and the knowledge of this fact may cause surprise at first sight. The reason for this shortage is that experts are usually focused on troubleshooting problems as soon as possible and do not have time or incentives to create a solid database on their work. On the other hand, developers are usually focused on creating products based on requirements that usually do not take real data into account (because there are no comprehensive databases with solved cases). A middleground between troubleshooting experts and tool developers is required where the creation of a database is done with the knowledge of troubleshooting experts but minimizing the effort required from them. This thesis has proposed a method and a software platform based on data analytics for achieving this middleground. The requirements for these types of platforms have been collected using feedback from experts that used the tool in its early development stage.

7.2.2 Data preparation for LTE faults

In data analytics, one of the main tasks is data preparation. This task is highly dependent on the nature of the data and the targeted DM method (which would not work without a data preparation stage). This thesis studied the properties of troubleshooting data and the requirements of automated troubleshooting systems (**Objective 2**) and proposed data preparation methods specifically prepared for this purpose. This thesis has also studied the Big Data aspects of troubleshooting data, justifying why traditional data processing algorithms do not provide the required performance. Firstly, in this thesis, a method for data cleaning that uses features of mobile network data has been proposed. This method uses the periodicity of the data to impute missing values. Secondly, an algorithm has been proposed for data reduction. This algorithm exploits the concept of degraded time intervals, that is, time intervals where an underlying problem visibily affects the performance of a network. This method delimits degradations in time, allowing the creation of time independent vectors of PIs for the DM algorithm. This methodology is also useful for marking degradations on time series for other uses, such as visualizations, training of troubleshooting personnel, etc.

7.2.3 LTE fault modelling

This thesis has proposed a method to create an LTE fault database model (**Objective 3**). This model studies the relation between problems and PIs in a real scenario, as opposed to other studies where these relations are studied on simulations. The model can be used for creating synthetic fault databases in order to train AI algorithms. In fact, this method was used to train a DM algorithm with good results over the original dataset, as shown in Chapter 6. The model can also be used for better understanding each problem or each PI on its own, as well as the relations among problems and PIs for the network where the analysis was done; and this can lead to better decisions on improving the network. The proposed methodology can be extended to other networks, with different sets of problems and PIs, to automatically extract models that describe their behaviours.

7.2.4 Datamining of LTE troubleshooting rules

Other contributions of this thesis were the study of the application of FLCs to diagnosis (**Objective 4**) and the design and implementation of two data mining methods to adjust an FLC for network troubleshooting based on a fault database (**Objective 5**). In the data mining process, the Big Data nature of the troubleshooting data was taken into account in one of these algorithms (the data driven method) by making it parallelizable. Although the FLC that was extracted was particularized for the target network, the DM method can be applied on any network (with a previous data preparation stage) to obtain a fine tuned automatic troubleshooting system. The only requirement is that experts of the troubleshooting team of these networks provide the required information when problems occur. Each new addition of a problem will then improve the accuracy of the system.

The rules extracted from different scenarios can also be collected in a centralized location in order to have a large knowledge base on LTE fault situations. Since FLCs have an RB and a DB, rules learned in one network can be applied in other networks using their adapted fuzzy sets. Also, expert knowledge extracted by other means (such as interviews) can be mixed with rules extracted by the DM algorithm.

Finally, the automatically extracted rules and fuzzy sets also have an informative value on their own, and can be used to better understand the causes of problems and the distributions of the PI values.

7.2.5 Knowledge Acquisition for LTE Self-healing

This thesis has studied closely the problem of KA, reviewing different approaches and determining the advantages of using data analytics. Specifically, KDD using troubleshooting information contained in fault databases is proposed.

The requirements for a KA platform have been enumerated, differentiating between requirements from the technical perspective concerning the fault data and performance monitoring subsystem; and the human perspective concerning the experts and taking into account the difficulties of engaging them in a time consuming process.

With the studied requirements, a KA platform was designed (**Objective 5**), developed and used to collect the fault database (**Objective 1**). The design process had into account the functionalities of the data preparation stages, so the information required from the expert was reduced to only three parameters: date, sector and diagnosis. This would make the inclusion of the KA process into the workflow of experts much more simple, since it could be integrated into the software that they already use to monitor network performance.

7.2.6 Big Data aspects of Self-healing

The use of data analytics in Self-healing is subject to the characteristics of the underlying fault data. This thesis has studied the properties of this data (**Objective 2**), determining that, due to their volume, variability and velocity of generation, they can be considered as Big Data compliant. The Big Data nature of the Self-healing problem has therefore been further studied (**Objective 6**).

Big Data aspects have been taken into consideration throughout the thesis, designing all the algorithms ensuring that they can be executed in a parallel manner without loss of accuracy.

Additionally, to further illustrate the Big Data principles in Self-healing, several published algorithms have been redesigned in order to run as parallelizable processes, therefore enabling the use of cloud computing to increase their performance.

7.3 Future work

In this Section, the lines of research that could continue the work of this thesis are described. Although the concept of KA using data analytics described in this thesis has been proven feasible, and its usefulness has been demonstrated and measured, there is still a long way to go until a marketable solution arises. In order to achieve this objective, several points must be addressed in the future:

- Improve the KA interface, providing them with more incentives for usage. Specifically, a point that was proposed was integrating the DI detection and processing in the KA interface, so when experts introduced the information on a problem, the DIs were shown to them with statistical information. Also, similar DIs on past cases could be shown to them.
- Further collection of problems, with a higher variety of networks and kinds of problems. This will help into creating more comprehensive models and more complete diagnosis systems.
- Automatic selection of relevant PIs. In this study, the PIs that were used in the DM stage were manually selected by experts. A very important future development will be to find a way of automatically deciding which PIs are good predictors for the faults.
- Development of online learning that can improve the quality of the output as new inputs are received without the need of executing the full DM process.
- More tests and improvements on the DM algorithm for the DB (the fuzzy sets).
- More studies and optimizations on the performance of the algorithms, specially in the Big Data aspect (i.e. implementation as parallel algorithms and study of its performance).

7.4 Publications and Projects

7.4.1 Journals

	Publications arising from this thesis	IF	Journal Rank
I	E. J. Khatib, R. Barco, P. Muñoz, I. de-la-Bandera, I. Serrano, "Self-healing in mobile networks with big data". IEEE Communications Magazine, vol. 54, no. 1, pp. 114-120, Jan. 2016.	5.125	Q1 (2/82) Telecommunications
II	E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, I. Serrano, "Data mining for fuzzy diagnosis systems in LTE networks". Expert Systems with Applications, vol. 42, no. 21, pp 7549-7559, Nov. 2015.	2.981	Q1 (19/130) Computer Science, Artificial Intelligence

III	E. J. Khatib, R. Barco, A. Gómez-Andrades, I. Serrano, "Diagnosis based on genetic fuzzy algorithms for LTE Self-Healing". IEEE Transactions on Vehicular Technology, vol. 65, no. 3, Mar. 2016.	2.243	Q1 (14/82) Telecommunications
IV	E. J. Khatib, A. Gómez-Andrades, I. Serrano, R. Barco, "Modelling LTE solved troubleshooting cases", *Journal of Network and Systems Management*, Under review.	1.078	Q3 (77/143) Computer Science, Information Systems
V	E. J. Khatib, R. Barco, P. Muñoz, I. Serrano, "Knowledge Acquisition for Fault Management in LTE Networks", *Wireless Personal Communications*, Under review.	0.701	Q4 (63/82) Telecommunications
VI	E. J. Khatib, R. Barco, I. Serrano, "Degradation Detection Algorithm for LTE Root Cause Analysis", *Wireless Personal Communications*, Under review.	0.701	Q4 (63/82) Telecommunications

	Publications related to this thesis	IF	Journal Rank
VII	A. Gómez-Andrades, P. Muñoz, E. J. Khatib, I. de-la-Bandera, I. Serrano, R. Barco, "Methodology for the Design and Evaluation of Self-Healing LTE Networks", *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6468-6486, Aug. 2016.	2.243	Q1 (14/82) Telecommunications
VIII	P. Muñoz, I. de-la-Bandera, E. J. Khatib, A. Gómez-Andrades, I. Serrano, R. Barco, "Root Cause Analysis based on Temporal Analysis of Metrics toward Self-Organizing 5G Networks", *IEEE Transactions on Vehicular Technology*, Online 2016.	2.243	Q1 (14/82) Telecommunications
IX	P. Muñoz, R. Barco, E. Cruz, A. Gómez-Andrades, E. J. Khatib, N. Faour, "A method for identifying faulty cells using a classification tree-based UE diagnosis in LTE", *EURASIP Journal on Wireless Communications and Networking*, Under review.	0.627	Q3 (191/255) Engineering, Electrical & Electronic

7.4.2 Patents

	Patents related to this thesis
X	P. Muñoz, R. Barco, I. Serrano, I. de-la-Bandera, E. J. Khatib. "Fault diagnosis in Networks". Nº of application: PCT/EP2015/058924 (24 April 2015). Nº of international publication: WO/2016/169616 (27 October 2016)

7.4.3 Conferences

	Publications arising from this thesis
XI	E. J. Khatib, R. Barco, I. Serrano, P. Muñoz, "LTE performance data reduction for knowledge acquisition". GLOBECOM 2014, Austin.
XII	E. J. Khatib, R. Barco, I. Serrano, "Captura del conocimiento para el modelado de fallos en redes LTE". XXIV Simposium nacional de la Unión Científica Internacional de Radio, Valencia 2014.
XIII	E. J. Khatib, R. Barco, A. Gómez-Andrades, "Diagnosis en LTE Self-Optimizing Networks basada en algoritmos genéticos". XXIII Simposium nacional de la Unión Científica Internacional de Radio, Santiago de Compostela 2013.

7.4.4 Related projects

This thesis has been partially funded by the following projects:

- Optimi-Ericsson, ref. 59288, Junta de Andalucía (Agencia IDEA, Consejería de Ciencia, Innovación y Empresa) and ERDF.
- Proyecto de Investigación de Excelencia P12-TIC-2905, Junta de Andalucía.

7.4.5 Research Stays

Additionaly, this thesis involved three stays abroad:

- Ericsson and AT&T centers in Los Angeles, California (USA), collaborating with troubleshooting experts on their daily work to observe the process of fault diagnosis, the tools and the requirements for the Knowledge Acquisition platform.
- Ericsson center in Plano, Texas (USA), collaborating with troubleshooting experts on the development of automatic troubleshooting systems based on diagnosis rules.
- Nokia center in Aalborg (Denmark), collaborating with research engineers on Machine Type Communications and 5G technologies.

Appendix A

Description of modelled PIs

This Appendix collects the descriptions of the PIs modelled in Chapter 5. These PIs are also used in the data mining algorithm tests in Section 6.4.

PI	Range	Description
Average CQI	[1, 15]	Average Channel Quality Indicator reported by the user terminals to the eNodeB during the ROP. It reflects the interference present in the downlink channel. CQI reports are usually periodic, but they can also be requested by the eNodeB. CQI values determine the modulation scheme to be used in the communication between the UE and the eNodeB; therefore, there is a direct relation between the CQI and the throughput available to the UEs.
Average Number of Active UEs	$[0,\infty)$	Average number of active users in the Downlink in the ROP. In practice, the number of users is limited by the network equipment. Originally, the LTE design objectives were to be able to serve at least 200 users with a 5 MHz spectrum allocation. A growing number of users will reduce the amount of available resources for each user.

Average RSSI	$(-\infty, +\infty)$ dBm	Average RSSI (Received Signal Strength Indicator) in the Uplink. It reflects the total signal level received in the eNodeB. The power measured by the RSSI includes both the signal and noise terms. A high RSSI is usually an indication that there are interference sources from either inside the LTE newtork (neighboring nodes from the same or other operators) or from outside (other radio systems). A high RSSI may also be the consequence of a large number of UEs, indicating a high traffic.
CS Fallback Rate	[0,100]%	Proportion of calls transferred to 2G/3G (Circuit Switched fallback). Some calls are transferred to 2G or 3G for special needs of the call (for instance, lack of support of Voice Over LTE in the network). A high CS fallback rate indicates that many users are attempting to use a service that is only supported by Circuit Switched networks. The solution is to deploy services such as Voice Over LTE.
Handover Success Rate	[0,100]%	Proportion of handovers to neighboring or cosite cells that are correctly finalized. Low HOSR values are indication of incorrect mobility configurations (e.g. bad neighbor relations) or of problems on the target cell (e.g. congestion causing the rejection of new connections).
Interfreq HO Preaparation Rate	[0,100]%	Proportion of handovers that are done between cells in different carriers (frequencies). New carriers are deployed on sites where there is normally a large number of users in order to increase capacity.
Intrafreq HO Preaparation Rate	[0,100]%	Proportion of handovers that are done between cells in the same carrier.
iRAT rate	[0,100]%	Proportion of handovers that are done between cells of different technologies. This can be done for offloading traffic from the LTE network in case of congestion or in cases where other technologies (such as 2G/3G) offer a better quality of service.

Number of Bad Coverage Reports	$[0,\infty)$	Number of Bad Coverage reports collected from the users of the cell. Bad coverage reports are sent when an A2 event occurs; indicating that the received power of the pilot signals is below a certain threshold.
Number of CPU Overload Alarms	$[0,\infty)$	Number of alarms that indicate a high activity in the CPU of the eNodeB. Two thresholds are configured in the eNodeB for the CPU load. When the lower threshold is reached, the alarm is registered and new connections are rejected in order to prevent CPU overload. If the second (higher) threshold is reached, already established connections are dropped to reduce load.
Number of ERAB Attempts	$[0,\infty)$	Number of ERAB Connection Requests. It reflects the offered traffic and determines the behaviour of the eNodeB and other magnitudes such as RSSI.
Retainability	[0,100]%	Proportion of calls that end correctly (inverse of the dropped call rate). Retainability is one of the most important PIs (i.e. it is a KPI) because it is normally checked first to detect if there is a problem with the eNodeB. This KPI is used to separate problems from normal cases in this dataset, but not for modelling. Therefore, this KPI has not been modelled.
Traffic Volume (DL)	$[0,\infty)$ GB	Quantity (GB) of transferred data in the Downlink.
Traffic Volume (UL)	$[0,\infty)$ GB	Quantity (GB) of transferred data in the Uplink.

Appendix B

DESCRIPTION OF COLLECTED PROBLEMS

This Appendix collects the descriptions of the problems collected during the data collection stage between November 2013 and April 2014. The labels marked with an asterisk come from the manually-coded rule based diagnosis system executed in June 2013 as described in Section 5.1.2.

- Congestion: the number of users is too high for the eNodeB. There are not enough resources for guaranteeing the service to all users. The root cause of this problem depends on its degree of recurrence. If it occurs only once, it may be due to a special event gathering many users in one geographic area (such as a traffic jam or a parade), or an unusual burst of traffic; and in this case it can be safely ignored. If the problem repeats regularly (on a daily basis on peak traffic hours, or on a venue in the coverage area such as a stadium or a concert hall that regularly congregates a large audience), then there is a capacity problem, and the network planification should include additional carriers on that sector, or new sectors (either normal cells or small cells) giving coverage to the problematic areas. Also, special configuration of the parameters can temporarily prepare a sector for a predicted increase in traffic, but it will not solve the problem completely.
- Coverage hole: some UEs located in a specific area inside the coverage of a sector are suffering a low signal level from all the near sectors. This happens normally when there is a geographic obstacle between the UE and the eNodeB antennas. This is usually best detected with drive tests or call traces, but since UEs report bad coverage to their host eNodeBs and their numbers are registered as counters, there is a way for approximately diagnosing this problem. Nevertheless, a high number of bad coverage reports can also indicate that the affected sector is in the border of the network (e.g. near zones where the LTE network is not deployed despite there being still potential users). To discriminate these cases, it must be determined somehow if the cell is in the border or not, either by calculating with its location information, or by inspecting the histogram of UE distance distributions. This is a compound PI (i.e. it has one entry per bin) that counts the number of distance measurements reported by the timing advance of each UE, and it is extracted

from the call traces as a synthetic counter. Anyway, the diagnosis is only approximate, and a definitive solution can only be given after measurements have been done on the specific location.

- CPU Overload: the traffic carried by the sector is too high for its capacity. Specifically, this label is applied on cases where the traffic is causing a high load on the CPU of the eNodeB. An automatic admission control algorithm acts in these cases and rejects new connections, causing a decrease in Accessibility. If the CPU load keeps growing (for instance, because already established connections are demanding an increasing volume of data), then a part of the established connections are dropped, with the consequence of a degradation in Retainability. This label covers a subset of the *Congestion* class, therefore, the root causes are similar.
- Handover to a high traffic site: when users move from the coverage area of a sector to the coverage area of a neighboring sector that is suffering from high traffic, the handover process may be interrupted by the admission control of the second site, causing a failure and a drop of the connection. Although the effects are seen in the first site, the root cause is located in the neighbor, and most likely the neighboring site is also detected as problematic (for instance, because of CPU congestion).
- Hardware problem: this label encompasses problems where the sector is giving a bad service due to a broken or misconfigured hardware element. Normally, the affected element is the antenna, either because the tilt angle is bad, or because the controller (the interface between the antenna and the eNodeB CPU) is faulty. Observation of engineer actions indicating changes in the tilt and system logs indicating hardware malfunction will determine the specific root cause.
- High MME drop call rate *: a large number of connections of the sector are dropped because of a problem in the MME (for instance, bad configuration or high number of managed users).
- High number of drops: problems that were detected because there was a low Retainability but not diagnosed (e.g. the final diagnosis was never reported, or the problem was solved with a restart of the eNodeB) are classified under this label. Since this label encompasses a varied set of problems that may not have a common pattern for being diagnosed, it cannot be used for the data mining process.
- High UL RSSI: problems classified under this label show a high Received Signal Strength Indicator (RSSI) in the uplink channel. The RSSI measures both signal and noise received by the eNodeB antenna. Usually a high RSSI may cause interference problems, leading to a reduced Retainability (since connections drop due to bad quality). If the high RSSI is due to a high number of active users in the sector, the Accessibility may also be affected, indicating that the root cause is not interference, but high traffic.
- IFHO Handover failures *: under some circumstances (normally a configuration problem), handovers to neighboring cells suffer a large number of failures. Since users cannot return to their origin sector (e. g. because they have moved out of the coverage area), their call is dropped. This results in a decrease in Retainability and Handover Success Rate (HOSR). The original cause is usually a configuration problem, for instance, an error in the Automatic Neighbor Relation (ANR) system.

- Interference: there is a source of radio waves in the same frequency as the carrier of the eNodeB. It may be either internal interference (caused by the LTE network where the eNodeB is located) or external. External interference might be caused either by a different LTE network, or a radio source that is not an LTE transmitter. This label is closely related to the High UL RSSI label, and in fact, it is often interchangeable when the interference is internal. The interference label normally implies a radio problem, as opposed to High UL RSSI that may indicate a configuration problem.
- Lack of RET: the antennas of an eNodeB have a configuration parameter called *electrical tilt*, which modifies the phase of the transmitted signals to achieve a similar effect to physically changing the tilt of the antenna. When a sector is diagnosed and the chosen solution is to change the tilt, it is usually done remotely by changing its electrical tilt to avoid the need of on-site works. This label covers the cases where this action failed because the eNodeB did not have this option or it was deactivated.
- Low coverage DL *: the signal of the sector in the Downlink is not properly reaching the distant users. This may be due to a bad configuration (i. e. a bad antenna tilt that causes that the signal is weak in the far reaches of the planned coverage area), or to the fact that the sector is in the border of the LTE network. When the cell is in the border of the network, it is not considered a problem; users that are in the border area are served as best as possible, but it is assumed that service with full quality cannot be given to them. But the two situations are undistinguishable unless the location of the sector is known. This has the consequence of a large number of cases with this problem; but since the number of false positives is guaranteed to be high, a border cell detection algorithm must also be set up to discard cells that are expected to show this behaviour because of their location.
- Low coverage DL *: the eNodeB cannot properly detect the signal of the distant UEs in the uplink. This label has the same particularities than *Low coverage DL*. Most of the cases covered in this label are also tagged as *Low coverage DL*, which is expected because both these labels have a high degree of false positives (mainly due to the contribution of border cells) and because in most cases the factors that cause low coverage in DL (such as bad antenna tilt) will also affect the UL connection.
- Missing neighbor *: This problem is caused when ANR fails to correctly add neighbors. This causes drops in the connections (decrease in Retainability) of users that are physically moving into the coverage area of an adjacent cell that is not configured as neighbor, since the handover protocol will fail (decrease in HOSR). This label encompasses cases that could also be included as handover failures, though not all cases labeled as handover failures can be labelled as missing neighbor. To correctly diagnose these kind of problems, it must be identified that two neighboring sectors have a low mutual HOSR, and then look up in the neighboring sector list if the relation is missing.
- No traffic: this is a problem where an eNodeB is ready for public service, but due to a problem in the configuration of UE authentication, it is not accepting most of the connections (causing a low Accessibility) and therefore it has a very low (or zero) traffic.
- Outage: occasionally, an eNodeB goes offline, causing an outage in the service. Outages may be done on purpose (e.g. for maintenance purposes, or to restart an eNodeB in order to apply a new configuration, etc...) and predicted, giving the opportunity of setting up a

compensation using neighboring eNodeBs. These cases are usually done on low traffic hours and the disconnection is registered in a PM parameter. If the eNodeB is not fully turned off (i.e. only the radio subsystem is stopped), it may even continue reporting CM/PM/FM data to the monitoring system. On the other hand, outages may also be unpredicted, for instance due to a power outage. In this case, eNodeBs will stop reporting to the monitoring system and no PM parameter collects the disconnection. In these cases, the outage can be observed in an increase in the traffic of neighboring cells, that will take the orphaned users, although with a lower QoS. This will also cause an increase in the number of bad coverage reports and a decrease in the Retainability of the neighbors, since the new users have a worse quality of signal.

- Overshooting: this is a configuration problem where the antenna covering a sector of an eNodeB has a wrong tilt angle, causing it to have a larger coverage than it should. The sector will then cover a larger area (and therefore a larger number of users and traffic volume) than expected. This will also cause an increase both in the received RSSI and in the RSSI of neighboring sectors (internal interference). Again, this is a subclass of Interference, but where the specific cause is the antenna tilt.
- RF Rejection due to S1 *: There is a large number of rejected connections due to a failure in the establishment of the S1 bearer. This is due to a configuration problem in the S1 interface between the eNodeB and the core network.
- RF Rejection due to UE *: There are a large number of rejected connections due to failure from the UEs. This may be caused by a concentration of problematic UEs in one specific cell, or more likely by a problem in the configuration of the eNodeB.
- RRC storm: a type of software problem caused by a memory leak in the eNodeB software. It causes the eNodeB to run out of memory and increase its CPU load. With a high CPU load, the traffic control algorithms automatically block the incoming connections, causing reattempts by the UEs. This shows up as a burst of RRC Connection Reattempts, hence the name given to the problem. The problem can be normally mitigated by restarting the eNodeB, but a software patch is the definitive solution of this problem.
- Swapped sector: each eNodeB usually controls several sectors (normally three). Each sector has an independent transmitter connected to an antenna facing on the corresponding angle (normally, with a separation of 120°). When the wrong connection is made (i.e. the transmitter of a sector is connected to the wrong antenna), the sector will be giving service to the wrong users. This will cause bad neighboring relations, since the physical area of a sector does not correspond with its configured area.

APPENDIX C

EXTRACTED MODEL PARAMETERS

C.1 Parametrized distributions

Average CQI

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Log-normal	[0.171, 0.0, 8.995]	0.022	-	-
High Traffic	Log-normal	[0.188, 0.0, 8.23]	0.167	-	-
No Traffic	Gamma	[2.79, 0.0, 2.908]	0.14	0.204	-
CPU Overload	Log-normal	[0.108, 0.0, 8.6]	0.085	-	-
Low Coverage	Log-normal	[0.19, 0.0, 8.232]	0.191	-	-

Average Number of Active UEs

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Gamma	[0.425, 0.0, 1.445]	0.088	0.001	-
High Traffic	Log-normal	[0.976, 0.0, 3.402]	0.142	-	-
No Traffic	Gamma	[0.344, 0.0, 0.011]	0.078	0.355	-
CPU Overload	Log-normal	[1.085, 0.0, 0.219]	0.083	-	-
Low Coverage	Gamma	[0.401, 0.0, 3.321]	0.182	-	-

Average RSSI

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Gumbel-R	[-117.385, 2.336]	0.04	-	-
High Traffic	Normal	[-106.469, 3.551]	0.138	-	-
No Traffic	Laplace	[-119.486, 0.685]	0.19	-	-
CPU Overload	Gumbel-R	[-115.894, 2.007]	0.126	-	-
Low Coverage	Gumbel-R	[-113.885, 2.992]	0.143	-	-

CS Fallback Rate					
Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Beta	[1.386, 3872118.525, -0.0, 44337.295]	0.015	0.064	-
High Traffic	Johnson SB	[11.444, 2.61, -0.001, 1.702]	0.131	-	-
No Traffic	Johnson SB	[9.455, 1.672, -0.007, 8.357]	0.059	0.656	-
CPU Overload	Johnson SB	[24.652, 4.203, -0.045, 27.147]	0.151	0.053	-
Low Coverage	Johnson SB	[28.463, 4.412, -0.029, 33.301]	0.254	0.136	-

Handover Success Rate					
Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Beta	[128319.146, 1.551, -1383.757, 1384.757]	0.032	-	0.426
High Traffic	Johnson SB	[-3.321, 0.742, -0.415, 1.415]	0.206	-	-
No Traffic	Johnson SB	[-0.602, 0.582, -0.012, 1.012]	0.126	0.437	0.197
CPU Overload	Beta	[201.792, 0.954, -13.495, 14.495]	0.129	-	0.105
Low Coverage	Johnson SB	[-10.016, 1.43, -12.785, 13.787]	0.136	-	0.136

Interfreq HO Preaparation Rate					
Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Johnson SB	[-1.099, 0.707, -0.113, 1.114]	0.106	0.628	0.316
High Traffic	Johnson SB	[-31.869, 3.358, -1.588, 2.588]	0.507	0.923	0.026
No Traffic	Beta	[1.072, 1.568, -0.0, 1.003]	0.079	0.731	0.147
CPU Overload	Beta	[0.576, 0.359, -0.108, 1.108]	0.31	0.526	0.316
Low Coverage	Beta	[0.319, 0.11, -0.103, 1.103]	0.321	0.455	0.318

Intrafreq HO Preaparation Rate					
Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Johnson SB	[-7.427, 1.095, -24.908, 25.91]	0.124	0.013	0.867
High Traffic	Johnson SB	[-2.945, 0.43, -0.222, 1.222]	0.109	-	0.41
No Traffic	Johnson SB	[0.455, 0.661, -0.001, 1.021]	0.152	0.52	0.319
CPU Overload	Beta	[103.896, 0.854, -6.735, 7.735]	0.158	-	0.421
Low Coverage	Johnson SB	[-6.494, 0.804, -0.467, 1.467]	0.249	-	0.682

iRAT Rate

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Johnson SB	[4.397, 0.944, -0.0, 1.005]	0.031	0.052	-
High Traffic	Beta	[1.88, 5121.589, -0.0, 26.434]	0.183	-	-
No Traffic	Johnson SB	[6.87, 1.168, -0.006, 31.756]	0.058	0.57	0.004
CPU Overload	Beta	[7.094, 331752302796579.2, -0.005, 807660514448.1]	0.166	0.105	-
Low Coverage	Johnson SB	[8.797, 1.296, -0.001, 43.748]	0.195	0.182	-

Number of Bad Coverage Reports

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Log-normal	[1.558, 0.0, 50.46]	0.033	0.036	-
High Traffic	Log-normal	[0.637, 0.0, 252.312]	0.119	-	-
No Traffic	Gamma	[0.648, 0.0, 13.881]	0.116	0.455	-
CPU Overload	Log-normal	[1.352, 0.0, 53.44]	0.125	-	-
Low Coverage	Log-normal	[1.013, 0.0, 636.855]	0.183	-	-

Number of CPU Overload Alarms

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Gamma	[0.297, 0.0, 21.035]	0.105	0.971	-
High Traffic	Gamma	[0.567, 0.0, 82.732]	0.146	0.718	-
No Traffic	Delta			1.0	-
CPU Overload	Log-normal	[1.288, 0.0, 92.303]	0.185	-	-
Low Coverage	Exponential	[0.0, 3.036]	0.298	0.818	-

Number of ERAB Attempts

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Exponential	[0.0, 5383.581]	0.039	-	-
High Traffic	Log-normal	[0.441, 0.0, 18402.446]	0.118	-	-
No Traffic	Gamma	[0.485, 0.0, 161.999]	0.105	0.416	-
CPU Overload	Log-normal	[0.56, 0.0, 2252.13]	0.113	-	-
Low Coverage	Gamma	[0.623, 0.0, 12800.772]	0.159	-	-

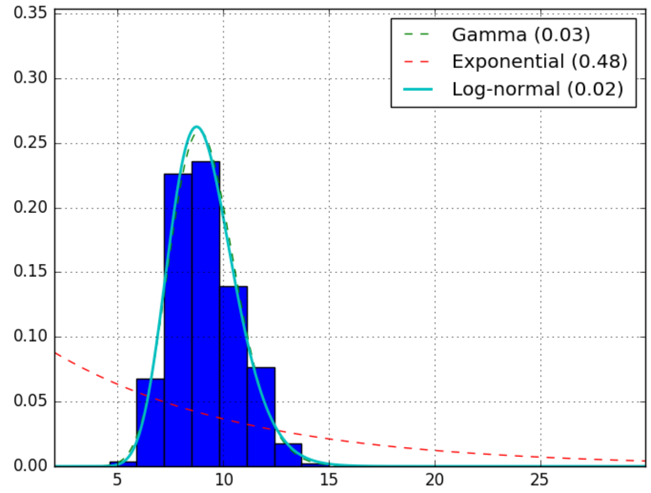
Traffic DL

Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Gamma	[0.617, 0.0, 1.552]	0.062	-	-
High Traffic	Log-normal	[0.357, 0.0, 1.689]	0.23	-	-
No Traffic	Gamma	[0.17, 0.0, 0.084]	0.087	0.219	-
CPU Overload	Gamma	[2.012, 0.0, 0.324]	0.103	-	-
Low Coverage	Exponential	[0.0, 0.514]	0.31	-	-

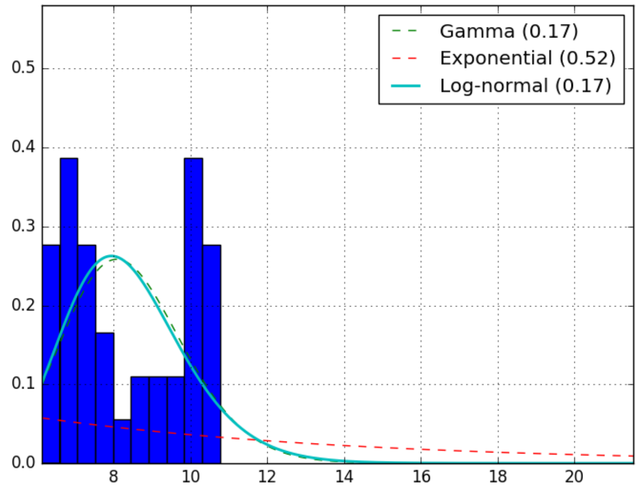
Traffic UL					
Cause	Model	Parameters	D-Value	P_0	P_1
Normal	Gamma	[0.581, 0.0, 0.19]	0.04	-	-
High Traffic	Log-normal	[0.533, 0.0, 0.318]	0.133	-	-
No Traffic	Gamma	[0.212, 0.0, 0.006]	0.082	0.215	-
CPU Overload	Log-normal	[0.925, 0.0, 0.044]	0.155	-	-
Low Coverage	Exponential	[0.0, 0.069]	0.166	-	-

C.2 PDFs

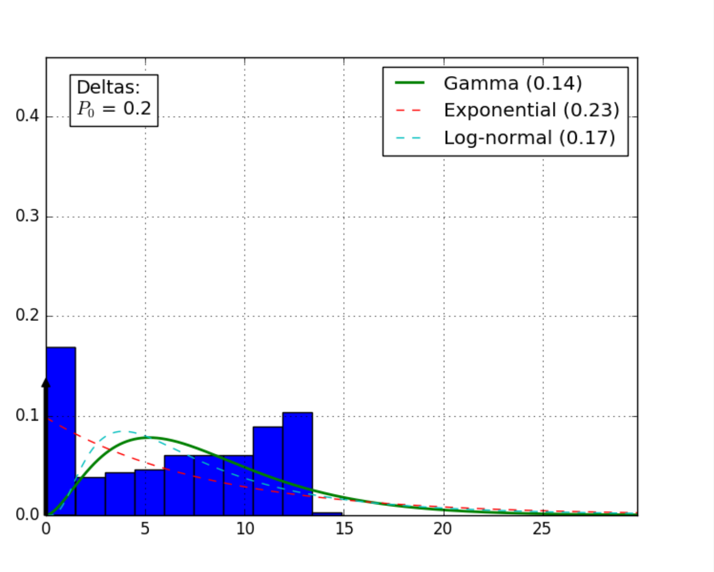
C.2.1 Average CQI



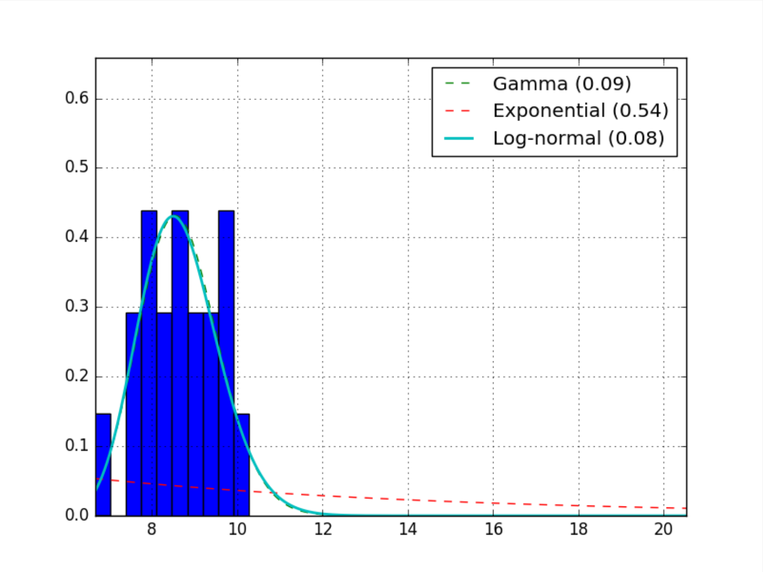
Histogram of *Average CQI* conditioned to *Normal*.



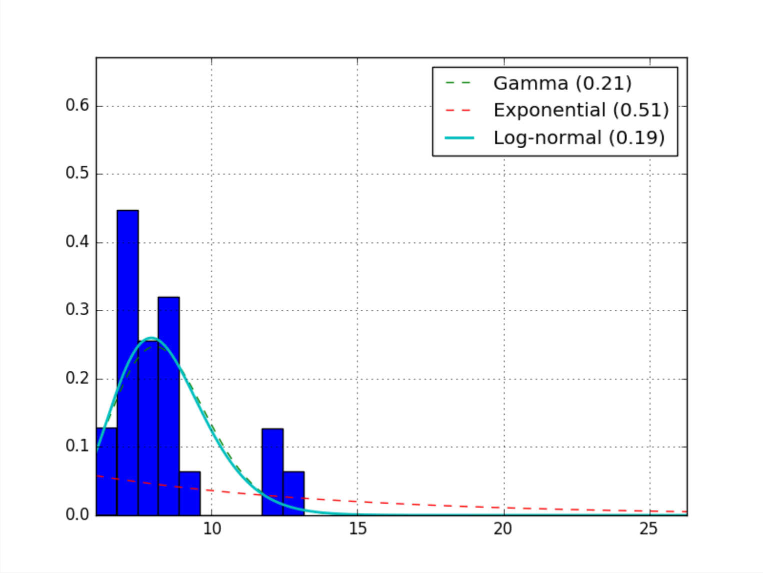
Histogram of *Average CQI* conditioned to *High Traffic*.



Histogram of *Average CQI* conditioned to *No Traffic*.

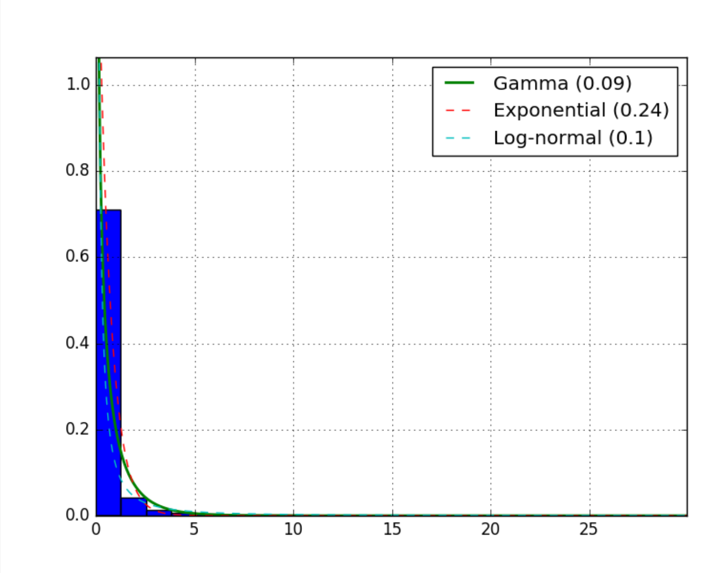


Histogram of *Average CQI* conditioned to *CPU Overload*.

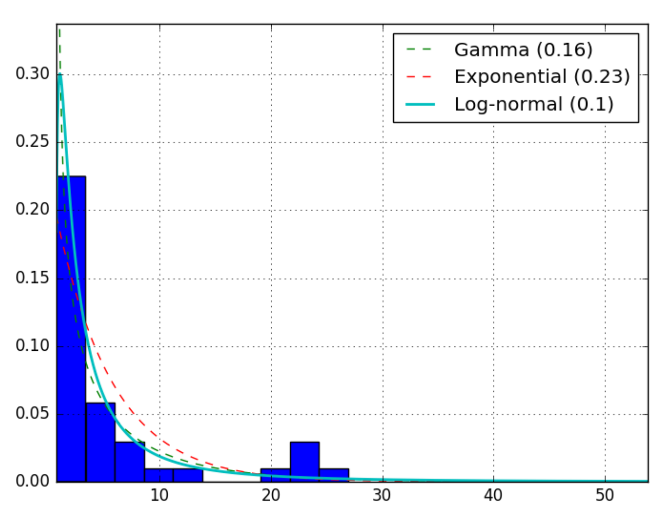


Histogram of *Average CQI* conditioned to *Low Coverage*.

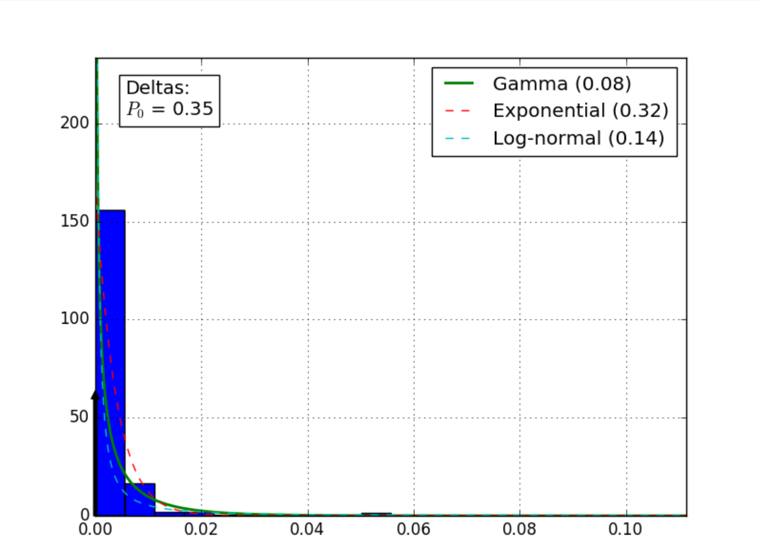
C.2.2 Average Number of Active UEs



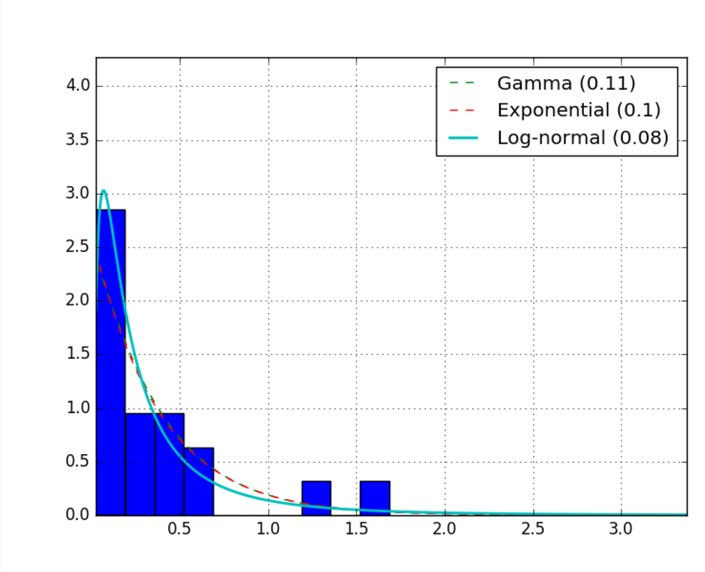
Histogram of *Average Number of Active UEs* conditioned to *Normal*.



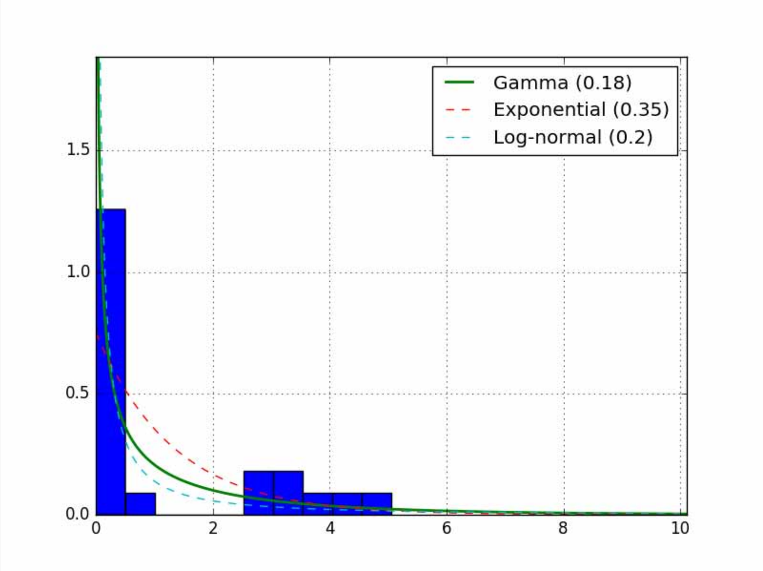
Histogram of *Average Number of Active UEs* conditioned to *High Traffic*.



Histogram of *Average Number of Active UEs* conditioned to *No Traffic*.

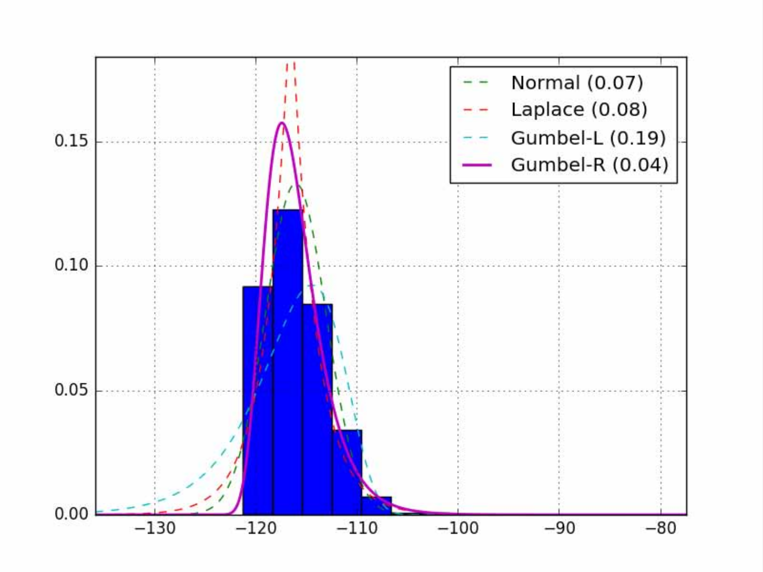


Histogram of *Average Number of Active UEs* conditioned to *CPU Overload*.

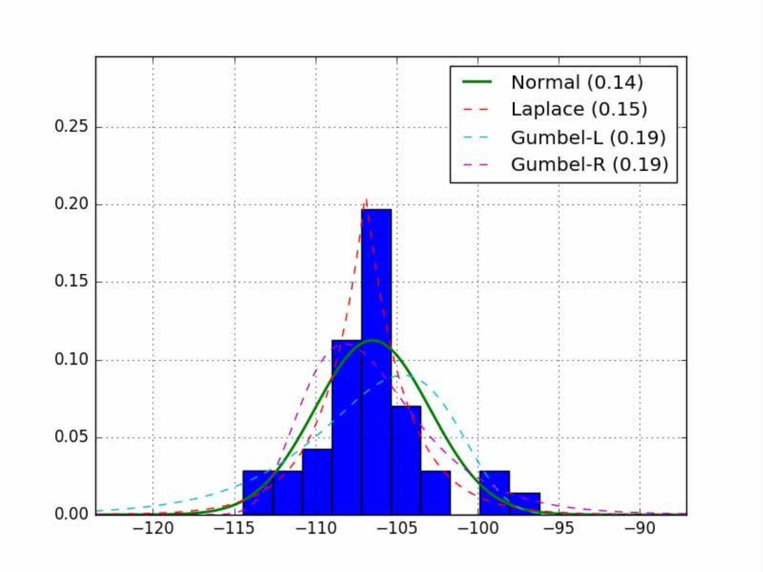


Histogram of *Average Number of Active UEs* conditioned to *Low Coverage*.

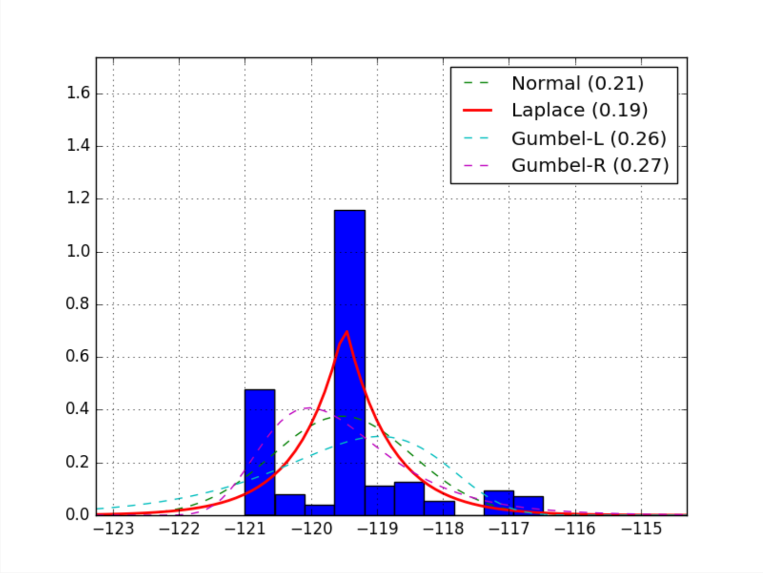
C.2.3 Average RSSI



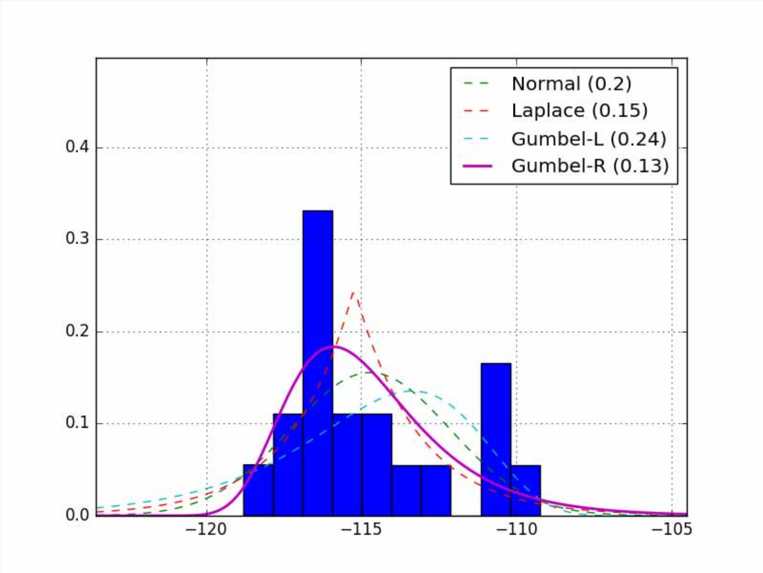
Histogram of *Average RSSI* conditioned to *Normal*.



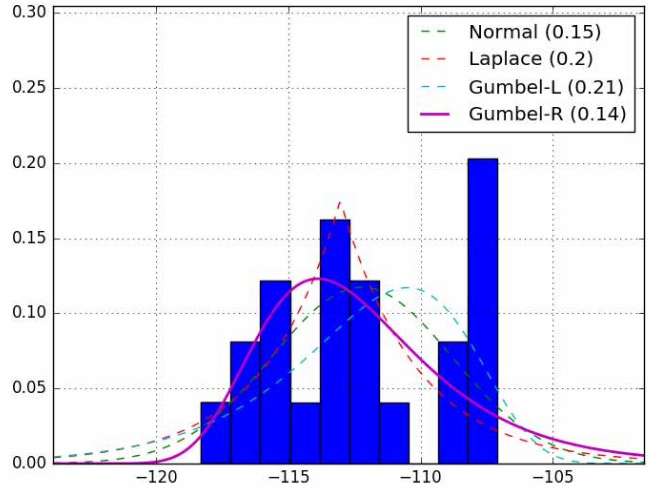
Histogram of *Average RSSI* conditioned to *High Traffic*.



Histogram of *Average RSSI* conditioned to *No Traffic*.

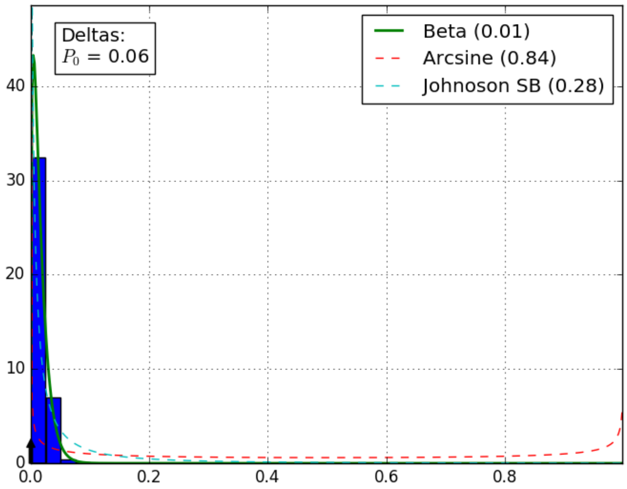


Histogram of *Average RSSI* conditioned to *CPU Overload*.

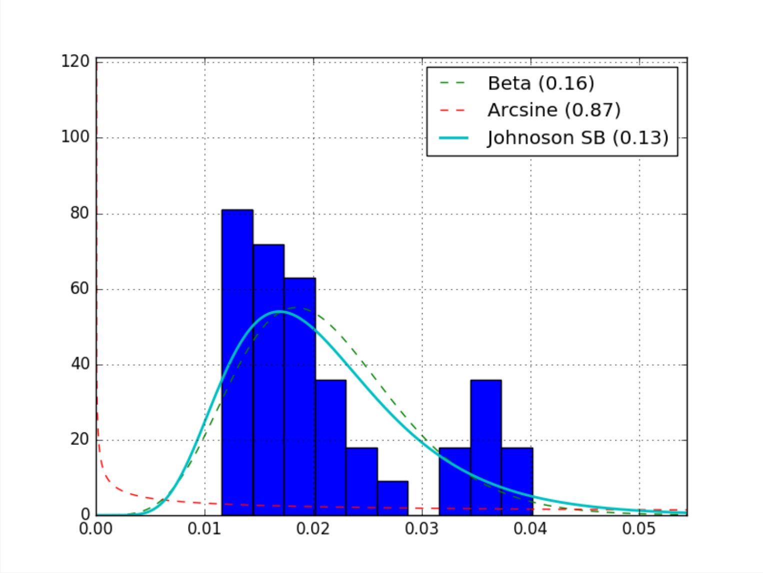


Histogram of *Average RSSI* conditioned to *Low Coverage*.

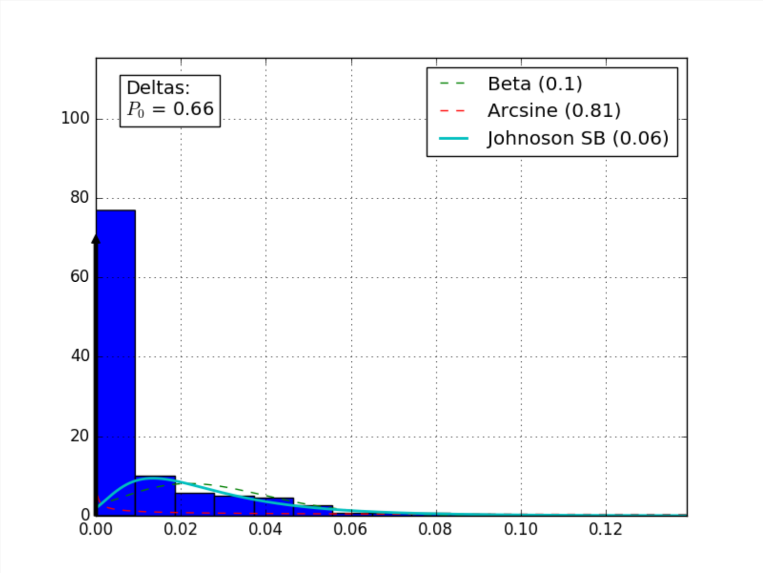
C.2.4 CS Fallback Rate



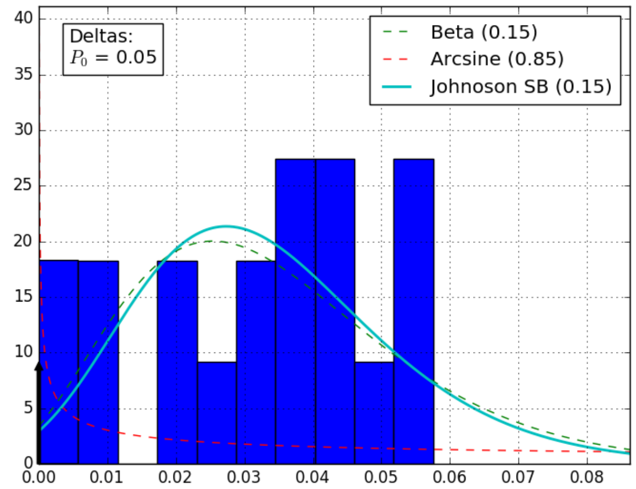
Histogram of *CS Fallback Rate* conditioned to *Normal*.



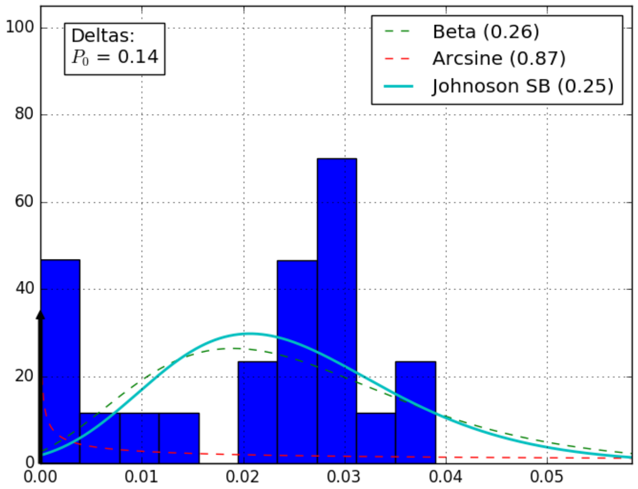
Histogram of *CS Fallback Rate* conditioned to *High Traffic*.



Histogram of *CS Fallback Rate* conditioned to *No Traffic*.

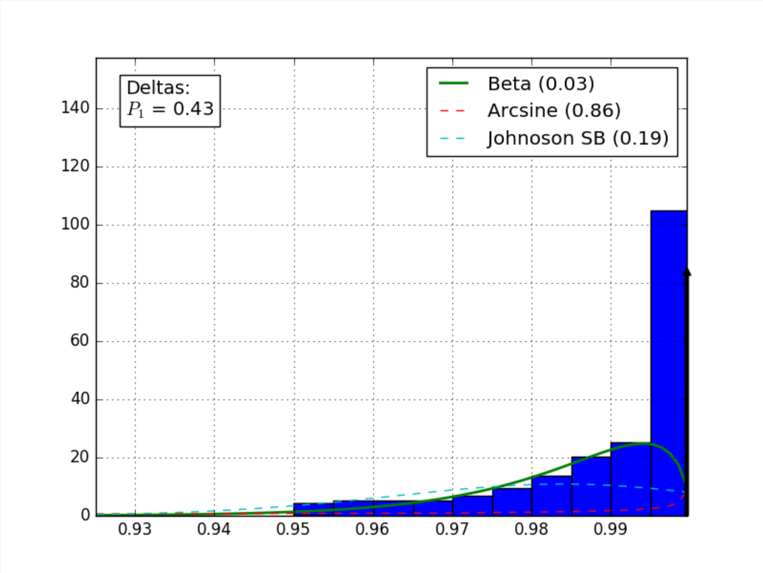


Histogram of *CS Fallback Rate* conditioned to *CPU Overload*.

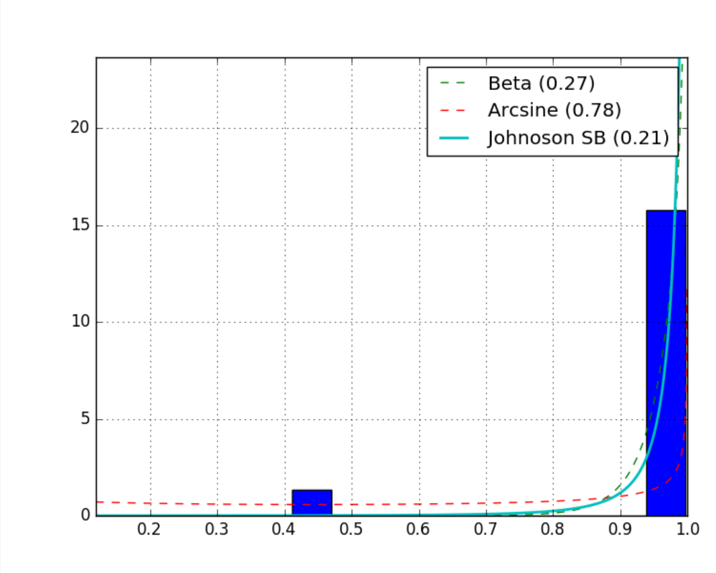


Histogram of *CS Fallback Rate* conditioned to *Low Coverage*.

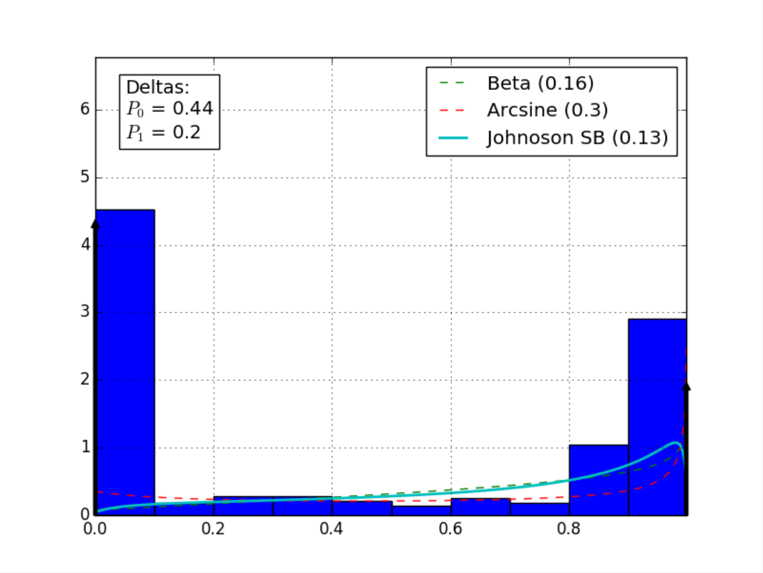
C.2.5 Handover Success Rate



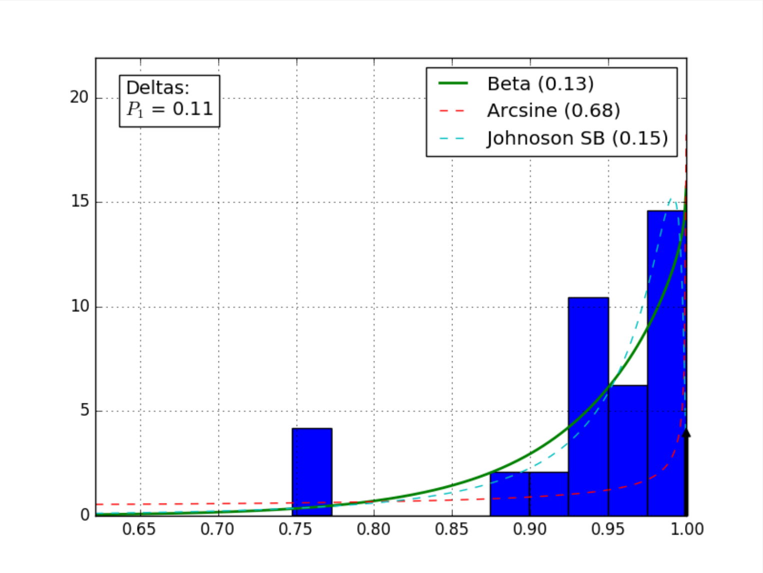
Histogram of *Handover Success Rate* conditioned to *Normal*.



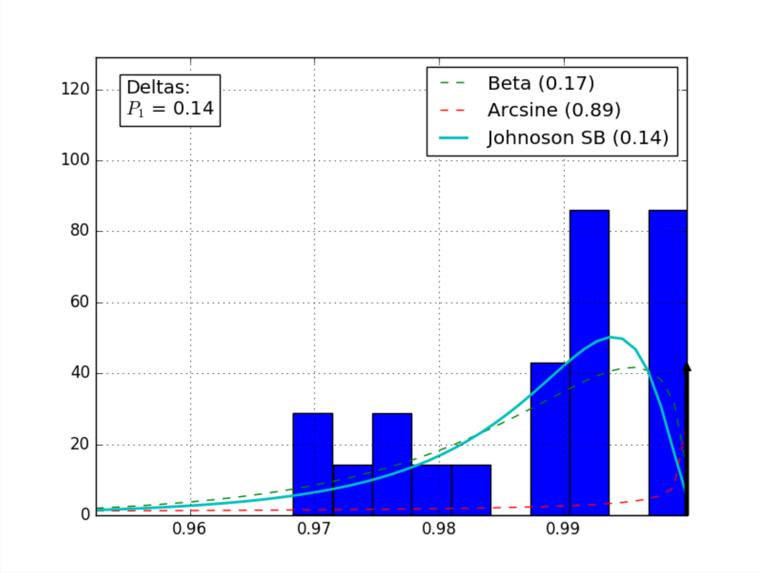
Histogram of *Handover Success Rate* conditioned to *High Traffic*.



Histogram of *Handover Success Rate* conditioned to *No Traffic*.

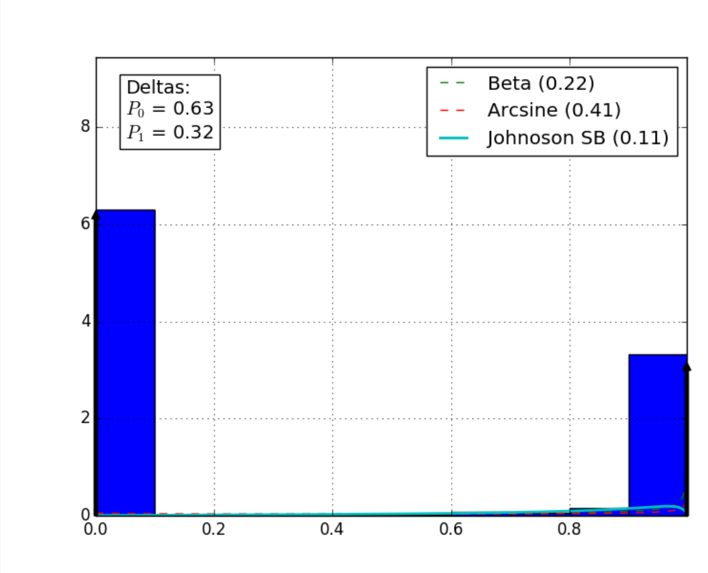


Histogram of *Handover Success Rate* conditioned to *CPU Overload*.

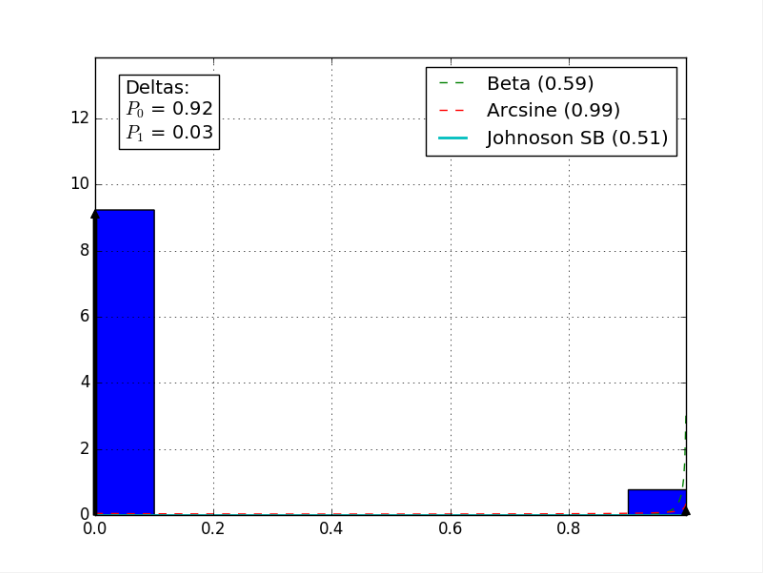


Histogram of *Handover Success Rate* conditioned to *Low Coverage*.

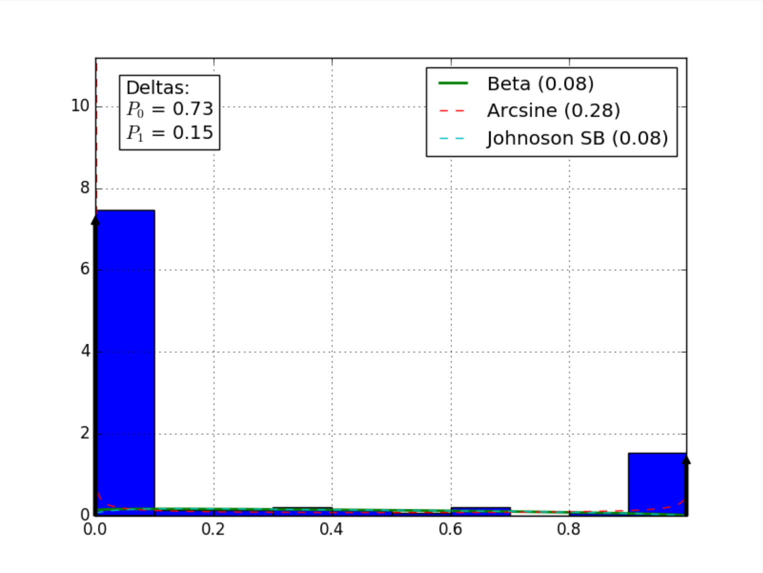
C.2.6 Interfreq HO Preaparation Rate



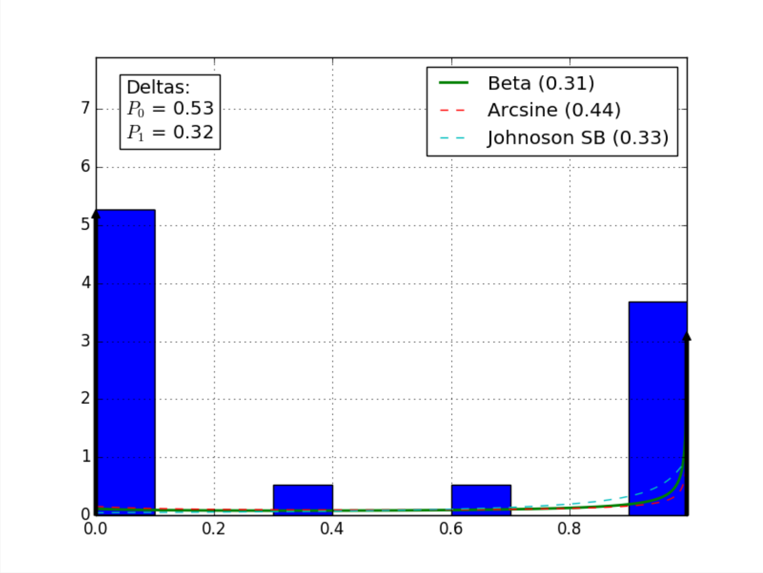
Histogram of *Interfreq HO Preaparation Rate* conditioned to *Normal*.



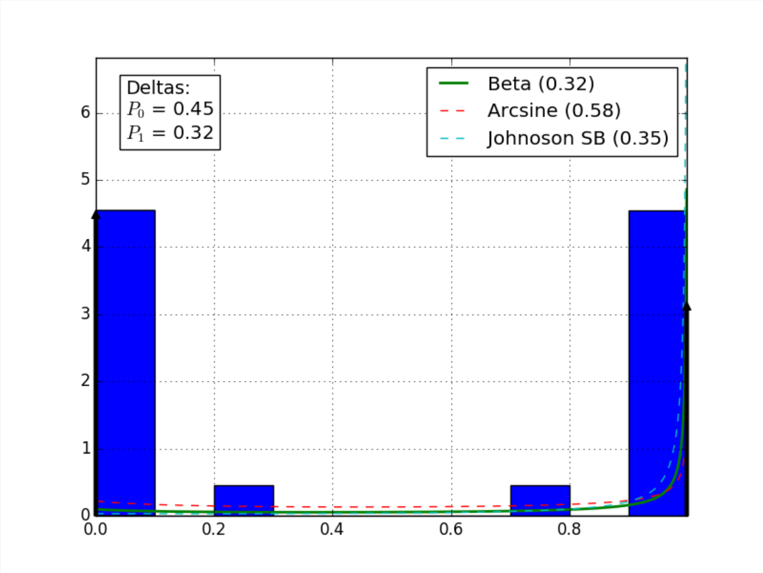
Histogram of *Interfreq HO Preaparation Rate* conditioned to *High Traffic*.



Histogram of *Interfreq HO Preaparation Rate* conditioned to *No Traffic*.

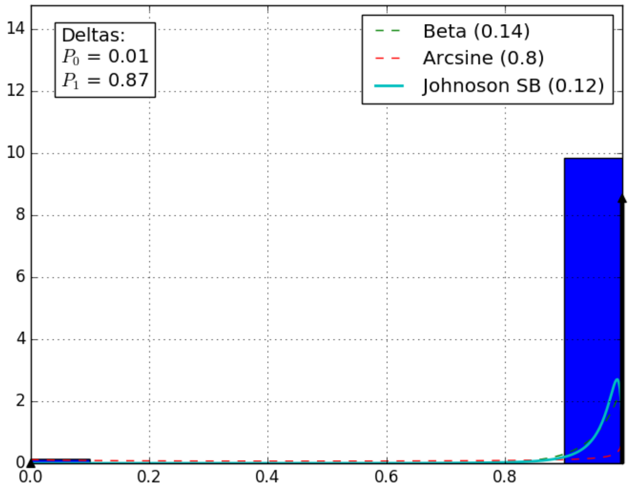


Histogram of *Interfreq HO Preaparation Rate* conditioned to *CPU Overload*.

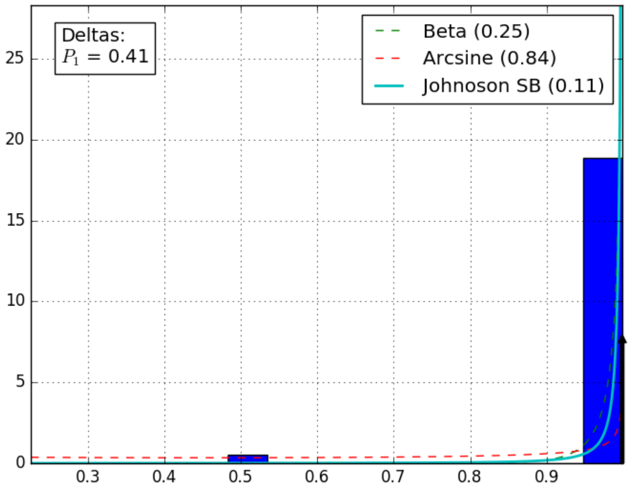


Histogram of *Interfreq HO Preaparation Rate* conditioned to *Low Coverage*.

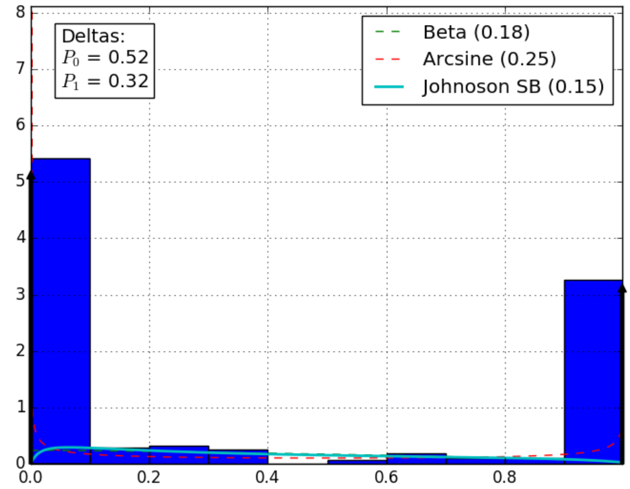
C.2.7 Intrafreq HO Preaparation Rate



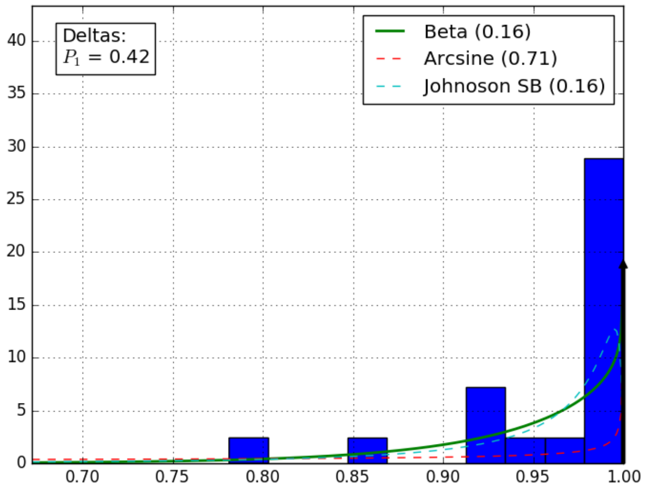
Histogram of *Intrafreq HO Preaparation Rate* conditioned to *Normal*.



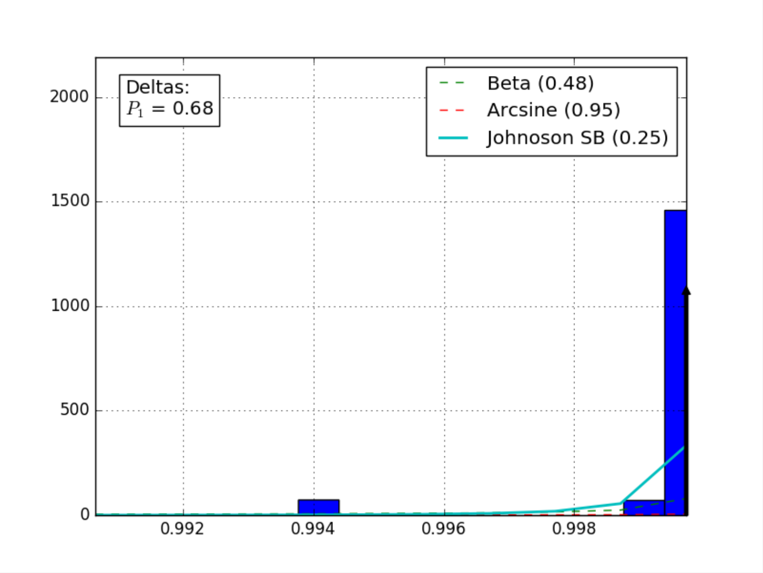
Histogram of *Intrafreq HO Preaparation Rate* conditioned to *High Traffic*.



Histogram of *Intrafreq HO Preaparation Rate* conditioned to *No Traffic*.

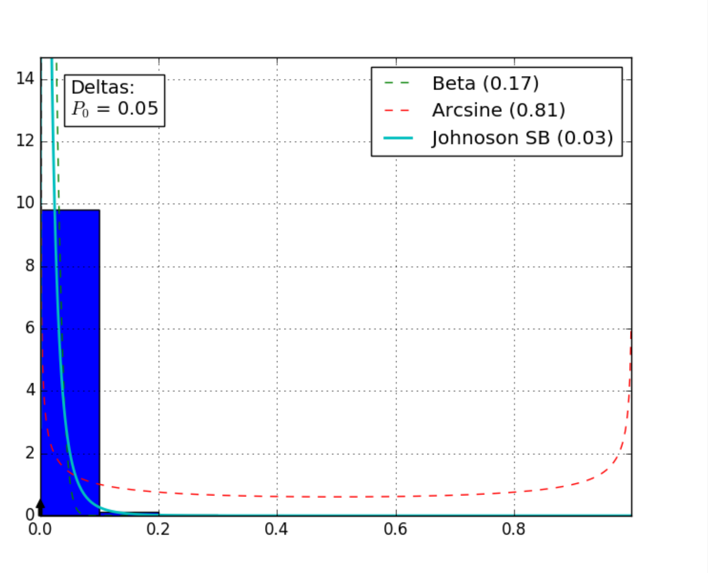


Histogram of *Intrafreq HO Preaparation Rate* conditioned to *CPU Overload*.

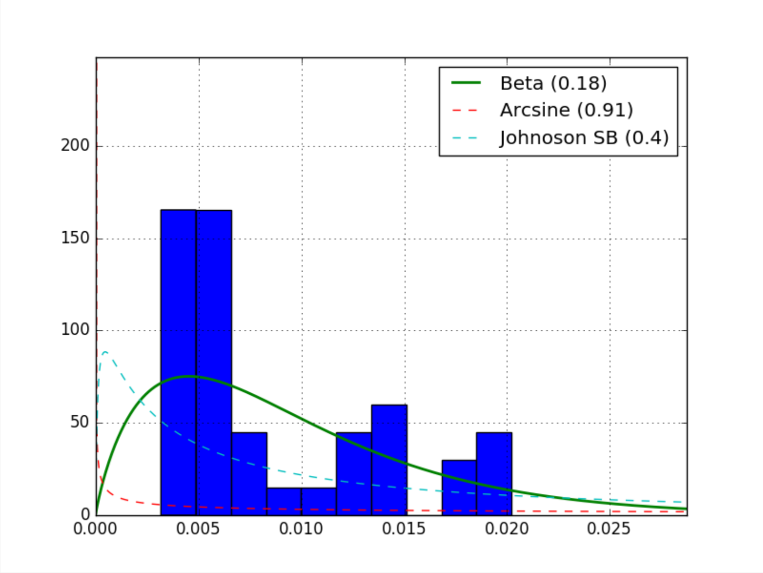


Histogram of *Intrafreq HO Preaparation Rate* conditioned to *Low Coverage*.

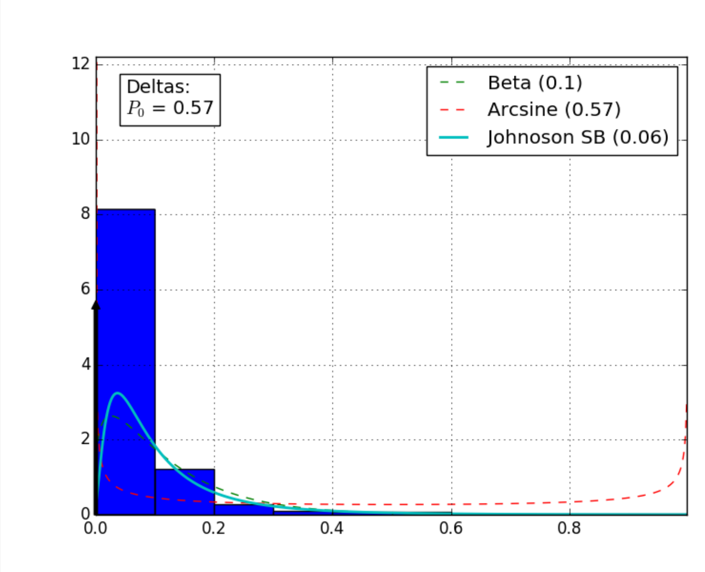
C.2.8 iRAT rate



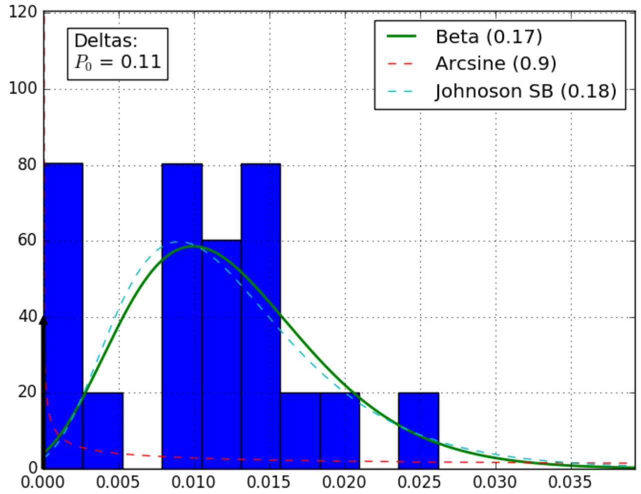
Histogram of *iRAT rate* conditioned to *Normal*.



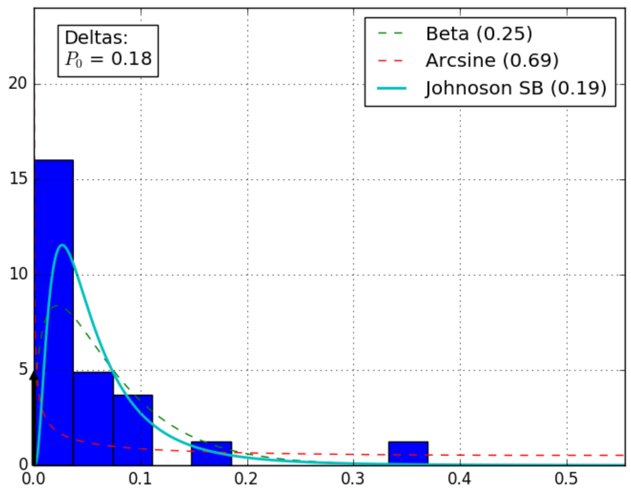
Histogram of *iRAT rate* conditioned to *High Traffic*.



Histogram of *iRAT rate* conditioned to *No Traffic*.

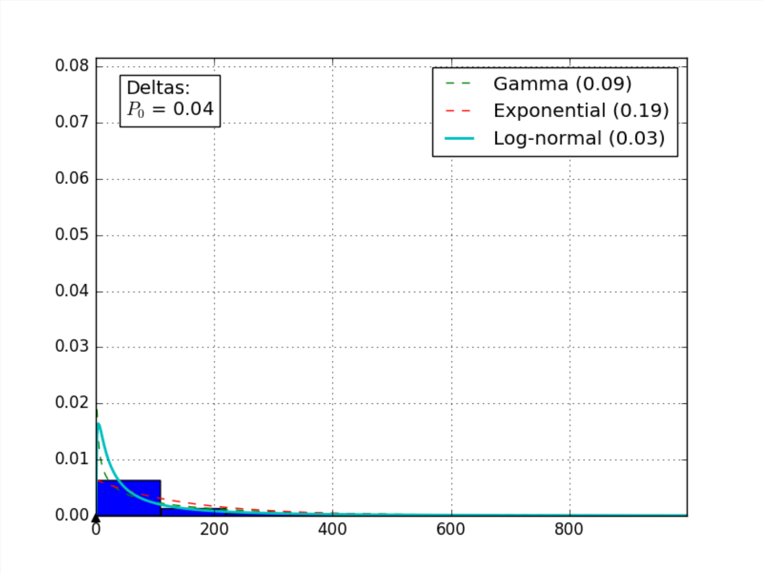


Histogram of *iRAT rate* conditioned to *CPU Overload*.

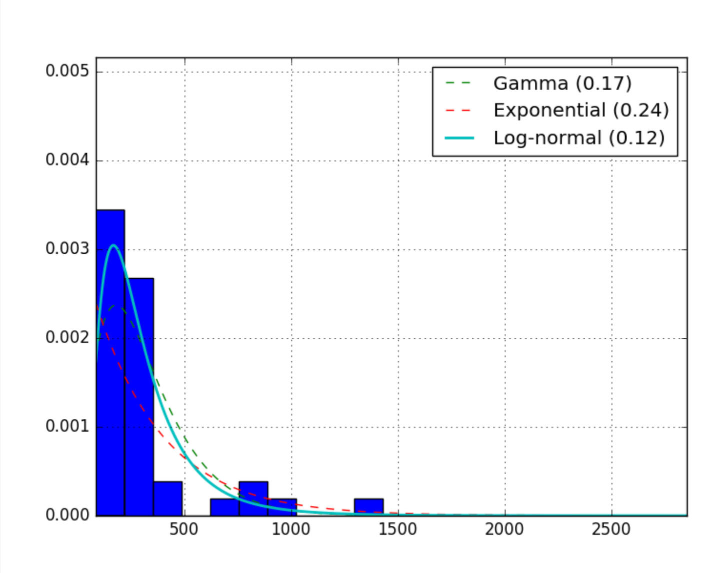


Histogram of *iRAT rate* conditioned to *Low Coverage*.

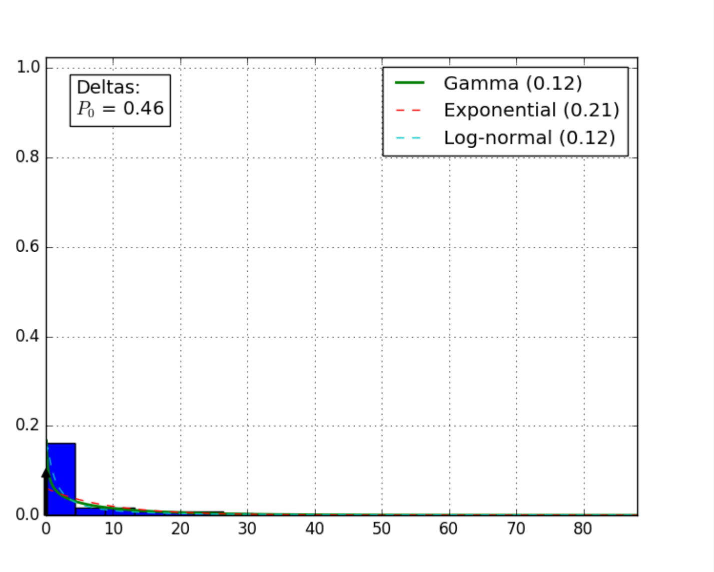
C.2.9 Number of Bad Coverage Reports



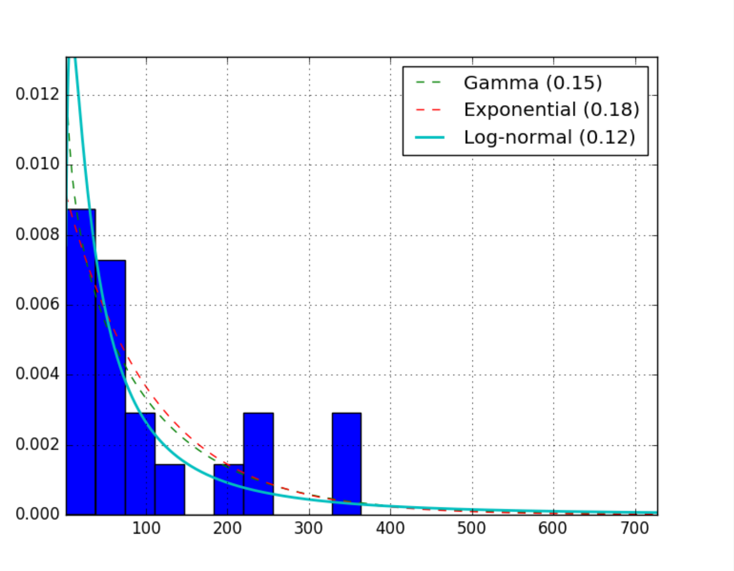
Histogram of *Number of Bad Coverage Reports* conditioned to *Normal*.



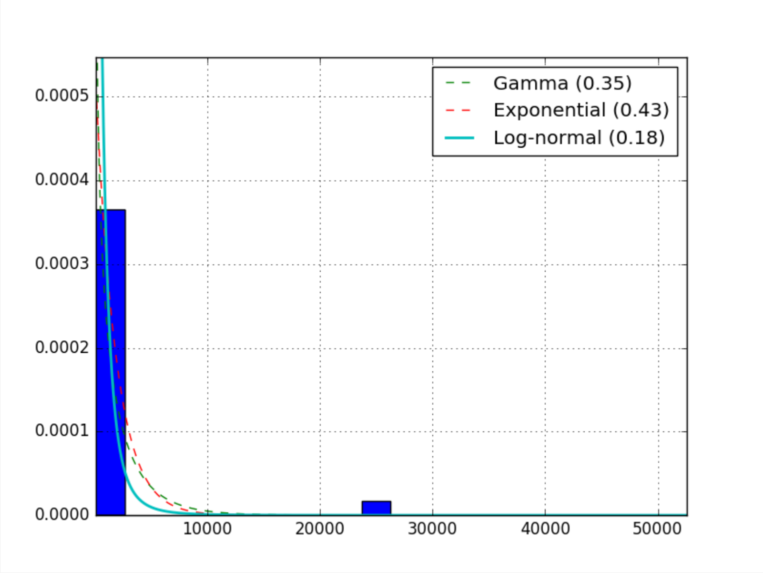
Histogram of *Number of Bad Coverage Reports* conditioned to *High Traffic*.



Histogram of *Number of Bad Coverage Reports* conditioned to *No Traffic*.

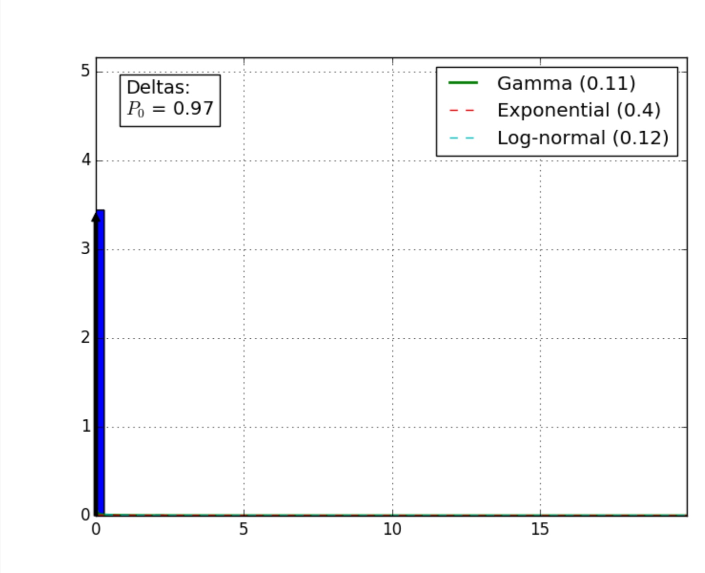


Histogram of *Number of Bad Coverage Reports* conditioned to *CPU Overload*.

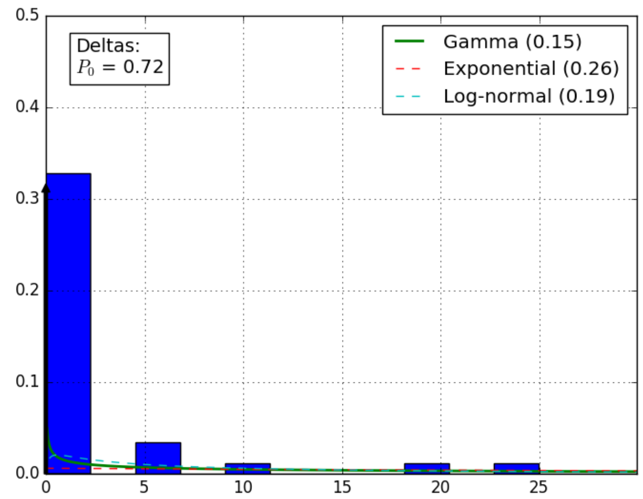


Histogram of *Number of Bad Coverage Reports* conditioned to *Low Coverage*.

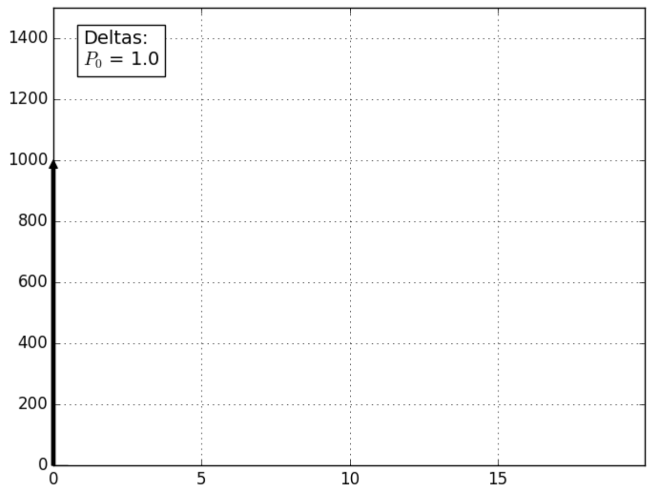
C.2.10 Number of CPU Overload Alarms



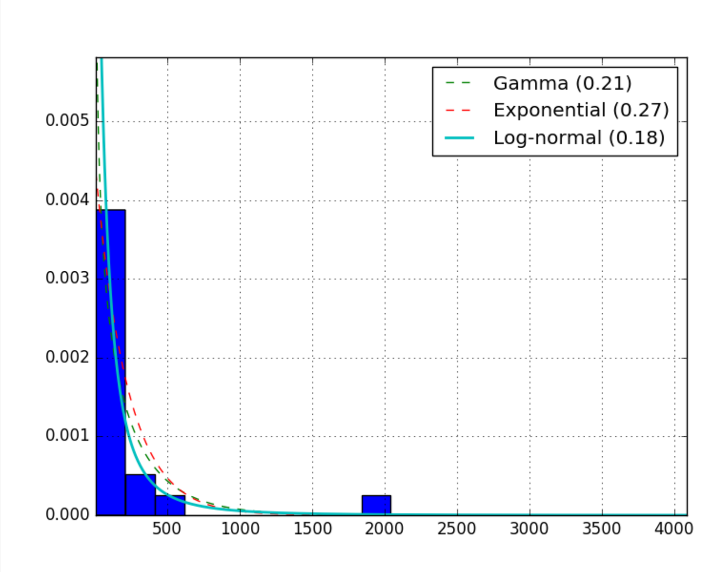
Histogram of *Number of CPU Overload Alarms* conditioned to *Normal*.



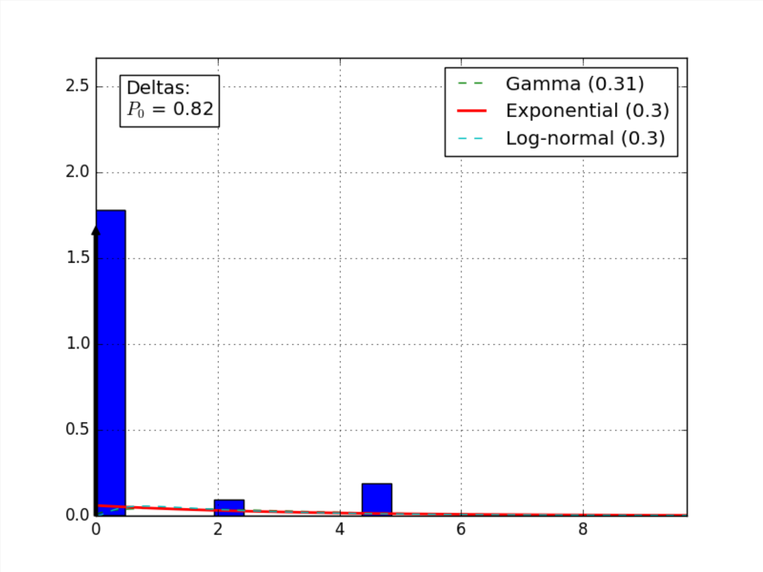
Histogram of *Number of CPU Overload Alarms* conditioned to *High Traffic*.



Histogram of *Number of CPU Overload Alarms* conditioned to *No Traffic*.

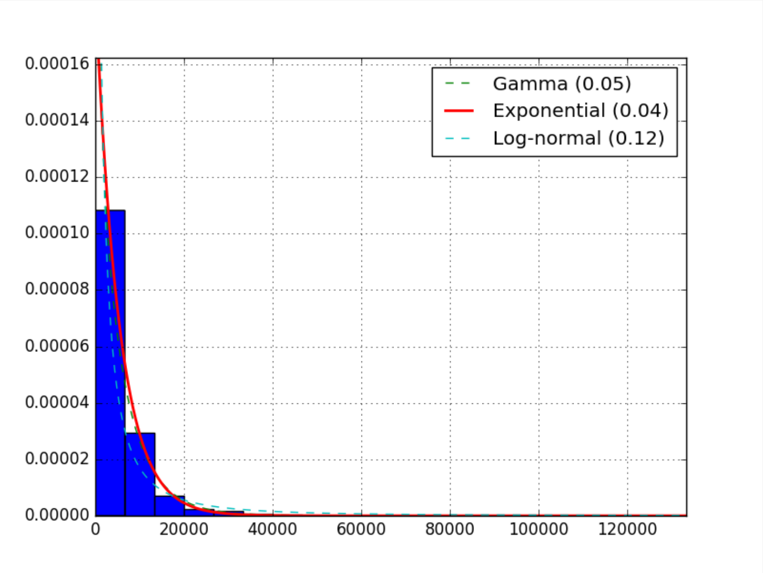


Histogram of *Number of CPU Overload Alarms* conditioned to *CPU Overload*.

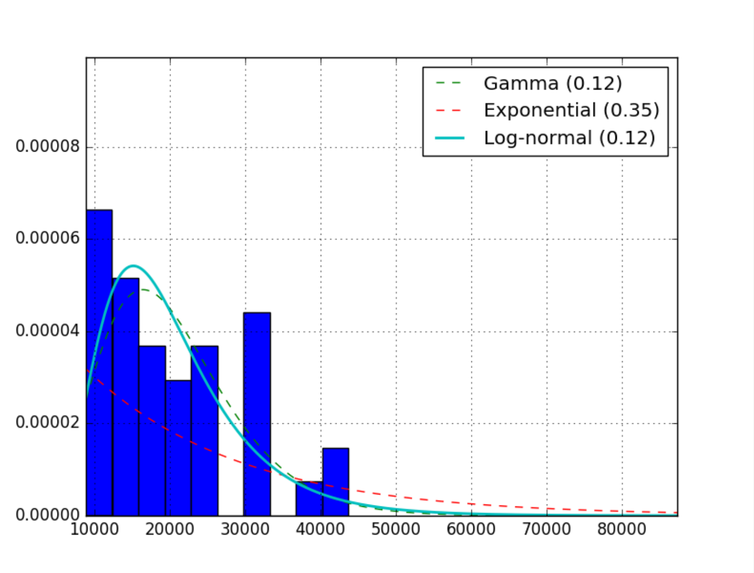


Histogram of *Number of CPU Overload Alarms* conditioned to *Low Coverage*.

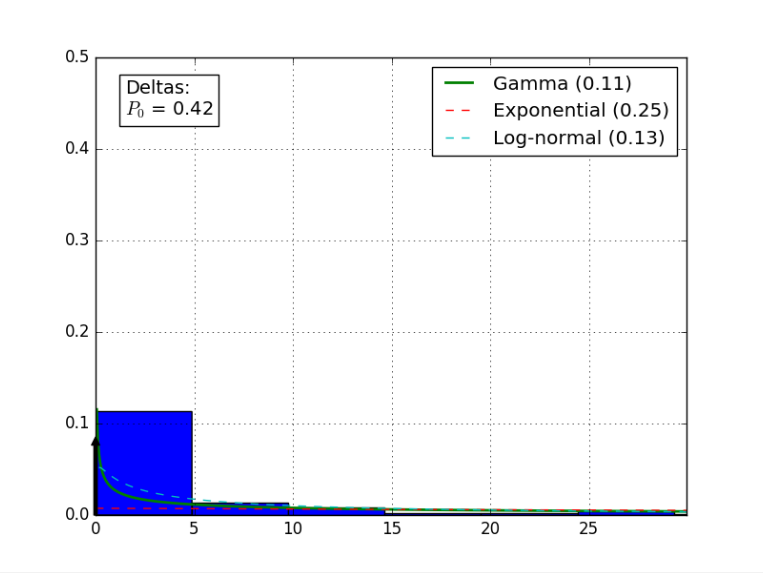
C.2.11 Number of ERAB Attempts



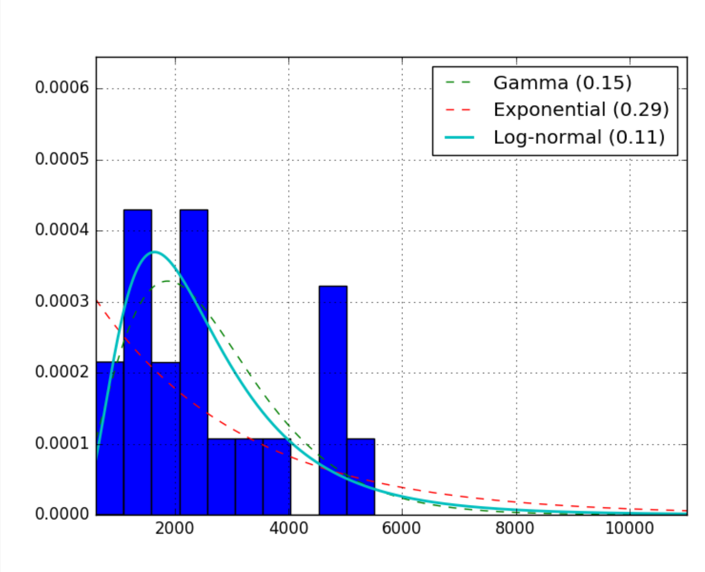
Histogram of *Number of ERAB Attempts* conditioned to *Normal*.



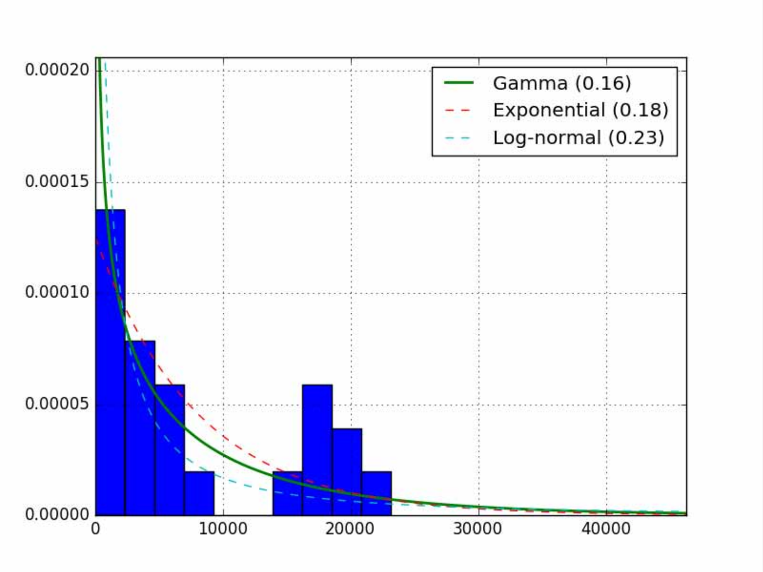
Histogram of *Number of ERAB Attempts* conditioned to *High Traffic*.



Histogram of *Number of ERAB Attempts* conditioned to *No Traffic*.

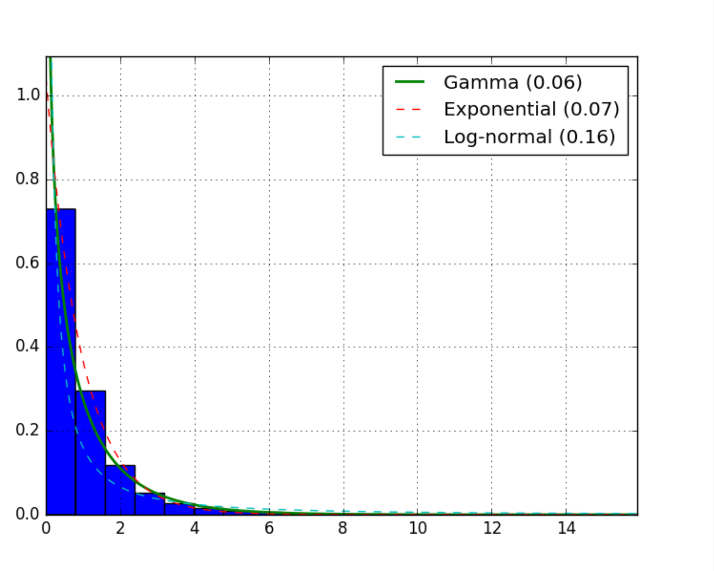


Histogram of *Number of ERAB Attempts* conditioned to *CPU Overload*.

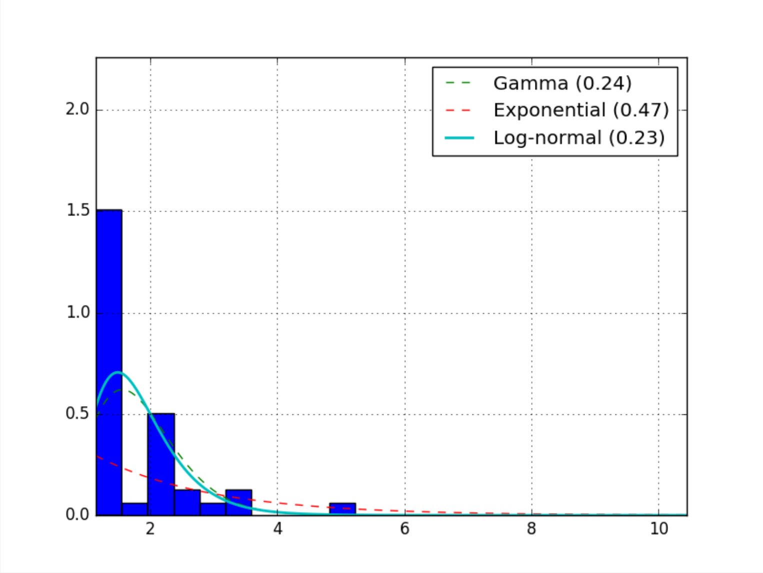


Histogram of *Number of ERAB Attempts* conditioned to *Low Coverage*.

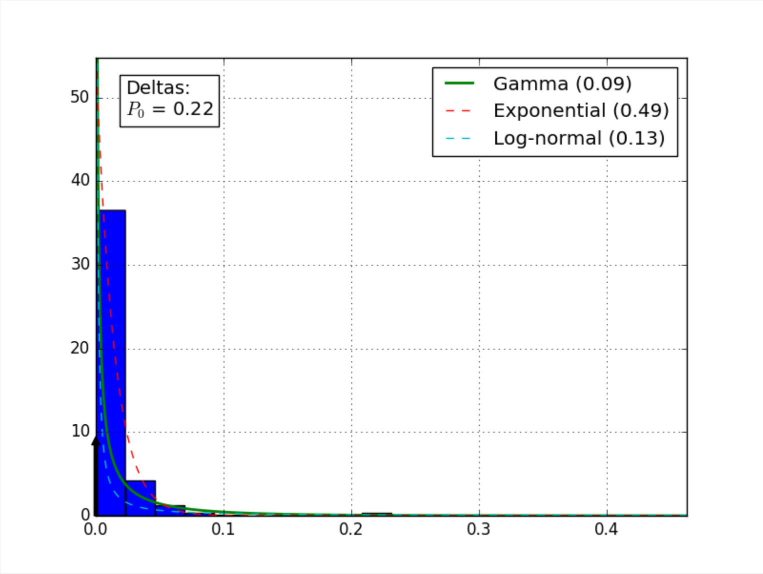
C.2.12 Traffic DL



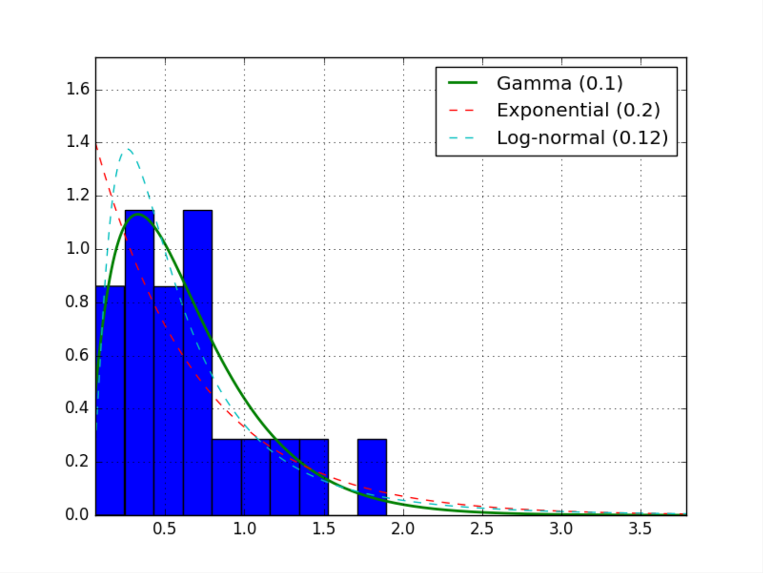
Histogram of *DL Traffic Volume* conditioned to *Normal*.



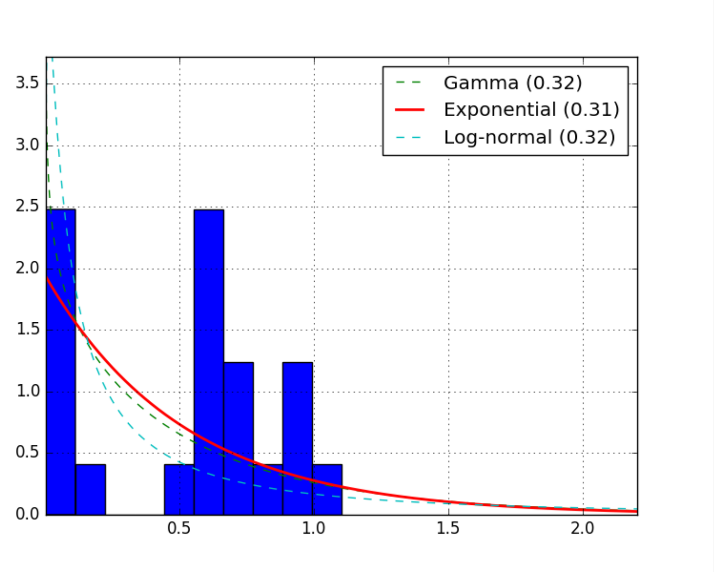
Histogram of *DL Traffic Volume* conditioned to *High Traffic*.



Histogram of *DL Traffic Volume* conditioned to *No Traffic*.

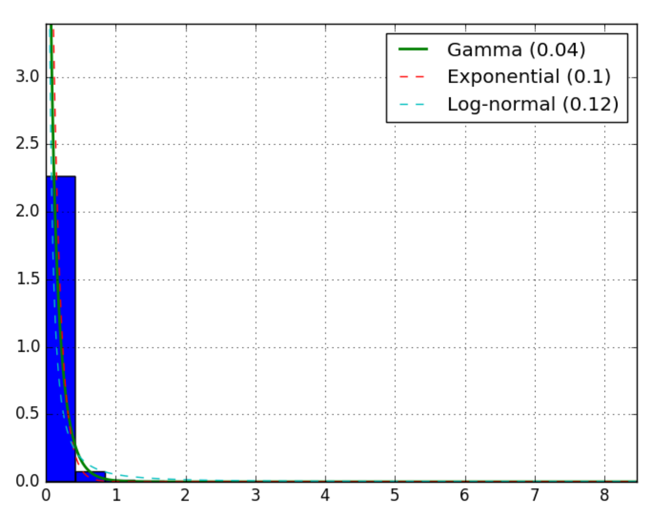


Histogram of *DL Traffic Volume* conditioned to *CPU Overload*.

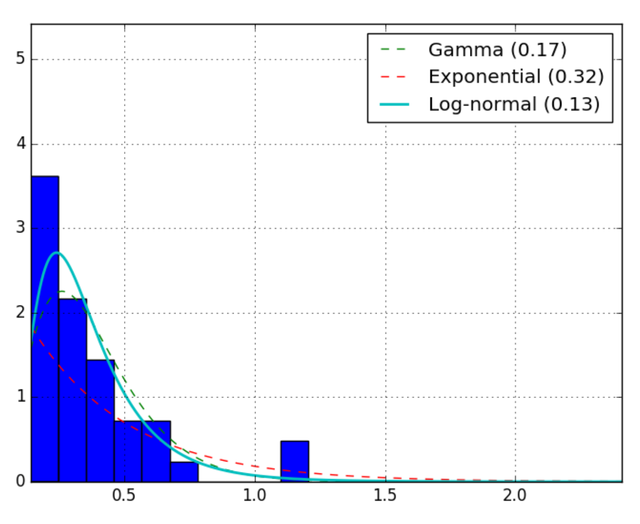


Histogram of *DL Traffic Volume* conditioned to *Low Coverage*.

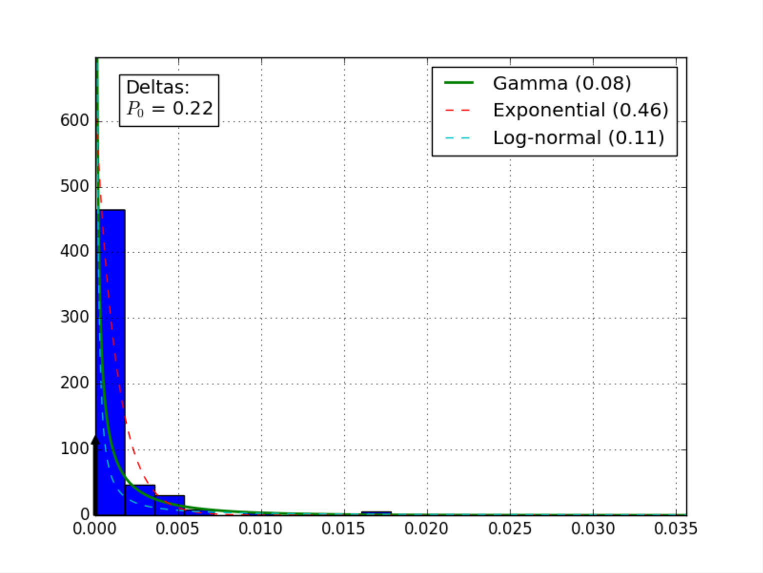
C.2.13 Traffic UL



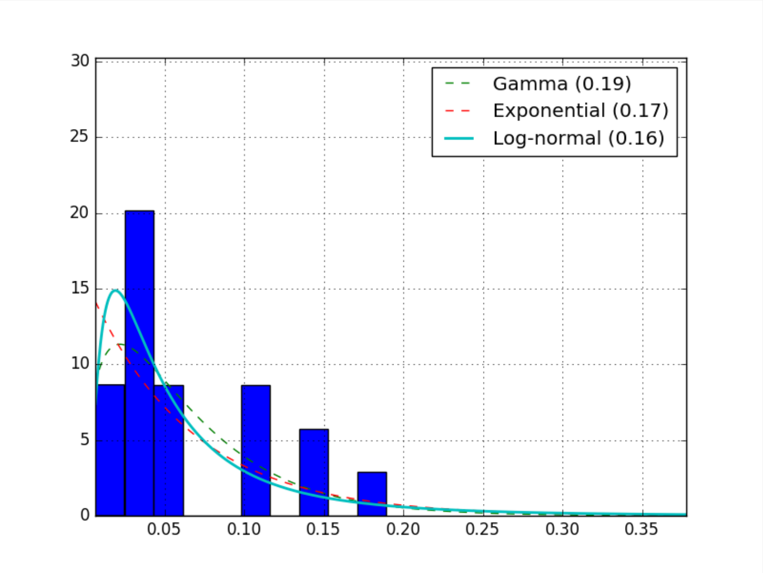
Histogram of *UL Traffic Volume* conditioned to *Normal*.



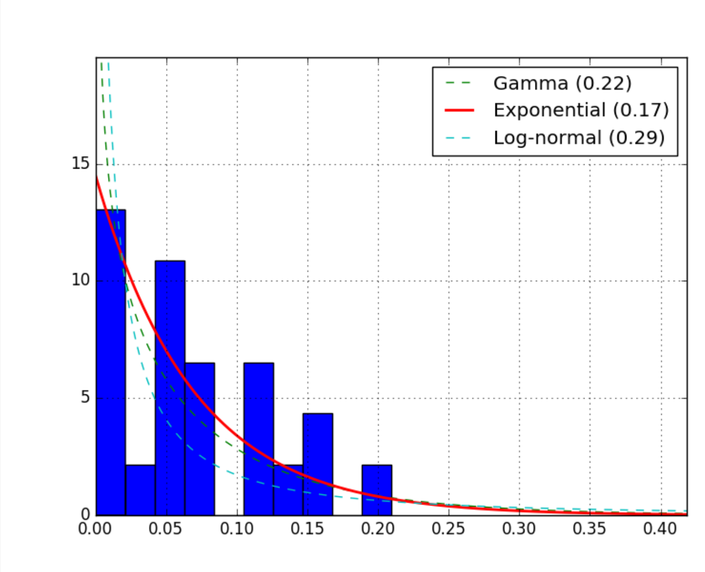
Histogram of *UL Traffic Volume* conditioned to *High Traffic*.



Histogram of *UL Traffic Volume* conditioned to *No Traffic*.



Histogram of *UL Traffic Volume* conditioned to *CPU Overload*.



Histogram of *UL Traffic Volume* conditioned to *Low Coverage*.

Appendix D

Resumen en español

D.1 Motivación

En las últimas décadas se ha dado un cambio de paradigma hacia la movilidad en las telecomunicaciones. Tradicionalmente las telecomunicaciones han implicado equipos vinculados a un lugar, pero tras este cambio el foco pasa a situarse a la comunicación entre usuarios que pueden moverse a grandes velocidades. Este nuevo enfoque involucra un amplio abanico de servicios orientados a una creciente base de usuarios. Como consecuencia las comunicaciones móviles acaparan una proporción creciente de la industria de las telecomunicaciones, y cobran cada vez más importancia en la sociedad moderna.

Las comunicaciones móviles para las masas comenzaron con la telefonía, y gradualmente se desplazaron hacia los servicios de datos. Los antecedentes de la telefonía móvil se remontan a la II Guerra Mundial. Sin embargo, la primera llamada por teléfono móvil utilizando un prototipo de una red celular moderna se realizó oficialmente el 3 de Abril de 1973 en Manhattan. El autor de dicha llamada fue Martin Cooper, un empleado de Motorola, que telefoneó a la sede de Bell Labs en Nueva Jersey. A principios de los años 90, una funcionalidad secundaria de GSM (Global System for Mobile Communications) comenzó a ser ampliamente utilizada por los usuarios: el primer mensaje SMS (Short Message System) fue generado por una máquina y enviado en el Reino Unido el 3 de Diciembre de 1992. En 1993 el primer mensaje SMS entre usuarios se envió en Finlandia. El SMS tuvo una gran acogida entre los usuarios dado su bajo coste, marcando una tendencia no esperada en el mundo de las telecomunicaciones. El servicio de voz ya no era la única opción. A finales de los 90 los operadores de redes móviles comenzaron a ofrecer servicios de datos. En la actualidad las redes móviles han evolucionado hasta un punto en el que la voz no es el servicio más utilizado. En la actualidad, las redes están principalmente orientadas a la transmisión de datos y los terminales han evolucionado hasta dar lugar a los smartphones, que son más cercanos a los ordenadores que a los teléfonos móviles de los años 90.

Para hacer frente a estos cambios, a lo largo del tiempo, varias generaciones de redes celulares han sido desarrolladas, desplegadas y operadas para dar servicio. La Primera Generación (1G) comenzó en Japón en 1979 y en los países nórdicos en 1981. A lo largo de los años 80, las redes 1G se desplegaron en otras regiones. Todas estas redes eran analógicas e incompatibles entre sí. La Segunda Generación (2G) de redes celulares apareció en los años 90. Dos estándares compitieron para ocupar el mercado: GSM en Europa y CDMA (Code Division Multiple Access) en los Estados Unidos. Estas redes superaron a 1G al añadir transmisión digital en lugar de analógica, seguridad mejorada, SMS y movilidad entre países que usasen el mismo estándar. Con la 2G, las comunicaciones móviles tuvieron una amplia difusión entre los usuarios, y la demanda de ancho de banda comenzó a crecer y a crear la necesidad de tecnologías mejoradas. Como respuesta a este crecimiento, la Tercera Generación (3G) se introdujo en la primera década del nuevo milenio. 3G se diseñó con la transmisión de datos como principal caso de uso (es decir, conmutación de paquetes en lugar de circuitos, comúnmente utilizada en la telefonía, aunque 3G soporta ambas). La red 3G utilizaba tan sólo un único protocolo, UMTS (Universal Mobile Telecommunications System), permitiendo por fin la movilidad global. Con la continuamente creciente demanda de ancho de banda, la Cuarta Generación (4G) pronto se hizo necesaria. LTE (Long Term Evolution) se adoptó como el protocolo para 4G, y ha sido recientemente desplegado en redes a nivel global. Además de ofrecer un mayor ancho de banda, LTE utiliza una arquitectura de red totalmente rediseñada y simplificada e introduce nuevos mecanismos para mejorar la calidad de servicio. En la actualidad se está investigando y estandarizando la Quinta Generación (5G) de redes móviles. Se espera que esta nueva generación traiga mayores anchos de banda así como mejoras en la Operación y Mantenimiento (O&M) que reduzca costes y tiempos de espera.

En este escenario de demanda creciente en ancho de banda, cobertura y calidad, la inversión en infraestructura (Capital Expenditure, CAPEX) debe ser alta. Las tareas de O&M cobran una gran importancia, ya que las redes deben estar siempre optimizadas para ofrecer el mejor servicio posible, y las tareas de resolución de problemas hacerse con el menor impacto en la percepción del usuario. Esto fuerza a que la inversión en OPEX (Operational Expenditure) también crezca significativamente. En un mercado con una fuerte competencia, minimizar la inversión en OPEX se convierte en una ventaja competitiva clave.

Para reducir OPEX, la automatización de O&M se usa cuando es posible, liberando a los expertos humanos de tareas repetitivas. Las redes con un alto grado de automatización reciben el nombre de Redes Autoorganizadas (Self Organizing Networks, SON [1]). Las funciones SON pertenecen a tres categorías: autoconfiguración (Self-configuration, los elementos nuevos se configuran de forma automática), autooptimización (Self-optimization, los parámetros de la red se actualizan de forma automática para dar el mejor servicio posible) y autocuración (Self-healing, la red se recupera automáticamente de problemas). El interés de los operadores de red y de los grupos de investigación ha dado lugar a varios proyectos de investigación y consorcios de industria: CELTIC Gandalf [2], FP7 E3 [3], FP7 SOCRATES [4], SELF-NET [5], UniverSelf [6], SEMAFOUR [7] y COMMUNE [8]. Estos proyectos no cubren de forma igualitaria las diferentes funciones SON; Self-healing es el problema menos estudiado dadas las limitaciones y los desafíos presentes en esta línea de trabajo.

Self-healing tiene cuatro procesos principales: detección (identificar que los usuarios tienen problemas en una celda), compensación (redirigir los recursos de la red para cubrir a los usuarios afectados), diagnosis (encontrar la causa de dichos problemas) y recuperación (realizar las acciones necesarias para devolver los elementos afectados a su operación normal). Entre estos

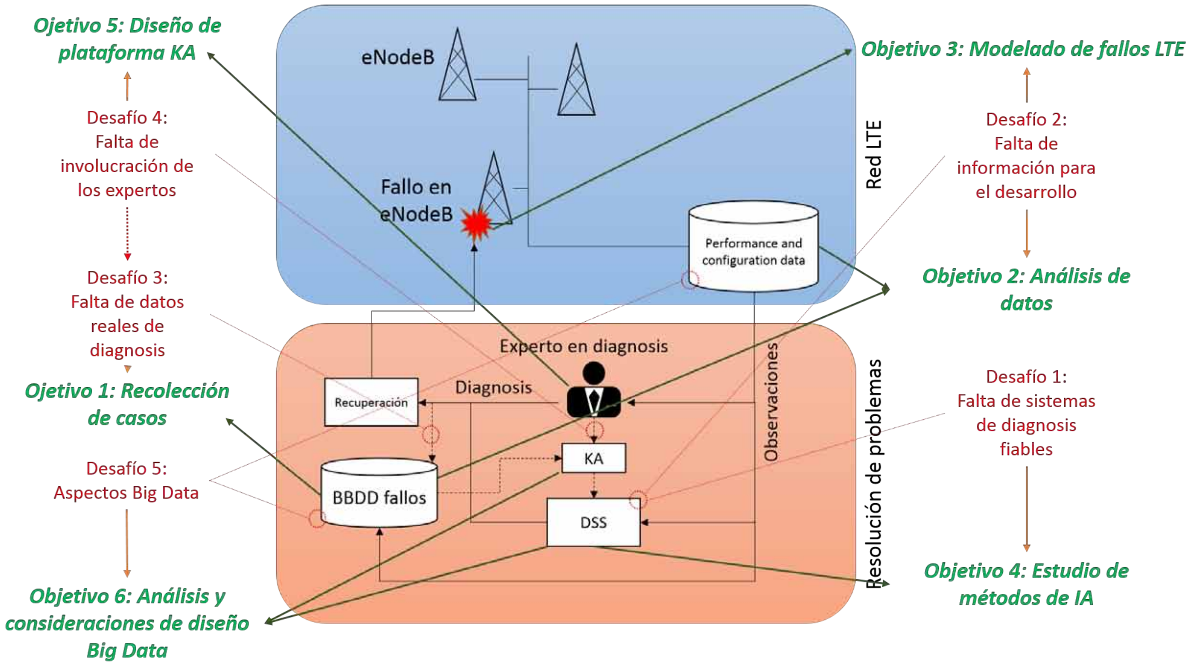

Ojetivo 5: Diseño de plataforma KA
Desafío 4: Falta de involucración de los expertos
Desafío 3: Falta de datos reales de diagnosis
Ojetivo 1: Recolección de casos
Desafío 5: Aspectos Big Data
Objetivo 6: Análisis y consideraciones de diseño Big Data
eNodeB
Fallo en eNodeB
Performance and configuration data
Red LTE
Experto en diagnosis
Diagnosis
Recuperación
KA
BBDD fallos
DSS
Observaciones
Resolución de problemas
Objetivo 3: Modelado de fallos LTE
Desafío 2: Falta de información para el desarrollo
Objetivo 2: Análisis de datos
Desafío 1: Falta de sistemas de diagnosis fiables
Objetivo 4: Estudio de métodos de IA

Figure D.1: Desafíos y objetivos principales

procesos, hay una cantidad significativa de estudios acerca de la detección [9][10][11][12], mientras que la diagnosis no ha recibido tanta atención a pesar de ser un componente clave de Self-healing. Esta tesis estudia el problema de la diagnosis automática, explorando la utilización de algoritmos de Inteligencia Artificial (IA) para realizar esta operación. Si bien hay algunos estudios acerca de este tema [13][14][15][16][17][18][19][20][21], en este momento la diagnosis automática no ha sido adoptada por el mercado dada la falta de técnicas y plataformas que puedan entrenar los algoritmos de IA de forma fácil y apropiada. Estos algoritmos deben ser entrenados normalmente por expertos en resolución de problemas, que no suelen tener disponibilidad para gastar el tiempo y realizar el esfuerzo necesarios. Por tanto, el principal objetivo de esta tesis es crear métodos para entrenar estos algoritmos para obtener resultados precisos con la mínima intervención necesaria por parte de expertos humanos.

D.2 Desafíos y objetivos

El objetivo principal de esta tesis es establecer las bases de un sistema que supera las limitaciones te impiden el desarrollo, prueba, despliegue y uso activo sistemas de diagnosis automática. La Figura D.1 muestra los principales objetivos de esta tesis en el escenario actual de la diagnosis automática.

Aunque la automatización de la resolución de problemas en LTE (específicamente diagnosis) es una necesidad para los operadores dadas las ganancias en tiempo, calidad de servicio y costes, aún no ha sido ampliamente adoptada. A pesar de que se han propuesto diversas aproximaciones basadas en algoritmos de IA [13][14][15][16][17][18][19][20][21], no han sido implementadas en herramientas comerciales. La causa de estas bajas tasas de adopción es la falta de algoritmos de diagnosis que funcionen bien y se adapten a las necesidades de los operadores en escenarios reales (**Desafío 1**). Por tanto, en la actualidad, la diagnosis de fallos es principalmente una tarea manual, que mantiene ocupados expertos humanos durante horas o días incrementando los costes. El valioso tiempo de los expertos se malgasta en la resolución de tareas repetitivas que podrían ser automatizadas de forma deficiente por sistemas de apoyo a la decisión (Decision Support Systems, DSS) basados en conocidas técnicas de IA. En esta tesis se utilizan controladores de lógica borrosa (Fuzzy Logic Controllers, FLC [22]) como el núcleo de un sistema DSS (**Objetivo 4**), por la facilidad de comprensión de su estructura y la relativa portabilidad de su base del conocimiento que facilita las tareas de importación, exportación e integración de conocimiento experto.

Generalmente, los DSS se crean siguiendo varios pasos: desarrollo (o selección entre las numerosas opciones disponibles [23][24][22][25][26][27]) de un núcleo basado en un algoritmo de IA que realiza el análisis de los datos disponibles (donde la falta de información acerca de las propiedades de datos procedentes de casos de resolución de problemas en escenarios reales limita el alcance y el realismo de las pruebas que se han hecho en estudios anteriores acerca de la diagnosis), entrenamiento de la solución sobre el sistema objetivo (que requiere datos reales para entrenar los algoritmos de IA o bien expertos que estén dispuestos a gastar el tiempo y realizar el esfuerzo para configurarlos de forma manual) y explotación a lo largo de la vida útil de la solución (incluyendo revisiones periódicas y actualizaciones del sistema que, una vez más, requieren datos reales o bien una configuración manual). A pesar de que hay muchas soluciones de IA disponibles y bien estudiadas, en el caso de la diagnosis, la selección del algoritmo específico está limitada por la falta de un análisis acerca de la naturaleza de los datos (**Desafío 2**). Por tanto, esta tesis estudia las propiedades de los datos que los expertos en resolución de problemas utilizan para hacer la diagnosis de forma manual (**Objetivo 2**) y de este modo poder hacer una selección bien fundada.

Los DSS necesitan disponer de una versión codificada del conocimiento experto que guíe sus capacidades de decisión. El formato con el que se codifica el conocimiento varía ampliamente de acuerdo al algoritmo IA utilizado. La integración de este conocimiento requiere de un proceso de adquisición del conocimiento (Knowledge Acquisition, KA), en el que la experiencia de los ingenieros de resolución de problemas es obtenida y guardada en un formato adaptado a los algoritmos de IA. Esto se puede hacer de dos maneras diferentes: bien mediante la intervención humana (mediante una entrevista [28][29][30][31] o instruyendo a los expertos acerca de cómo configurar el método de IA escogido, que suele ser un proceso tedioso que conduce a la falta de colaboración -**Desafío 4**- y desarrollo en el campo de la diagnosis automática -**Desafío 1**), o bien mediante KDD [32][33] (Knowledge Discovery and Datamining), que realiza un análisis estadístico de las entradas y salidas del trabajo de los expertos en busca de patrones repetitivos y los traduce al formato apropiado. KDD reduce significativamente el esfuerzo necesario en la fase de entrenamiento de los DSS, reduciendo las molestias en el flujo de trabajo de los ingenieros de resolución de problemas. No obstante, a pesar de la disponibilidad de los métodos KDD para el entrenamiento de sistemas de diagnosis automática [34][35][36][37], éstos aún no han visto un alto grado de adopción por parte de los operadores de red.

Un aspecto común de todos los pasos necesarios en la creación de una solución de diagnosis automática basada en DSS es la necesidad de datos representativos. Como se ha señalado anteriormente, el diseño (o selección) del algoritmo de IA que realiza las funciones de diagnosis necesita un conocimiento previo acerca del tipo de datos que será procesado. Esta información no se puede obtener a menos que haya datos de casos reales de resolución de problemas. La falta de dichos datos (**Desafío 3**) conduce a menudo al entrenamiento de los algoritmos con datos simulados, lo cual no produce resultados óptimos ni convincentes para los operadores de red. Otra solución típica es utilizar algoritmos de aprendizaje no supervisado, que en el caso de la diagnosis consisten en la búsqueda de patrones sin conocer el problema asociado. Esto conduce a sistemas de KA que requieren la intervención de los expertos para identificar y etiquetar los patrones reconocidos, haciendo que el sistema de nuevo dependa de la disponibilidad de dichos expertos. Hay una necesidad clara para casos reales de resolución de problemas, por lo que uno de los objetivos de esta tesis es crear una base de datos con estos casos (**Objetivo 1**). El volumen de datos disponibles en una red moderna para la resolución de problemas, así como la variedad de formatos en los que vienen puede suponer una carga de procesado excesiva para los métodos de computación tradicionales (**Desafío 5**), por lo que los métodos Big Data pueden ser necesarios.

Con un conjunto de datos de casos reales de resolución de problemas, la mayor parte de estas limitaciones se puede resolver. La falta de información acerca de la naturaleza de los datos puede resolverse mediante el análisis de este conjunto de datos, la búsqueda de las mejores técnicas para procesarlo (**Objetivo 2**, **Objetivo 6**) y la creación de un modelo (**Objetivo 3**) que identifique claramente el comportamiento de cada variable bajo diferentes situaciones. Un modelo de los datos también puede ayudar con el problema de la falta de datos, dado que puede utilizarse para mejorar y validar los resultados de los escenarios simulados, así como para generar nuevos conjuntos de datos que imiten fallos de red. Los objetivos concretos de esta tesis (Resumidos en la Figura D.1) son los siguientes:

Objetivo 1: **Recolección de una base de datos de fallos**: Si bien los datos acerca del rendimiento las redes son fácilmente accesibles en las bases de datos de los operadores, las causas de los fallos no suelen ser almacenadas o documentadas (**Desafío 3**) junto a los datos afectados dado que es una tarea que se sale del flujo de trabajo rutinario de los expertos (**Desafío 4**). El principal motivo de esto es que el proceso de recoger los datos manualmente, adjuntar un informe y guardarlos en una base de datos común es un proceso tedioso que aporta poco valor a corto plazo para los expertos en resolución de problemas. Por tanto, es necesario un sistema que pueda realizar esta tarea de forma rápida y con la mínima intervención por parte de los expertos. Si esta información se recoge de una herramienta que los expertos ya utilizan para inspeccionar los valores de los datos de rendimiento de la red que utilizan en el proceso diagnosis, el esfuerzo requerido es mínimo. En ese caso, el proceso de recolección de datos estaría integrado con las observaciones de los expertos, necesitando tan sólo que estos etiquetasen los datos con la diagnosis y dejando que el sistema haga el resto de las tareas de recolección, procesado y almacenamiento. En esta tesis, se desarrolla una plataforma software que requiere tan sólo tres variables (nombre del sector afectado, fecha en la que se diagnostica el problema y diagnosis) para realizar el proceso. Cuando este sistema se integra dentro de una herramienta de visualización de datos que los expertos utilizan de forma frecuente, dos de estas variables (nombre del sector afectado y fecha) pueden ser extraídas del contexto de utilización (es decir de los datos que se muestran en pantalla). Por tanto, los expertos tan sólo necesitan dar una diagnosis (respondiendo al **Desafío 4**). A continuación este sistema se utiliza para recoger un conjunto de casos reales de resolución de problemas. Esta base de datos contiene datos de rendimiento de una celda concreta afectada por un problema en una ventana de tiempo junto con una etiqueta que identifica el problema diagnosticado por los expertos. La existencia de esta base de datos resuelve el problema de la falta de datos reales (**Desafío 3**) qué impide el desarrollo de sistemas de diagnosis automática.

Objetivo 2: **Análisis de los datos recogidos**: Una vez resuelto el problema de la falta de datos, se puede atajar el problema de la falta de conocimiento acerca de la naturaleza de los datos (**Desafío 2**). Esta etapa del estudio incluye la caracterización de diferentes tipos de datos que han sido recogidos para diseñar mejor los algoritmos de IA que realizarán la diagnosis. Además, dado que los datos recogidos tendrán unas propiedades que no son compatibles con los requisitos de los algoritmos de minería de datos (Data Mining, DM) que entrenarán a los algoritmos de IA, una de las tareas principales de esta etapa será determinar el procesado requerido y el diseño de los métodos para realizarlo. De hecho, estos pasos de preprocesado son parte del proceso KDD. Esta etapa resuelve la falta de conocimiento y allana el camino para el diseño de algoritmos de IA mejores y el entrenamiento de los mismos con datos reales de la red LTE.

Objetivo 3: **Creación de modelos para los fallos más comunes en LTE**: con el doble propósito de generar nuevos datos de casos de fallo realistas y para tener un mejor conocimiento de los problemas bajo estudio, se generará un modelo de la base de datos recogida. Las variables de medida de rendimiento se caracterizarán individualmente usando modelos estadísticos condicionados a la ocurrencia de cada uno de los fallos más comunes. Este modelo podrá utilizarse además para validar simulaciones, dado que proporciona una fuente de datos realistas que puede servir de patrón de validación. El producto final de esta etapa ayuda a entender mejor el comportamiento de los datos (respondiendo al **Desafío 2**) y resuelve el problema de la falta de casos para las pruebas de algoritmos de IA al permitir la generación de datos realistas (**Desafío 3**).

Objetivo 4: **Estudio de métodos de IA para la diagnosis**: con el conocimiento disponible acerca de los datos de rendimiento de la red, los problemas más comunes y el proceso manual de resolución de problemas, el siguiente paso lógico será diseñar un algoritmo de IA que realice la diagnosis automática. En esta tesis, se utilizan los FLC [23][24] como mejor opción, dado que son fácilmente entendibles tanto por expertos humanos (dado que usan un lenguaje cercano al hablado) como por máquinas. Los FLC facilitan la generación auntomática de reglas de diagnosis y su integración con reglas preexistentes. Esta solución es también atractiva para los operadores de red, ya que su claridad hace que sea menos confusa que otras alternativas, como las redes bayesianas [25] o las redes neuronales [27].

Objetivo 5: **Diseño y pruebas de una plataforma de KA**: para obtener unos buenos resultados de diagnosis, los métodos de IA escogidos deben ser entrenados con datos de casos reales, de modo que tengan el conocimiento de problemas que se puedan encontrar una vez desplegados en una red real. Con este objetivo, esta tesis diseña un proceso KDD para entrenar un FLC, basado en la información extraída y los procesos desarrollados en la fase de análisis de la base de datos recogida. El producto de esta fase será un sistema software que, una vez proporcionada una base de datos de diagnosis de casos resueltos, devuelve un conjunto de reglas de diagnosis que pueden ser utilizados en un FLC. Estas reglas se adaptan a los escenarios reales donde serán utilizadas (respondiendo al **Desafío 1**). Además, este software podrá coordinarse con la plataforma de recolección de datos, de modo que cuando se introducen nuevos casos en la base de datos, se mejora el modelo de conocimiento del FLC asociado.

Objetivo 6: **Análisis de los aspectos Big Data y consideraciones de diseño**: en las redes modernas, especialmente en LTE, la cantidad de datos de rendimiento y configuración es enorme. Estos datos se generan en un amplio abanico de fuentes de datos (tales como los terminales de usuario, los nodos de acceso, etc.) y formatos (diferentes tipos de fichero, resoluciones temporales, etc.), lo que incrementa el número de pasos de procesado necesarios. Además, dado el elevado número de eventos que suceden en la red por unidad de tiempo, la información se genera de forma continua a una alta velocidad. Todos estos aspectos (**Desafío 5**) son parte del paradigma Big Data [40]. En los problemas Big Data, las técnicas tradicionales de procesado no son lo suficientemente potentes (es decir, no son capaces de procesar la cantidad de trabajo requerida en el tiempo disponible), por lo que se utilizan técnicas y plataformas nuevas. Esta tesis estudiará los aspectos Big Data de los datos extraídos de la red, y los tendrá en cuenta en todo momento en el diseño de los algoritmos utilizados para tratar dichos datos.

La combinación de estas partes constituye la visión general del sistema propuesto en esta tesis: una plataforma en la que cada vez que un experto diagnostica un problema en la red, éste puede reportarlo con un esfuerzo mínimo y almacenarlo en el sistema. La parte central de este sistema es un algoritmo de diagnosis que evoluciona y mejora aprendiendo de cada nuevo ejemplo, hasta llegar al punto en el que los expertos pueden confiar en su precisión para los problemas más comunes. Cada vez que surja un nuevo problema, se añadirá a la base de datos del sistema, incrementando así aún más su potencia. El fin es liberar a los expertos de tareas repetitivas, de modo que puedan dedicar su tiempo a desafíos cuya resolución sea más gratificante. Además, los resultados intermedios del sistema (modelos de los problemas más comunes, resultados del preprocesado de datos, etc.) pueden ser utilizados como información de apoyo al proceso manual de diagnosis.

A continuación se hará un breve resumen de los capítulos 2 a 6 de la tesis.

D.3 Resolución automática de problemas en LTE

En el capítulo 2 se describen las tecnologías de red que forman la base de la tesis. Concretamente, se hace una introducción de la red LTE, comentando cómo el mercado demandaba anchos de banda cada vez mayores y cómo esto condujo a las especificaciones de 4G, que se vieron plasmadas en el protocolo LTE. A continuación se describe el proceso de estandarización de LTE. En la segunda parte del capítulo se describen las redes SON, y las funcionalidades de autoconfiguración, autooptimización y autocuración. Se describen aspectos de la implementación de dichas funcionalidades, estableciendo la relación con los algoritmos de IA. Finalmente, se realiza una descripción más detallada de la función de autocuración, describiendo en primer lugar el proceso manual de resolución de problemas, y a continuación el estado del arte en tecnologías que automaticen dicha funcionalidad.

D.4 Inteligencia Artificial

El capítulo 3 introduce los conceptos de IA utilizados en la tesis para implementar la solución de KA. Este capítulo se divide en cuatro partes. En la primera parte de describen los sistemas DSS. Se introduce el concepto de conocimiento experto de manera formal, y su papel en los sistemas basados en el conocimiento (Knowledge-Based System, KBS). Se exploran asimismo las soluciones basadas en KBS desarrolladas con anterioridad para la diagnosis de fallos, y se describe en detalle la utilización de los controladores de lógica borrosa (FLC). A continuación, se introduce con mayor detalle el problema de la adquisición del conocimiento, revisando las metodologías manuales utilizadas tradicionalmente y los problemas que causan en el escenario de la diagnosis en redes móviles. En la tercera parte, se introducen las técnicas KDD, describiendo este campo de estudio y las condiciones de contorno para su aplicación en el escenario de esta tesis. Finalmente, se explora el campo de las tecnologías Big Data, estableciendo las condiciones que separan un problema tradicional de analítica de datos de un problema en el que es necesario utilizar estas técnicas.

D.5 Adquisición del conocimiento para sistemas de diagnosis en redes LTE

En el capítulo 4 se explora el proceso de recogida de datos de casos reales de resolución de problemas en la red LTE. En primer lugar, se describen las propiedades de los datos disponibles para la diagnosis, especificando las diferentes fuentes de datos y formatos. Se exploran los problemas de dimensionalidad, que hacen que se la adquisición del conocimiento en este escenario sea un problema Big Data. Para ejemplificar la utilización de técnicas Big Data en redes LTE, se exponen varios casos de uso en los que se modifican técnicas de autocuración para que sean utilizadas en plataformas de computación en la nube. En la segunda parte del capítulo 4 se describe el problema de la adquisición del conocimiento, estableciendo su separación en dos partes: captura y modelado. La parte de captura del conocimiento comprende los procesos necesarios para obtener los datos que contienen el conocimiento experto, mientras que el modelado se encarga de extraer ese conocimiento y convertirlo a un formato que pueda ser utilizado por un KBS. Esta parte del capítulo revisa los requisitos necesarios para la creación de un sistema de captura del conocimiento, tanto desde el punto de vista técnico (de cara a las propiedades de los datos) como desde el punto de vista humano (de cara a las necesidades y motivaciones de los expertos en diagnosis). Finalmente, se describe la interfaz software desarrollada para realizar la captura del conocimiento.

D.6 Modelado de bases de datos de problemas en LTE

El capítulo 5 se centra en el análisis de la base de datos de casos problemáticos recogida durante la realización del estudio. Este capítulo se divide en tres partes. En la primera se describe la base de datos en crudo, describiendo los distintos problemas recogidos y sus proporciones, así como otra información acerca del proceso de captura del conocimiento. A continuación se describen los procesos necesarios para el preprocesado de los datos recogidos y su conversión a un formato apropiado para los procesos de modelado descritos más adelante y de minería de datos descritos en el capítulo 6. Finalmente, se describe un proceso de modelado de datos y se aplica sobre los datos descritos, mostrando las relaciones entre los problemas y los principales indicadores de rendimiento recogidos.

D.7 Minería de datos en LTE

En el capítulo 6 se describe el proceso que extrae las reglas de diagnosis a partir de los datos de casos de fallo apropiadamente formateados. En primer lugar, se introducen las nociones de minería de datos necesarias, así como un repaso de la taxonomía de los algoritmos disponibles. A continuación se describen en detalle dos algoritmos de minería de datos diseñados para la extracción de las reglas de diagnosis (un algoritmo genético y uno basado en el algoritmo WM [96]) y se realizan pruebas con datos simulados para caracterizar su comportamiento. A la luz de los resultados obtenidos, se concluye que el algoritmo basado en WM es más apropiado dada su posibilidad de ser ejecutado en la nube. Finalmente, se aplica el algoritmo elegido sobre los datos reales de la red descritos en el capítulo 5.

D.8 Conclusiones

En esta sección se revisarán los resultados de la tesis, las contribuciones y las líneas de trabajo futuro.

D.8.1 Resultados

Como resultado del estudio de la tesis se obtienen varios resultados asociados a la red bajo estudio. Estos resultados pueden obtenerse con facilidad en otras redes aplicando los pasos descritos en la metodología. Por tanto, aunque estos resultados no representen siempre el comportamiento general de las redes LTE, su comprensión en el contexto de un escenario conocido arroja luz para los desarrollos futuros. Concretamente, los resultados obtenidos son los siguientes:

- **Base de datos de casos de diagnosis resueltos**: La base de datos recogida en el capítulo 5 es el primer paso en el proceso KDD. Es la salida del proceso de selección (descrito en el capítulo 4), por lo que es el primer subproducto que contiene conocimiento experto. La base de datos de casos resueltos debe cumplir una serie de requisitos, tales como tener el suficiente número de ejemplares de cada caso, un buen número de casos distintos, etc. El cumplimiento de estos requisitos depende en gran medida de la calidad del trabajo de los expertos (es decir, que cometan pocos fallos en el proceso de diagnosis). La base de datos extraída cubre un período de 10 meses (desde Junio de 2013 hasta Abril de 2014). En este intervalo, se han recogido un total de 475 casos clasificados en 21 tipos de problemas distintos. No obstante, de estos casos, sólo un subconjunto con los datos de mayor calidad fue seleccionado para el proceso de KDD, con lo el tamaño final de la base de datos procesada fue de 47 casos divididos en 4 problemas.
- **Preprocesado de los datos de fallos**: La preparación de los datos para el proceso de minería de datos es un paso indispensable. Sin este paso, los patrones buscados por el algoritmo de minería de datos estarían escondidos entre el ruido. En el caso de los datos de fallos de red LTE, la preparación consiste en limpiar (rellenar los valores perdidos en las series temporales) y reducción de la dimensionalidad (eliminando la componente temporal de los datos). El resultado de la fase de preprocesado es un conjunto de datos que contiene vectores con valores de indicadores de rendimiento degradados. Dichos vectores representan un intervalo temporal en el que la red no presta servicio con una calidad aceptable y vienen etiquetados con una diagnosis dada por los expertos en resolución de problemas. El resultado de la aplicación de los algoritmos de preprocesado sobre la base de datos seleccionada fue un conjunto de 359 vectores.
- **Modelado de fallos de LTE**: El siguiente paso en el estudio de la base de datos de casos resueltos fue el modelado de la misma. El modelo extraído representa las relaciones entre lso problemas seleccionados y un subconjunto de indicadores de rendimiento. Este proceso constituye en sí mismo un proceso de minería de datos, aunque no forma parte del sistema de KA, dado que no está orientado a la extracción de un sistema de diagnosis. En esta tesis, se ha utilizado el modelo para mejorar los resultados al lograr el balanceo de datos (es decir, que cada problema tenga un número similar de vectores que lo representen en el proceso de minería de datos). El resultado del proceso de modelado es un conjunto de funciones distribución de probabilidad para cada indicador de rendimiento condicionadas a la ocurrencia de un problema. El modelo resultante se muestra en el apéndice C.
- **Reglas de diagnosis**: El paso principal del proceso de adquisición del conocimiento es la minería de datos. Con los datos limpios y reducidos, la fase de minería de datos extrae los parámetros de un FLC que se utiliza para la diagnosis. Dado que los parámetros de los FLC se dividen en dos partes (funciones de pertenencia y reglas), se utilizan dos algoritmos de minería de datos en este paso. El FLC resultante contiene información de las funciones de pertenencia adaptadas a la red en la que se despliega y reglas de diagnosis basadas en el conocimiento experto. Los resultados de esta etapa se probaron sobre los datos originales mostrando un alto índice de aciertos.

D.8.2 Contribuciones

La contribución general de esta tesis es el desarrollo de una metodología para la adquisición del conocimiento en el proceso de resolución de problemas en redes LTE (**Objetivo 5**). El producto de esta metodología es un sistema de diagnosis automática basado en un FLC. Esto reducirá el trabajo repetitivo de los expertos, que tendrán de este modo más tiempo para emplearlo en problemas más complejos y menos susceptibles de ser automatizados. Además, el sistema de diagnosis automática reducirá el tiempo requerido para resolver problemas, reduciendo así los cortes de servicio y mejorando la experiencia de usuario. De forma más pormenorizada, las contribuciones de esta tesis son las siguientes:

- **Base de datos de casos de diagnosis resueltos**: La base de datos constituye en sí misma una importante contribución (**Objetivo 1**) que puede ser utilizada como herramienta para un elevado número de aplicaciones relacionadas con la resolución de problemas: formación de expertos en diagnosis, planificación para mejoras y optimización de la red, análisis de la evolución de la red, etc. En el desarrollo de funciones SON, la existencia de esta base de datos responde a una necesidad que no está satisfecha; los expertos en diagnosis no suelen tener tiempo para crear bases de datos de este tipo, y los desarrolladores no suelen tener los conocimientos necesarios. Como consecuancia, tradicionalmente, los sistemas SON se desarrollan sin datos reales, dando lugar a pruebas poco realistas.
- **Algoritmos de preprocesado**: Esta tesis propone varios algoritmos de preprocesado para el tratamiento de los datos recogidos. Para ello, previamente se ha hecho un análisis de las propiedades de los mismos (**Objective 2**). Se ha propuesto un algoritmo de limpieza de datos que utiliza valores históricos y las propiedades de periodicidad para rellenar los huecos, y un algoritmo de reducción de datos que extrae los valores de los indicadores de rendimiento en condiciones de degradación.
- **Modelo de fallos de LTE**: El modelo de los fallos de red obtenido en esta tesis (**Objetivo 3**) constituye una valiosa aportación que ayuda a la comprensión de las relaciones entre las medidas de rendimiento y los fallos modelados. La información aportada por el modelo puede conducir a mejoras en la red y en el diseño de algoritmos SON. Además, el modelo puede ser utilizado para generar datos emulados con un alto realismo. El procedimiento utilizado para extraer el modelo también puede ser utilizado para modelar nuevos problemas y en redes distintas.
- **Algoritmos de minería de datos**: Otra contribución de esta tesis es el estudio de la aplicación de FLCs a la diagnosis (**Objetivo 4**) y el diseño e implementación de dos algoritmos de minería de datos que ajustan un FLC basándose en una base de datos de casos de diagnosis. El FLC extraído consigue una alta tasa de aciertos en la diagnosis de la red en pruebas; y el método utilizado para ajustarlo puede ser utilizado en otras redes. Además de la utilidad para la diagnosis, las reglas son una buena fuente de información acerca de la naturaleza de los problemas observados, ya que vienen dadas en un lenguaje próximo al humano.
- **Adquisición del conocimiento para la autocuración en LTE**: Esta tesis ha estudiado el problema de la adquisición del conocimiento, estableciendo las ventajas de utilizar KDD para este fin. Se ha enumerado los requisitos de una plataforma de adquisición del

conocimiento, tanto desde el punto de vista técnico como el humano. Además, se ha diseñado y desplegado una plataforma basada en estos requisitos (**Objetivo 5**) que es capaz de recoger y procesar los datos con tan sólo la fecha, el sector afectado y la diagnosis como entrada.

- **Estudio de los aspectos Big Data de la autocuración**: El uso de KDD en la autocuración está sujeto a las características de los datos de rendimiento de la red. Esta tesis ha estudiado las características de estos datos (**Objetivo 2**), determinando que dado su volumen, variabilidad y velocidad de generación, pueden considerarse como un problema Big Data. Por tanto, la naturaleza Big Data del problema ha sido estudiada con profundidad (**Objetivo 6**). Los aspectos Big Data de los datos se han tenido en cuenta a lo largo del estudio en el diseño de los algoritmos, asegurando que puedan ejecutarse de modo paralelo y ejecutarse en una nube de computación. De forma adicional, se ha propuesto el rediseño de varios algoritmos publicados, adaptándolos a la posibilidad de la ejecución en modo paralelo.

Además, se identifican varias líneas de trabajo futuro que podrán ampliar estos resultados y aportaciones:

- Mejoras en la interfaz diseñada para la captura del conocimiento, simplificando la interacción con los expertos y además dándoles resultados parciales que puedan servir de ayuda en el proceso de diagnosis e incentiven el uso de la plataforma.
- Integración de la plataforma en un paquete software cerrado que realice todos los pasos de forma consecutiva. Dicha solución iuntegrada deberá ser además fácil de desplegar y comenzar a utilizar.
- Recolección de un mayor número de casos problemáticos, con nuevas causas y en distintas redes.
- Selección automática de los indicadores de rendimiento relevantes.
- Desarrollo de aprendizaje online que mejore la calidad del sistema de diagnosis de forma automática cada vez que se introduzca un caso nuevo.
- Mejoras y optimizaciones sobre los algoritmos desarrollados.

D.9 Publicaciones y proyectos

D.9.1 Revistas

	Publicaciones derivadas de esta tesis	FI	Ranking
I	E. J. Khatib, R. Barco, P. Muñoz, I. de-la-Bandera, I. Serrano, "Self-healing in mobile networks with big data". IEEE Communications Magazine, vol. 54, no. 1, pp. 114-120, Jan. 2016.	5.125	Q1 (2/82) Telecomunicaciones

II	E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, I. Serrano, "Data mining for fuzzy diagnosis systems in LTE networks". Expert Systems with Applications, vol. 42, no. 21, pp 7549-7559, Nov. 2015.	2.981	Q1 (19/130) Informática, Inteligencia Artificial
III	E. J. Khatib, R. Barco, A. Gómez-Andrades, I. Serrano, "Diagnosis based on genetic fuzzy algorithms for LTE Self-Healing". IEEE Transactions on Vehicular Technology, vol. 65, no. 3, Mar. 2016.	2.243	Q1 (14/82) Telecomunicaciones
IV	E. J. Khatib, A. Gómez-Andrades, I. Serrano, R. Barco, "Modelling LTE solved troubleshooting cases", *Journal of Network and Systems Management*, Under review.	1.078	Q3 (77/143) Informática, Sistemas de Información
V	E. J. Khatib, R. Barco, P. Muñoz, I. Serrano, "Knowledge Acquisition for Fault Management in LTE Networks", *Wireless Personal Communications*, Under review.	0.701	Q4 (63/82) Telecomunicaciones
VI	E. J. Khatib, R. Barco, I. Serrano, "Degradation Detection Algorithm for LTE Root Cause Analysis", *Wireless Personal Communications*, Under review.	0.701	Q4 (63/82) Telecomunicaciones

	Publicaciones relacionadas con esta tesis	FI	Ranking
VII	A. Gómez-Andrades, P. Muñoz, E. J. Khatib, I. de-la-Bandera, I. Serrano, R. Barco, "Methodology for the Design and Evaluation of Self-Healing LTE Networks", *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6468-6486, Aug. 2016.	2.243	Q1 (14/82) Telecomunicaciones
VIII	P. Muñoz, I. de-la-Bandera, E. J. Khatib, A. Gómez-Andrades, I. Serrano, R. Barco, "Root Cause Analysis based on Temporal Analysis of Metrics toward Self-Organizing 5G Networks", *IEEE Transactions on Vehicular Technology*, Online 2016.	2.243	Q1 (14/82) Telecomunicaciones

IX	P. Muñoz, R. Barco, E. Cruz, A. Gómez-Andrades, E. J. Khatib, N. Faour, "A method for identifying faulty cells using a classification tree-based UE diagnosis in LTE", *EURASIP Journal on Wireless Communications and Networking*, Under review.	0.627	Q3 (191/255) Ingeniería Eléctrica & Electrónica

D.9.2 Patentes

	Patentes relacionadas con esta tesis
X	P. Muñoz, R. Barco, I. Serrano, I. de-la-Bandera, E. J. Khatib. "Fault diagnosis in Networks". Nº of application: PCT/EP2015/058924 (24 April 2015). Nº of international publication: WO/2016/169616 (27 October 2016)

D.9.3 Conferencias

	Publicaciones derivadas de esta tesis
XI	E. J. Khatib, R. Barco, I. Serrano, P. Muñoz, "LTE performance data reduction for knowledge acquisition". GLOBECOM 2014, Austin.
XII	E. J. Khatib, R. Barco, I. Serrano, "Captura del conocimiento para el modelado de fallos en redes LTE". XXIV Simposium nacional de la Unión Científica Internacional de Radio, Valencia 2014.
XIII	E. J. Khatib, R. Barco, A. Gómez-Andrades, "Diagnosis en LTE Self-Optimizing Networks basada en algoritmos genéticos". XXIII Simposium nacional de la Unión Científica Internacional de Radio, Santiago de Compostela 2013.

D.9.4 Proyectos relacionados

Esta tesis ha sido parcialmente financiada por:

- Optimi-Ericsson, ref. 59288, Junta de Andalucía (Agencia IDEA, Consejería de Ciencia, Innovación y Empresa) and ERDF.
- Proyecto de Investigación de Excelencia P12-TIC-2905, Junta de Andalucía.

D.9.5 Estancias

Adicionalmente, en el estudio de esta tesis se han realizado tres estancias en el extranjero:

- Centros de Ericsson y AT&T en Los Angeles, California (EEUU), colaborando con expertos en resolución de problemas en su trabajo rutinario para observar el proceso de diagnosis de fallos, las herramientas y los requisitos para la plataforma de adquisición del conocimiento.
- Centro de Ericsson en Plano, Texas (EEUU), colaborando con expertos en resolución de problemas en el desarrollo de una solución de diagnosis automática basada en reglas.

- Centro de Nokia en Aalborg (Dinamarca), colaborando con ingenieros de investigación y desarrollo en comunicaciones tipo máquina y tecnologías 5G.

Bibliography

[1] 3GPP. *Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements (Rel 11), TS 32.541*, September 2012.

[2] Z. Altman, R. Skehill, R. Barco, L. Moltsen, R. Brennan, A. Samhat, R. Khanafer, H. Dubreil, M. Barry, and B. Solana. The Celtic Gandalf framework. In *MELECON 2006 - 2006 IEEE Mediterranean Electrotechnical Conference*, pages 595–598, May 2006.

[3] ICT-2007-216248 E3. Project presentation report. Technical report deliverable D0.2, version 1.0, May 2008.

[4] INFSO-ICT-216284 SOCRATES. Use cases for Self-Organising Networks. Technical report deliverable D2.1, version 1.0, March 2008.

[5] INFSO-ICT-224344 Self-NET. System deployment scenarios and use cases for cognitive management of future internet elements. Technical report deliverable D1.1, version 1.0, October 2008.

[6] FP7-257513 UniverSelf. Self-diagnosis and self-healing for IMS VoIP and VPN services technical report case study - part I, version 1.0, September 2012.

[7] INFSO-ICT-316384 SEMAFOUR. Self-Management for Unified Heterogeneous Radio Access Networks. Technical report deliverable D6.1, version 1.0, October 2012.

[8] CP08-004 COMMUNE. Outline requirements for COMMUNE. Technical report deliverable. D2.1, version 1.0, April 2012.

[9] Guilherme A Barreto, Joao CM Mota, Luıs GM Souza, Rewbenio A Frota, Leonardo Aguayo, José S Yamamoto, and Pedro EO Macedo. A new approach to fault detection and diagnosis in cellular systems using competitive learning. In *Proceedings of the VII Brazilian Symposium on Neural Networks (SBRN04)*, 2004.

[10] Yu Ma, Mugen Peng, Wenqian Xue, and Xiaodong Ji. A dynamic affinity propagation clustering algorithm for cell outage detection in self-healing networks. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2266–2270. IEEE, 2013.

[11] S. Chernov, M. Cochez, and T. Ristaniemi. Anomaly detection algorithms for the sleeping cell detection in LTE networks. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2015.

[12] P. Szilágyi and C. Vulkán. LTE user plane congestion detection and analysis. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International*

Symposium on, pages 1819–1824, Aug 2015.

[13] R. Barco, L. Díez, V. Wille, and P. Lázaro. Automatic diagnosis of mobile communication networks under imprecise parameters. *Expert Systems With Applications.*, Vol.36 (1):489–500, January 2009.

[14] L. Bennacer, L. Ciavaglia, A. Chibani, Y. Amirat, and A. Mellouk. Optimization of fault diagnosis based on the combination of bayesian networks and case-based reasoning. In *IEEE Network Operations and Management Symposium (NOMS)*, April 2012.

[15] P. Szilagyi and S. Novaczki. An automatic detection and diagnosis framework for mobile communication systems. *IEEE Transactions on Network and Service Management*, 9, no.2:184–197, June 2012.

[16] G. A. Barreto, J. C. M. Mota, L. G. M. Souza, R. A. Frota, and L. Aguayo. Condition monitoring of 3G cellular networks through competitive neural models. *IEEE Trans. Neural Networks*, Vol.16(5):1064–1075, 2005.

[17] R. Barco, P. Lázaro, L. Díez, and V. Wille. Continuous versus discrete model in auto-diagnosis systems for wireless networks. *IEEE Transactions on Mobile Computing*, Vol.7 (6):673–681, June 2008.

[18] Raquel Barco, Volker Wille, and Luis Díez. System for automated diagnosis in cellular networks based on performance indicators. *European Transactions on Telecommunications*, 16(5):399–409, 2005.

[19] Raquel Barco, Volker Wille, Luis Díez, and Matías Toril. Learning of model parameters for fault diagnosis in wireless networks. *Wireless Networks*, 16(1):255–271, 2010.

[20] Rana M Khanafer, Beatriz Solana, Jordi Triola, Raquel Barco, Lars Moltsen, Zwi Altman, and Pedro Lazaro. Automated diagnosis for UMTS networks using bayesian network approach. *IEEE Transactions on Vehicular Technology*, 57(4):2451–2461, 2008.

[21] Szabolcs Nováczki. An improved anomaly detection and diagnosis framework for mobile network operators. In *Design of reliable communication networks (DRCN), 2013 9th international conference on the*, pages 234–241. IEEE, 2013.

[22] CC. Lee. Fuzzy logic in control systems: fuzzy logic controller. *IEEE Transactions on I. Systems, Man and Cybernetics*, pages 404–418, 1990.

[23] Lotfi A. Zadeh. Fuzzy sets. *Information and control*, 1965.

[24] Hamid R Berenji. Fuzzy logic controllers. In *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, pages 69–96. Springer, 1992.

[25] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs.* Springer Science & Business Media, 2009.

[26] Yufei Yuan and Michael J Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and systems*, 69(2):125–139, 1995.

[27] Simon Haykin and Neural Network. Neural networks, a comprehensive foundation. 2004.

[28] A. Hart. *Knowledge Acquisition for Expert Systems. Second Edition.* Artificial Intelligence Systems, McGraw-Hill, 1992.

[29] R. Maier. *Knowledge Management Systems: Information and Communication Technologies for Knowledge Management.* Springer, 2007.

[30] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1):161–197, 1998.

[31] R. Barco, P. Lázaro, V. Wille, L. Díez, and S. Patel. Knowledge acquisition for diagnosis model in wireless networks. *Expert Systems With Applications.*, Vol.36, Issue 3, Part 1:4745–4752, April 2009.

[32] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 2003.

[33] E. Triantaphyllou and G. Felici. *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques.* Springer, 2006.

[34] C. Hounkonnou and E. Fabre. Empowering self-diagnosis with self-modeling. In *8th international conference Network and service management (CNSM), and 2012 workshop on systems virtualization management (SVM)*, October 2012.

[35] Alexander B Trunov and Marios M Polycarpou. Automated fault diagnosis in nonlinear multivariable systems using a learning methodology. *IEEE Transactions on neural networks*, 11(1):91–101, 2000.

[36] Hamid Nejjari and Mohamed El Hachemi Benbouzid. Monitoring and diagnosis of induction motors electrical faults using a current Park's vector pattern learning approach. *IEEE Transactions on Industry Applications*, 36(3):730–735, 2000.

[37] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[38] Moazzam Islam Tiwana and Mohsin Islam Tiwana. A novel framework of automated RRM for LTE SON using data mining: Application to LTE mobility. *Journal of Network and Systems Management*, 22(2):235–258, 2014.

[39] S. Chernov, F. Chernogorov, D. Petrov, and T. Ristaniemi. Data mining framework for random access failure detection in LTE networks. In *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pages 1321–1326, Sept 2014.

[40] P. Russom. Big Data analytics. *TDWI Best Practices Report, Fourth Quarter*, 2011.

[41] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2.* Next Generation Mobile Networks (NGMN) Alliance, ts 36.300 edition, December 2012.

[42] 3GPP. *Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN), version 8.0.0, TR 25.913*, January 2009.

[43] NGMN. *Use Cases Related to Self-Organising Network, Overall Description.* Next Generation Mobile Networks Alliance (NGMN), http://www.ngmn.org, April 2007.

[44] 3GPP. *Self-Organizing Networks (SON); Concepts and requirements, TS 32.500*, December 2011.

[45] S. Hamalainen, H. Sanneck, and C. Sartori. *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency.* Wiley, 2011.

[46] 3GPP. *Self-configuring and self-optimizing network use cases and solutions, version 9.0.0, TR 36.902*, September 2012.

[47] P. Muñoz, R. Barco, and I. de la Bandera. On the potential of handover parameter optimization for Self-Organizing Networks. *IEEE Transactions on Vehicular Technology*, 62, no.5:1895–1905, June 2013.

[48] P. Muñoz, R. Barco, and I. de la Bandera. Optimization of load balancing using fuzzy Q-learning for next generation wireless networks. *Expert Systems With Applications*, 40, no. 4:984–994, March 2013.

[49] M. Amirijoo, L. Jorguseski, T. Kürner, R. Litjens, M. Neuland, L. C. Schmelz, and U. Türke. Cell outage management in LTE networks. In *6th International Symposium on Wireless Communication Systems (ISWCS)*, 2009.

[50] H. Eckhardt, S. Klein, , and M. Gruber. Vertical antenna tilt optimization for LTE base stations. In *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011.

[51] R. Razavi. Self-optimisation of antenna beam tilting in LTE networks. In *IEEE 75th Vehicular Technology Conference (VTC Spring)*, 2012.

[52] Ana Gómez-Andrades, Raquel Barco, Immaculada Serrano, Patricia Delgado, Patricia Caro-Oliver, and Pablo Muñoz. Automatic root cause analysis based on traces for LTE Self-Organizing Networks. *IEEE Wireless Communications*, 23(3):20–28, 2016.

[53] J. M. Ruiz-Aviles, S. Luna-Ramírez, M. Toril, and F. Ruiz. Fuzzy logic controllers for traffic sharing in enterprise LTE femtocells. In *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, pages 1–5. IEEE, 2012.

[54] W-C. Chung, C-J. Chang, and L-C. Wang. An intelligent priority resource allocation scheme for LTE-A downlink systems. *Wireless Communications Letters, IEEE*, 1(3):241–244, 2012.

[55] M.Z. Asghar, S. Hamalainen, and T. Ristaniemi. Self-healing framework for LTE networks. In *IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, September 2012.

[56] I. de la Bandera, R. Barco, P. Muñoz, and I. Serrano. Cell outage detection based on handover statistics. *Communications Letters, IEEE*, 19(7):1189–1192, 2015.

[57] P. Muñoz, R. Barco, I. Serrano, and A. Gómez-Andrades. Correlation-based time-series analysis for cell degradation detection in SON. *IEEE Communications Letters*, 20(2):396–399, Feb 2016.

[58] S. Fortes, R. Barco, and A. Aguilar-Garcia. Location-based distributed sleeping cell detection and root caus analysis for 5G ultra-dense networks. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):149, 2016.

[59] A. Gómez-Andrades, P. Muñoz, I. Serrano, and R. Barco. Automatic root cause analysis for LTE networks based on unsupervised techniques. *IEEE Transactions on Vehicular Technology*, 65(4):2369–2386, April 2016.

[60] COMMUNE Project. Specification of knowledge-based reasoning algorithms, 2012. Deliverable 4.1.

[61] Zhengxin Jiang, Peng Yu, Yulin Su, W. Li, and X. Qiu. A cell outage compensation scheme based on immune algorithm in LTE networks. In *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*, pages 1–6, Sept 2013.

[62] L. Kayili and E. Sousa. Cell outage compensation for irregular cellular networks. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1850–1855,

April 2014.

[63] O. Onireti, A. Zoha, J. Moysen, A. Imran, L. Giupponi, M. Ali Imran, and A. Abu-Dayya. A cell outage management framework for dense heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 65(4):2097–2113, April 2016.

[64] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, and H. Efendic. Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. *Information Sciences*, 259(0):304 – 320, 2014.

[65] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, M. Pichler, and H. Efendic. Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. *Information Fusion*, 20(0):272 – 291, 2014.

[66] C. Guardiola E. Lughofer. On-line fault detection with data-driven evolving fuzzy models. *Control and intelligent systems*, 36(4):307, 2008.

[67] P. R. Innocent, R. I. John, and J. M. Garibaldi. Fuzzy methods for medical diagnosis. *Applied Artificial Intelligence*, 19(1):69–98, 2004.

[68] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving.* Addison-Wesley Pub. Co., Inc., Reading, MA, 1984.

[69] R. Akerkar and P. Sajja. *Knowledge-based systems.* Jones & Bartlett Publishers, 2010.

[70] A.J. Gonzalez and D. D. Dankel. *The Engineering of Knowledge-Based Systems, Theory and Practice*, chapter Chapter 1, Section 1.5. Prentice Hall, 1993.

[71] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 2014.

[72] Janet Kolodner. *Case-based reasoning.* Morgan Kaufmann, 2014.

[73] Guus Schreiber. *Knowledge engineering and management: the CommonKADS methodology.* MIT press, 2000.

[74] Marwan Awad Ahmed. Hybrid knowledge acquisition framework in mobile network.

[75] A. Hanemann. A hybrid rule-based/case-based reasoning approach for service fault diagnosis. In *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, volume 2, pages 5 pp.–, April 2006.

[76] J. F. Junior, A. Pereira, W. M. Junior, and A. Veloso. A KDD-based methodology to rank trust in e-commerce systems. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 577–584, Nov 2013.

[77] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602 – 613, 2011.

[78] Nan-Chen Hsieh. An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems With Applications*, 27(4):623–633, 2004.

[79] Charu C Aggarwal. *Managing and mining sensor data.* Springer Science & Business Media, 2013.

[80] Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges. *Sensors*, 13(12):17472–

17500, 2013.

[81] Nathan Marz and James Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[82] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[83] R. Cattell. Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[84] Django web framework. `https://www.djangoproject.com/`.

[85] Python programming language. `http://www.python.org/`.

[86] Celery distributed task queue. `http://www.celeryproject.org/`.

[87] Glenn E Krasner, Stephen T Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[88] SQLAlchemy database toolkit for Python. `http://www.sqlalchemy.org/`.

[89] Pandas data analysis library. `http://pandas.pydata.org/`.

[90] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *Computer.*, 36(10):38–44, 2003.

[91] F. J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[92] SciPy scientific computing tools for Python. `http://scipy.org/`.

[93] SciPy documentation. `http://scipy.org/docs.html`.

[94] O. Cordon, F. Herrera, and P. Villar. Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems*, 9(4):667–674, August 2001.

[95] A. Gonzalez and R. Pérez. SLAVE: A genetic learning system based on an iterative approach. *Fuzzy Systems, IEEE Transactions on*, 7(2):176–191, April 1999.

[96] L-X. Wang and Jerry M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics*, 22.6:1414–1427, 1992.

[97] GeNIe modeling environment. `http://genie.sis.pitt.edu/`, 2014.

[98] U. Fayyad and K. B. Irani. Multi-interval discretization of continuous valued attributes for classification learning. *Proc. International Joint Conference on Artificial Intelligence*, pages 1022–1027, August 1993.