Self-Healing in Mobile Networks with Big Data

E. J. Khatib*, R. Barco*, P. Muñoz*, I de la Bandera*, I. Serrano†

* Universidad de Málaga, Communications Engineering Dept., Málaga, Spain

† Ericsson, PBO RA Continuous Analysis

***Abstract*—Mobile Networks have rapidly evolved in the last years due to the increase in multimedia traffic and in the offered services. This leads to a growth in the volume of control data and measurements that are used by Self-Healing systems. To maintain a certain Quality of Service (QoS), Self-Healing systems must complete their tasks in a reasonable time. The conjunction of a big volume of data and the limitation of time requires a Big Data approach to the problem of Self-Healing. This paper reviews the data that Self-Healing uses as input and justifies its classification as Big Data. Big Data techniques applied to Mobile Networks are examined and some use cases along with their Big Data solutions are surveyed.**

***Index Terms*—Self-Organising Networks, Self-Healing, Big Data, Radio Resource Management, data mining**

I. Introduction

With each new generation of communication technologies, the control data and the network performance measurements in the Operation and Maintenance (O&M) subsystem grow both in volume and speed. A larger number of users and quantity of transferred data means a larger amount of measurements. In addition the higher demand for quality and larger bandwidths pushes for a faster rate of generation and consumption of these data. Downtimes due to problems that are not solved quickly or an incorrect optimization cause a high opportunity cost and increased overall O&M costs.

For these reasons, the NGMN Alliance [1] and the 3GPP [2] defined *Self-Organizing Networks (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. SON functions take as input the measurements generated by the mobile networks, and produce an output that depends on the purpose of the function.

SON functionalities can be classified in three large categories:

- Self-Configuration: functionalities that automate the planning and deployment of the network.
- Self-Optimization: functionalities that keep the configuration parameters always working in the optimal level to offer the best QoS.
- Self-Healing: functionalities that automate the solution of problems, reducing human intervention and minimizing downtime. Self-healing includes fault detection, diagnosis, compensation and recovery.

Self-Healing [3] aims to automate troubleshooting, which is one of the most important O&M tasks. The main objective of troubleshooting is finding and fixing malfunctions in the network. In the currently deployed LTE networks, this task is manually done by monitoring some variables that reflect the state of the network. Self-Healing functions include data processing algorithms that analyze O&M data in order to identify and fix problems. Some attempts at defining an implementation for Self-Healing functions have been done using Knowledge Based Systems (KBS), such as Bayesian Networks [3], Fuzzy Logic Controllers [4] or Case Based Reasoning [5] for automatic diagnosis. These systems have been proposed theoretically without acknowledging the problems inherent to large databases.

When data largely increase, traditional processing techniques have a very poor performance. The Big Data paradigm deals with this type of datasets by applying new techniques that exploit the latest hardware and software innovations. In mobile networks, performance measurement data have already passed that threshold, and therefore, it must be studied through the paradigm of Big Data. In [6], [7], a general vision of Big Data for SON functions was given, without going into details on each SON functionality. This paper goes one step further, focusing on the Self-Healing functions. In particular, in this paper, Self-Healing is reframed as a Big Data problem, and some specific requirements for the development of Big Data compliant Self-Healing functions are proposed. To further illustrate these principles, some use cases are shown, where modifications on previously existing algorithms that work with Big Data compliant inputs are proposed in order to use Big Data processing techniques.

In Section II, the data sources and their uses for troubleshooting in mobile networks are reviewed. Next, in Section III the Big Data concept is explained, and the characteristics of Self-Healing functionalities that are compliant with Big Data principles are described. In Section IV some use cases are exposed and Big Data compliant solutions proposed. Finally in Section V, the conclusions of the study are reviewed.

II. Troubleshooting Data in Cellular Networks

Each element in a cellular network, interacts with the rest of the network in order to establish communication, and each interaction produces several events (such as connection establishments, handovers, etc.). When a problem happens, the statistical data of these events can hold valuable information about the root cause. Therefore, in modern networks, all the elements save information about the events that they observe.

A. Data sources

The troubleshooting process uses data sources that indicate the state and behavior of the network. These data sources are usually recorded on-site in the Base Station (BS: the element that connects the network to the user terminals), which is connected to a data collection subsystem that regularly gathers all the information in a centralized database. The mainly used data sources are:

- Performance Management (PM) metrics: each BS 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). Other measurements are also taken and averaged during this time interval, such as the received power.
- Fault Management (FM) alarms: along with the counters, BSs 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 BS 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 took place. Each call trace contains registers (such as counters and alarms) and measurements related to a specific connection between a User Equipment (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 (timestamped 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.

PM, FM and CM are collected and stored in a raw format in the form of timestamped variables (time series) indexed by BS, but in order to make them more usable, they are usually transformed into more human-readable composite variables, the Performance Indicators (PIs). PIs measure a specific magnitude, such as the proportion of established or dropped connections over a time interval. PIs are aggregated over more varied time scales, such as hours or days. A reduced set of the PIs is representative of the general behavior of the element, and therefore are used to better understand it. These are the Key PIs (KPIs). On their turn, KPIs are aggregated over groups of elements, in order to represent the general behavior of the network or parts of it. Call traces are stored as files indexed by the session identifier, but they also have references to the BS and UE. Some PIs are calculated using the data of call traces. For this, the data in call traces referring to a specific BS are aggregated in *synthetic counters* and then used to calculate the new PIs. These PIs are saved and used along with the PIs calculated from CM/PM/FM.

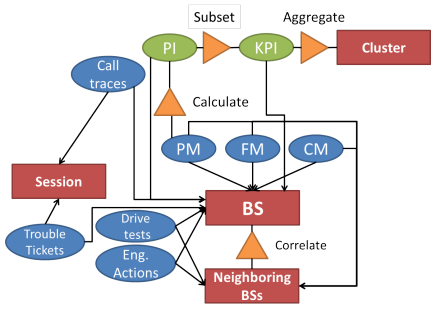

Fig. 1. Data types and the relationships established by the information each type contains.

It is important to note that, although the data sources are collected physically in individual network elements (usually the serving BSs), the information they contain may be relevant for other elements too, such as neighboring BSs. Figure 1 shows the complex relationships between the data types (marked in blue for the raw collected data and in green for the composite data) and the network elements of which they contain information (marked in red), as well as the processes that relate them (marked in orange).

B. Troubleshooting procedure

The manual troubleshooting workflow has four main subtasks:

- Detection: the process of determining that there is a problem in the network, and pinpoint the element or elements that are affected. To do this, troubleshooting engineers will usually monitor a very reduced set of KPIs. When one or more of these KPIs is degraded, a list of the Worst Offenders is extracted, that is, the elements (BSs, for instance) that are degrading mostly the KPI averages.
- Diagnosis: once the problematic elements have been determined, troubleshooting engineers must find out the root cause (i.e. why they are failing). The study of low level indicators, as well as logs or event records may help in the determination of the root cause. In some cases, active measurements are taken, such as drive tests.
- 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 give service to the users in the affected area. Since this temporary configuration is sub-optimal, the users may perceive a reduced QoS.
- Recovery: once the root cause is known, the required actions to fix it are taken. These actions range from simple resets or configuration changes that can be ordered remotely, to hardware fixes or replacements that need on-site reparations. 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.

C. Manual data processing

To better understand the meaning of the data, experts often use statistical techniques to simplify the representation of the collected variables. 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 discretized variables, heuristic rules of the *"if ... then ..."* form 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 coded as programs and 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 they are responsible for the observed behavior.

There is no normalized course of action for applying these processes. It is usually up to the judgment and experience of the troubleshooting expert which techniques to use and when based on their observations.

III. Big Data in Self-Healing

In the last years, Big Data solutions have been proposed for many applications. Generally, wherever a large user base is present, there are applications where large datasets are used and therefore, processes will have to be optimized for them. Mobile networks are a perfect example of such a scenario, and in this paper, the focus is set on their troubleshooting.

A. Introduction to 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 increasing exponentially. All these data contain information about a wide spectrum of aspects

that may be interesting for all types of businesses. As produced data grow, the information contained in them increases both in quantity and in accuracy, and so does the demand for extracting that information. The amount of available data is often too big and unstructured to be treated with traditional statistic methods to achieve the results and the 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 of Big Data [8]:

- Big Volume: the quantity of data that must be processed is large, either because there are many individual small information units (such as PM counters that are very simple and structured, but they are generated by a large number of BSs) or because each information unit is large (like call traces that contain variable fields, many measurements and the information is potentially referred to several different BSs). 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. In the case of mobile networks, each BS produces the CM, PM and FM data in each ROP. The number of individual variables is in the order of hundreds to thousands for each BS, and the number of BSs in a network ranges in the tens of thousands. Additionally, each of the millions of individual connections generates a call trace.
- 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. In a cellular network, the data is generated in every ROP, which usually is 15 minutes long. 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.
- 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 software drivers). Also, the data units may contain information about different entities that may or may not be needed for a specific application. As shown in Section II, in cellular networks, data formats are very varied. 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.

Big Data makes heavy use of distributed computing both for data processing (cloud computing) and storing (data warehousing). Some *de-facto* standards are the lambda

system architecture [9] for the high level design of Big Data systems, the MapReduce [10] programming architecture for low level algorithm design and NoSQL [11] databases for storage.

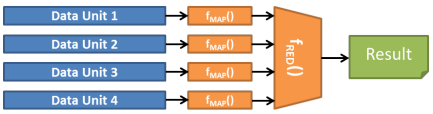

Fig. 2. MapReduce architecture

B. Big Data techniques

In mobile networks, we propose to use the lambda architecture, which is a common architecture for cloud computing. 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.

To reduce the processing time, parallelization is heavily used in Big Data. 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 2), 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.

Data Sources do not always provide structured data that can easily be stored on a table without loss of information. The NoSQL paradigm refers to database implementations that do not use the traditional table based architecture. In wireless networks, we propose to apply the *key-value* model, which is a common architecture, where each *record* (or *value*) is uniquely identified by a hashable variable (the *key*). On its turn, each record can contain a variable number of fields, which simplifies the storage of unstructured data and reduces the required storage space.

C. Applying Big Data techniques to Self-Healing

The main objective of Self-Healing is to automate the troubleshooting task described in Section II-B by using programs that replicate the manual processes described in Section II-C. Commercial requirements (the demand for QoS) create the need for a fast and reliable troubleshooting system that minimizes downtime. Therefore, automation in troubleshooting is required.

Self-healing algorithms are usually implemented us-

ing Knowledge Based Systems (KBS) that imitate the process of human experts in order to accomplish a task. KBS are algorithms composed of two parts:

- Knowledge Base (KB): a codified representation of the field knowledge, that is, the knowledge that the experts need in order to complete the task. To generate and improve the KB, a continuous Data Mining (DM) process is run in the Batch Layer.
- Inference Engine (IE): the procedures that use the KB in order to complete the task. The IE conforms the Speed Layer.

Some KBS that have been previously used in troubleshooting are Fuzzy Logic [4] or Bayesian Networks [12]. It is important to follow the guidelines of Big Data when designing these implementations, that is, creating parallelizable algorithms. 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.

IV. Use cases

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

A. Data reduction

Big datasets are often collected to be inspected by human operators or for unspecified future uses. Therefore, it is a common problem that the relevant information for a specific application is hidden among redundant or even unrelated information. Therefore, for many SON algorithms, a necessary prior step is extracting the subset of data where the information is contained. In troubleshooting, some algorithms need to extract information from degraded indicators. These values can be extracted from historical O&M databases that store the timestamped indicator values (all the data types described in Section II), but in such databases, both normal and degraded values are indistinctly saved, with no indication of the status (i.e. normal / degraded). Therefore it is important to devise methods to isolate the time intervals where the behavior of each element is degraded.

Since the O&M databases are very large, it is important to perform this process with a parallelizable algorithm. Each parallel process must be independent from the result of other processes. An easy way to divide the work in this case is to analyze each BS individually. The algorithm described in [13] detects Degraded Intervals (DIs, time intervals where the behavior of a BS is degraded) by analyzing KPIs. For each BS, the time series of one KPI is sequentially analyzed and compared against a *good* threshold and a *bad* threshold. A DI starts when the value of the KPI falls below (or rises above) the *bad* value, and ends when it crosses again the *bad* threshold and then rises above (or falls below) the *good* threshold. This algorithm can easily be parallelized since each BS can be processed by a separate instance (by implementing the detection algorithm as a *map* function). The input database is filtered by BS, and each instance of the *map* function reads the KPI time series of the filtered table. On the output, the result is aggregated, adding a field identifying the BS and by finally applying a *reduce* function that saves all the detected DIs in the same table. An example of a result of the algorithm is shown in Figure 3, where the *Retainability* KPI is used to determine the DIs. This specific use case comes from a dataset containing 47 samples, each with two weeks of hourly data; making a grand total of 15792 entries. The total number of detected DIs is 359, and since the KPIs are aggregated over the duration of the DI, the reduced database contains 359 entries, representing a reduction of 97.73% of data volume.

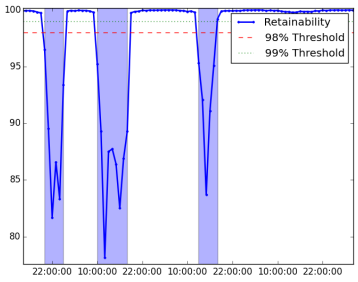

Fig. 3. Detected Degraded Intervals

B. 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 BS 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 BSs measurements to detect the outage by calculating its impact. Since in a network the number of BSs 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 bad user experience.

In [14], an algorithm for detecting sleeping cells based on the decrease of handovers with neighboring cells is described. To find sleeping cells, for each BS, the number of incoming handovers for the current and previous ROP is aggregated from the neighbor BSs 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 (a 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. Each instance of the algorithm tests sequentially each of the BSs contained in one partition by looking into the data of its neigbors, 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.

Figure 5 shows the results obtained in [14] 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 (i.e. the percentage of wrong detected cases among the total outage cases) 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. Thes results show that the increase in the volume of processed data improves the detection capacity.

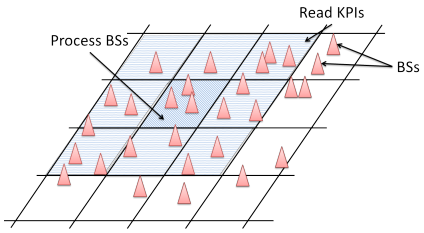

Fig. 4. Terrain division for the detection of sleeping cells

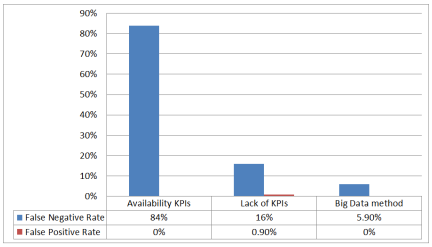

Fig. 5. Comparison between the proposed Big Data method and other common techniques

C. Diagnosis based on KPI correlation

In troubleshooting, once a problem is detected in a BS, the diagnosis task is launched. Normally, detection is done by observing very high level KPIs, whereas diagnosis takes into account all the available information to find the root cause. 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 behavior of the BS, a list of the most correlated indicators will give an important clue to the root cause analysis. In [15], a method that performs this analysis is described. This algorithm takes into account the PIs of the affected BS 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

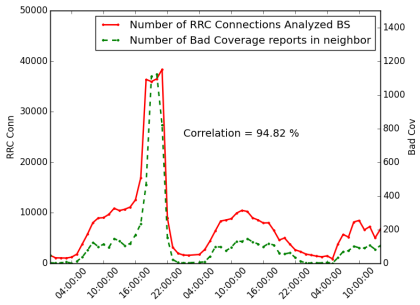

Fig. 6. Correlation between the KPI and a PI of a neighboring sector

each PI can be processed independently, the algorithm is easily parallelizable. Again, the correlation process is implemented as a *map* function, and a *reduce* function creates a list of PIs ordered by correlation. In Figure 6, a time series of a KPI in the diagnosed BS (the *Number of Radio Resource Control Connections*) is shown, along with a highly correlated PI of a neighboring BS (a counter of *Bad Coverage* reports) and their correlation.

V. Conclusions

Automation of the O&M of the mobile networks is a need for mobile operators due to several forces driving the costs up: the increase in traffic volume, variety of services, number of subscribers, demand for QoS and competitiveness among operators. Modern networks (especially LTE and the future 5G networks) produce large volumes of O&M data, that contains the information needed for the Self-Healing functions that operators need. In fact, the volume of this data, along with the variety and time restrictions, calls for the use of the Big Data paradigm. Big Data is a set of new techniques that exploit the ever-increasing processing power of computer networks. Specifically, in this paper, the main focus is parallelization. Several use cases have been displayed as examples where data processing algorithms can extract information contained in Big Data compliant data

sources. In order for these algorithms to work properly, small modifications have been proposed in order to run the algorithm in parallel processes that do not interfere with each other. Parallelization reduces the processing time (or increases the processing capability) of these algorithms, so SON functions can be applied in a manner that is transparent for the users of the network. The proposals of this paper can be extended to any SON algorithm that can be extended in a way that guarantees that the results of the parallelized version are the same as the original. In order to further test the proposed solutions of the use cases, they can be implemented using the commercially available Big Data processing solutions.

ACRONYMS

BS	Base Station
CM	Configuration Management
DM	Data Mining
FM	Fault Management
IE	Inference Engine
KB	Knowledge Base
KBS	Knowledge Based Systems
KPI	Key Performance Indicator
O&M	Operation and Maintenance
PI	Performance Indicator
PM	Performance Management
QoS	Quality of Service
ROP	Report Output Period
SON	Self-Organizing Networks
UE	User Equipment

ACKNOWLEDGMENT

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.

REFERENCES

[1] NGMN, *Use Cases Related to Self-Organising Network, Overall Description*, Next Generation Mobile Networks (NGMN) Alliance, April 2007.

[2] 3GPP, *Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements (Rel 11), TS 32.541*.

[3] R. Barco, P. Lázaro, and P. Muñoz., "A unified framework for self-healing in wireless networks," *IEEE Communications Magazine*, vol. 50, pp. 134–142, December 2012.

[4] E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, and I. Serrano, "Data mining for fuzzy diagnosis systems in LTE networks," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7549 – 7559, 2015.

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

[6] N. Baldo, L. Giupponi, and J. Mangues-Bafalluy, "Big Data Empowered Self Organized Networks," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of.* VDE, 2014, pp. 1–8.

[7] A. Imran and A. Zoha, "Challenges in 5G: how to empower SON with big data for enabling 5G," *Network, IEEE*, vol. 28, no. 6, pp. 27–33, 2014.

[8] P. Russom, "Big data analytics," *TDWI Best Practices Report, Fourth Quarter*, 2011.

[9] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, 1st ed. Manning Publications Co., 2015.

[10] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[11] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12–27, 2011.

[12] R. Barco, P. Lázaro, L. Díez, and V. Wille., "Continuous versus discrete model in autodiagnosis systems for wireless networks," *IEEE Transactions on Mobile Computing*, vol. Vol.7 (6), pp. 673–681, June 2008.

[13] E. J. Khatib, R. Barco, I. Serrano, and P. Muñoz, "LTE performance data reduction for knowledge acquisition," in *Globecom Workshops (GC Wkshps), 2014*, Dec 2014, pp. 270–274.

[14] I. de la Bandera, R. Barco, P. Muñoz, and I. Serrano, "Cell outage detection based on handover statistics," *Communications Letters, IEEE*, vol. 19, no. 7, pp. 1189–1192, 2015.

[15] P. Muñoz, R. Barco, I. Serrano, I. de la Bandera, and E. J. Khatib, "Fault diagnosis in networks," Patent, 4 24, 2015.