Noname manuscript No.
(will be inserted by the editor)

Knowledge Acquisition for Fault Management in LTE Networks

Emil J. Khatib · Raquel Barco · Pablo Muñoz · Inmaculada Serrano

Received: date / Accepted: date

Abstract Self-healing is one of the main functionalities of Self-Organizing-Networks (SON). Among self-healing functions, *diagnosis* or *root cause analysis*, consisting of identifying the fault cause in problematic cells, is one of the most complex tasks. Expert systems, such as Fuzzy Logic Controllers or Bayesian Networks, have been previously proposed to implement automatic diagnosis systems in the radio access segment of mobile communication networks. In order to achieve accurate results, these diagnosis systems should contain the knowledge of experienced LTE troubleshooting experts. However, these experts normally have neither the time nor the expertise in artificial intelligence to define the expert system. In this work, we propose a novel knowledge acquisition system that obtains this knowledge in the least possible intrusive way. The proposed method collects the Performance Indicators data from the relevant time intervals together with the expert's diagnosis and uses them as inputs for a Data Mining algorithm to extract diagnosis rules.

Keywords knowledge acquisition · LTE · fault management · troubleshooting

Emil J. Khatib
Universidad de Málaga
Tel.: +34 952134164
E-mail: emil@uma.es

Raquel Barco
Universidad de Málaga
E-mail: rbarco@uma.es

Pablo Muñoz
Universidad de Málaga
E-mail: pabloml@ic.uma.es

Inmaculada Serrano
Ericsson, PBO RA Continuous Analysis
E-mail: inmaculada.serrano@ericsson.com

1 Introduction

The growth in complexity and the number of new services offered by mobile networks in the last decade have lead to the need of extra investment in human resources for network maintenance, consequently, raising Operational Expenditure (OPEX). To relieve this increase, automation of the network functions is required. For this reason, the Next Generation Mobile Networks (NGMN) Alliance and the 3rd Generation Partnership Project (3GPP) defined Self Organizing Networks (SON)[13][1] as the theoretical groundwork for automating operation and maintenance of complex mobile networks. The SON functionalities in the Long Term Evolution (LTE) and 5G networks are classified into three big groups: *Self-configuration*, *Self-optimization* and *Self-healing*. This work is centered in Self-healing, the ability to automatically recover from failures.

The self-healing [3] procedure consists of automatic troubleshooting (detecting, diagnosing and recovering from faults in the radio access network) and, in the meantime, compensating the affected users with the service of alternative sectors. This paper is focused on improving the diagnosis phase, also called root cause analysis. Based on the authors experience in close relation with industry experts, in commercial networks, this process is largely manually done, with the implied costs of time and prolonged downtimes. These costs are partially reduced with the use of specialized tools that assist experts in their troubleshooting chores [12], but the manual work is still required. Therefore, there is an urgent need of automating the troubleshooting process.

A significant research effort has been invested in this field. The UniverSelf project [19] proposes a diagnosis system based on a combination of Bayesian Networks (BNs) and case based reasoning [6]. The COMMUNE [8] project uses a system based on comparing current cases with stored cases labeled with their root causes [17]. In [2] a troubleshooting system based on BNs is described, and the process for acquiring the required knowledge is described in [4]. Self-healing research is ongoing for future 5G networks with the SELFNET project [16], although development is still in its early design stages [15].

All these studies use Knowledge Based Systems (KBS) [12] for their implementation. KBS include a Knowledge Base (KB) that holds the domain knowledge of the experts and varies according to the problem, and an Inference Engine (IE), that applies the knowledge on the specific problem. The performance of KBS for diagnosis is strongly dependent on the quality of its KB. Therefore, the KB must contain knowledge that is somehow extracted from field experts. The process of filling the KB with expert knowledge is known as *Knowledge Acquisition* (KA). Traditionally, there are two main ways to perform KA for the previous systems: by interviewing the experts (involving a *knowledge engineer*) or by asking the experts to create the KB. The problem with these methods is that they are both too intrusive in the expert's daily routine, and therefore cause that they do not collaborate in the process of development, ultimately leading to deficient KBS that perform poorly and are not commercially attractive [12]. This is the reason why self-healing systems

in mobile networks have not yet seen a practical large scale deployment. In order to overcome those limitations, in [4] the interview method is improved by substituting the knowledge engineer with a Knowledge Acquisition Tool. However, the time required from the troubleshooting expert is still too demanding and out of the scope of his daily work.

In this paper, a third approach to build the KB is proposed, which is based on applying Data Mining (DM) on a pool of historical data. In this case, KA is approached as a *knowledge modeling* process [21], enabling the use of a wide range of techniques of Knowledge Discovery and Data Mining (KDD)[10]. A challenge of this method is that, although the intervention of the experts is no longer required in the construction of the KB, someone must provide the KA system with the appropriate training data (the *training* set). DM has been previously proposed as a solution for KA [17], but there are no studies that approach the challenges that data from real networks poses on the process.

In the context of the Radio Access Network (RAN) of cellular networks, the data pool that contains information about the troubleshooting processes applied by the experts is the status information (measured by counters, Performance Indicators, etc.) of the sectors in the network during the occurrence of the problems, including the fault cause. This concept of fault databases (databases containing a record of past faults) has been explored in other fields [18][7], and it has been previously used for learning the diagnosis model in mobile communication networks [17][5]. However, those studies were based on two unrealistic assumptions: (a) a fault database with a large volume of fault cases is available and (b) the Performance Indicators (PIs) are ready to be used by the automatic diagnosis algorithms. Although the existence of a fault database has many benefits apart from the possibility of training KBSs (e.g. traceability of the troubleshooting actions, generation of reference cases for training future experts, etc), in practice there are currently no such databases related to mobile networks. Although the values of the main PIs are saved in most network management systems, the fault causes in problematic cells are not normally recorded. This is due to the fact that registration of solved troubleshooting cases is not part of the normal workflow of troubleshooting experts. The PIs collected by the network management systems very often cannot be used directly DM because of multiple reasons: missing values, redundant information, lack of definition in the time when the fault occurred, etc.

In order to overcome both impairments, which limit the development of self-healing systems in commercial networks, this paper analyzes the requirements of KA in mobile network troubleshooting and proposes an integrated method that provides mechanisms to:

- Collect a fault database with minimal human intervention, in an unobtrusive way so the collaboration of experts is optimized in the process of KA.
- Process the collected fault data preparing it for DM.

– Perform DM on the collected fault data to extract a KB for a diagnosis system.

The proposed mechanisms are then implemented and tested on fault data extracted from a real network.

The remainder of this article is organized as follows. Section 2 poses the requirements of an implementation of a KA system. In Section 3 the proposed architecture is described in detail. Section 4 presents the application of the method on a real network. Finally, the conclusions are reviewed in Section 5.

2 Requirements of the KA System

2.1 User Interface requirements

The most important aspects that must be addressed in the design of the UI are the needs and/or limitations of the users (in our case, the troubleshooting experts).

– 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 case is not *intrusive*.
– 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.

2.2 Knowledge Base requirements

It is important that the collected data is as *clean* (i.e. no missing or corrupted information) and *correct* (i.e. no wrong or conflicting diagnoses) as possible. Otherwise, the resulting KBS will produce poor results. 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. At the same time, it is important that there is a comparable number of instances for each problem; that is, that the database is *balanced*, so that a small number of cases of one dataset cannot be confused with outliers.

The output produced by the KA system is a KB that will be used in a KBS to perform diagnosis. The validity of this KBS (and therefore the KB) depends on the proportion of troubleshooting cases it is able to diagnose correctly. There are several important metrics that assess the performance of the KBS:

– Diagnosis Error Rate (E_d): proportion of cases with a certain root cause that the KBS diagnoses with a different root cause.

$$E_d = \frac{N_{de}}{N_p} \tag{1}$$

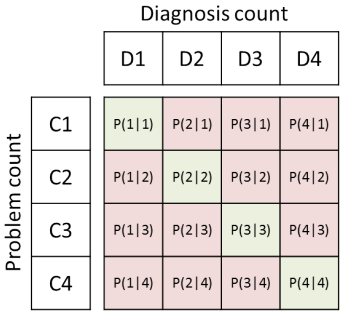


Fig. 1 Example of a confusion matrix

– Undiagnosed Error Rate (E_u): proportion of cases with a problem that cannot be diagnosed, that is, cases that produce no output (or unknown cause) in the KBS.

$$E_u = \frac{N_{un}}{N_p} \tag{2}$$

– False Positive Error Rate (E_{fp}): proportion of normal cases (those with no problems) that are diagnosed as problematic. Only cases that are false positives in the detection phase may reach the diagnosis stage and therefore produce a false positive.

$$E_{fp} = \frac{N_{fp}}{N_n} \tag{3}$$

The error rate requirements may vary according to the specifications. Another important measurement is the *confusion matrix*, which shows the proportion of cases of one specific problem (rows) that are diagnosed as each of the possible problems (columns). The diagonal of this matrix is the success rate of the diagnosis system for each problem. Fig. 1. Each cell in this matrix represents the probability $P(i|j)$ of cases labeled with Cause j (Cj) that are diagnosed (Di) by the diagnosis system as having Cause i.

3 Method for Knowledge Acquisition for Wireless Networks

In this section a KA method is proposed. Fig. 2 shows a diagram of the individual processes applied over the data in the method.

There are two main stages in KA:

– Knowledge Capture: collection of the information about the case.
– Modeling: processing of the information where the obtained data is prepared (cleaned, reduced and balanced) and then used for data mining.

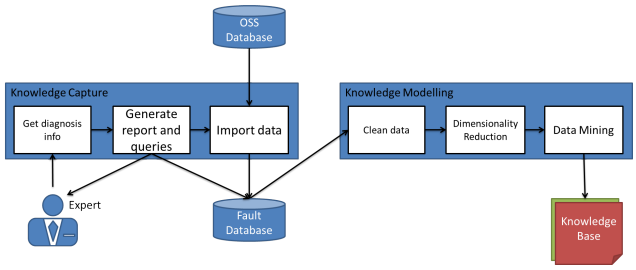


Fig. 2 Flowchart of the KA system

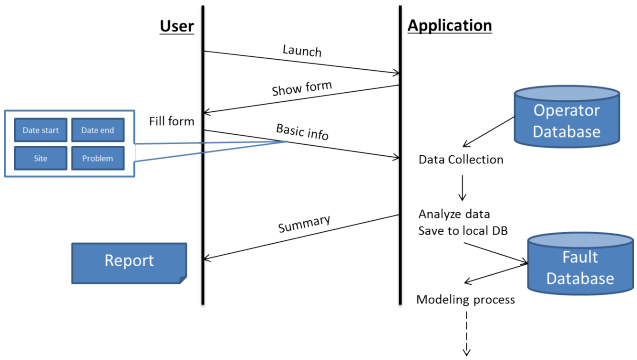


Fig. 3 Interaction between the UI and the user.

3.1 Knowledge Capture

In the Knowledge Capture stage, the troubleshooting expert is asked to provide some information about the problem, mainly, the date of the beginning and end of the problem, the sector where the problem occurred and the root cause. Fig. 3 shows the process of interaction with the expert. If the proposed method is embedded into the visualization software that experts use to diagnose networks, the date of the problem and sector information can be extracted from the context (i.e. the data visualized by the expert at a given moment). In that case, experts only need to provide a diagnosis, therefore greatly reducing the required effort for adding cases to the fault database.

With this information, the system can automatically obtain a full data set representing the status of the sector during the occurrence of the problem. This data set may contain several data types:

- Counters: variables that count occurrences of specific events in a sector, such as *dropped calls* or *failed handovers*, etc.
- Performance Indicators (PIs): composite variables computed from the values of individual counters, such as *dropped call rate* or *handover success rate*.

- Configuration parameters: variables or constants that determine the behavior of a sector. Wrong values or changes in parameters may cause problems that result in abnormal behaviors of counters and PIs.
- Call traces: activity registers of individual calls.
- Consistency checks: some automated checks are done on the values of counters, PIs and configuration parameters to detect specific problems or situations such as cell identifier conflicts.
- Alarms: indicators or flags that report a specific problematic situation.

3.2 Data Cleaning

Once all the information has been retrieved, the Modelling stage starts. Sometimes in the monitoring subsystem of mobile networks, some values of counters may fail to register due to problems such as bad network connectivity, site outages or even because the monitoring system is not programmed to log those counters. The fact that there are missing data may itself be an important information, but it may also complicate the data mining process.

In this work, single missing values will be filled using the interpolation between the previous and next hour and several consecutive missing values will be filled using the following formula:

$$x_m = \frac{\bar{x_{48}}}{\bar{x}} \bar{x_H}(h) \tag{4}$$

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

3.3 Data Reduction

The next step is transforming the data to an appropriate format for the DM stage. At this point, the information available about the sector suffering a problem is a huge set of time-dependent variables saved in a database. To produce valid results in the context of data mining, it is vital to perform a reduction in the dimensionality and volume of the information; so that the format of the data set matches that required by the DM stage.

The result of the data reduction stage is a vector $\bar{x}$ that describes the behavior of relevant variables as briefly as possible. This vector must at least contain all the features that an expert would observe to diagnose the problem. The vector $\bar{x}$ is joined with a label L_p that identifies the root cause in a tuple $(\bar{x}, L_p)$. This tuple will then be used, along with tuples representing other problems, to feed the data mining stage.

The implemented method explores a KPI time series to delimit the degradation periods, mirroring the manual process of troubleshooting. Two thresholds, corresponding to the *"high"* (h) and *"low"* (l) boundaries of fuzzy sets

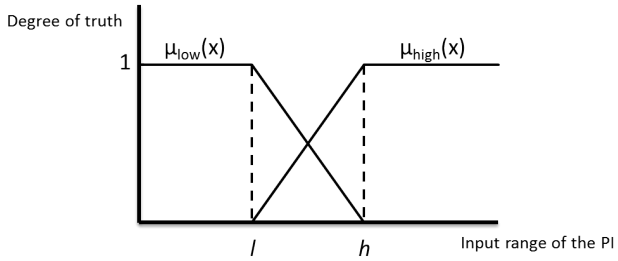


Fig. 4 Continuous PI with fuzzy sets defined on its domain

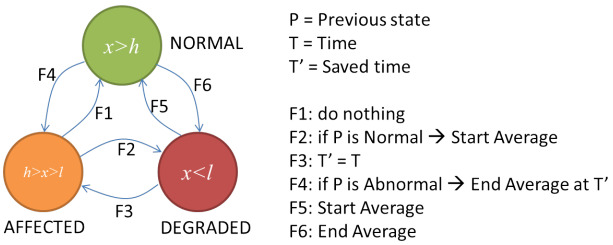


Fig. 5 Data reduction algorithm

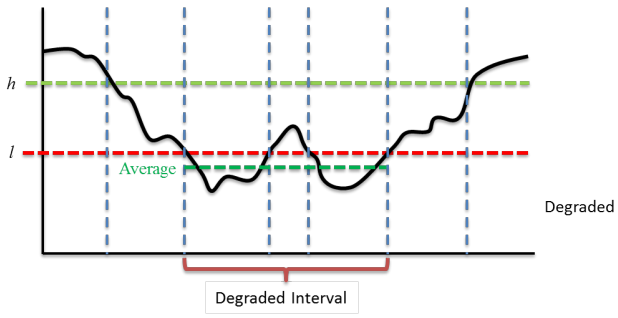


Fig. 6 Example of the data reduction algorithm over a single PI

defined over the domain of the KPI (Fig. 4) are used for this purpose. These boundaries are usually predefined either by the experts or even by the network provider.

A KPI that normally takes a *high* value is considered degraded when it crosses the l boundary. Every time that the value crosses above the l boundary, the time of the occurrence is saved. When the value crosses above the h boundary, the last saved time is considered as the end of the degradation period. Each degradation period obtained in this way is considered as a separate occurrence of the problem. Once the degradation intervals are detected, the averages of all the other PIs are calculated for each interval (aggregation), and a vector with their values is generated. This algorithm is depicted as a state machine in Fig. 5 and an example of its operation is shown in Fig. 6.

3.4 Data Balancing

Once data is reduced, a database containing labeled samples of data is obtained. Each label defines a *class* (in this case, a class represents a specific problem). The distribution of samples among these classes is highly dependent on a large number of factors, such as experts knowledge (not all experts are specialized on the diagnosis of the same problems), the probability of occurrence of a problem, the configuration of the network, etc. This is very often a problem in Data Mining, since classes with a low representation tend to have a lower impact in the final result. For instance, a value that would be considered as an outlier in a class that has many instances, may have a higher count that all the instances of a class with a low representation. This would mean that the final result would either take into account the outliers of the larger class or qualify the smaller class as an outlier.

Two similar solutions to this problem are either dropping randomly the samples of larger classes or randomly creating copies of the samples of smaller classes. Nevertheless, by creating copies of the smaller datasets, there are possibilities of amplifying the effects of outliers if the number of original instances is very low. Therefore, in this study, all the classes are reduced in size to that of the smallest one.

3.5 Data Mining

Once the data is clean, reduced and balanced, a DM algorithm can be applied to extract (or *learn*) patterns. The output of the DM is a *model* of the knowledge contained in the used data set. In the case of troubleshooting this is the knowledge used by experts when diagnosing the problem. The specific DM algorithm to be used varies according to the format of the output, that is, the format of the KB used in the target KBS (e.g. in the case of a Fuzzy Logic Controller (FLC), the algorithm should be targeted at obtaining fuzzy rules and fuzzy sets, whereas if the KBS is a BN, it should calculate statistical relations among variables and conditioned probabilities).

Although the method proposed in this paper is valid for any KBS, as an example to illustrate the DM stage an FLC will be used. In FLCs, the KB is divided in two parts: a *Data Base* (DB), that contains the models of the input and output variables, and the *Rule Base* (RB), that expresses the relations among the variables using *"if .. then ..."* rules, similar to those that the experts use to manually diagnose the problems. In the implementation of the DM done in this study, there will be separate processes for learning the DB and RB:

1. *Data Base Learning*: The Data Base (DB) in an FLC contains the definitions of the fuzzy sets defined over the variables. In Fuzzy Logic, membership functions indicate the degree in which a value belongs in a fuzzy set. There are several standard membership functions (e.g. trapezoidal, singleton) characterized by a set of parameters that allow to adapt such functions to the problem. The process of learning the DB is obtaining these

parameters from the data. For LTE troubleshooting, trapezoidal membership functions are an appropriate model for the fuzzy sets of the variables that will be used as inputs to the FLC. Fig. 4 shows a typical continuous PI with the *"high"* and *"low"* membership functions defined on its domain. DB learning consists of the obtention of parameters l and h. In this work we will use the Enthropy Minimization Discretization (EMD) technique described in [9] and improved in [14], which is a supervised algorithm that finds the best threshold that divides the domain of the PI based on the entropy. This algorithm is used first to find only one best division, and then again over the resulting subsets in order to find a secondary threshold. The best result among these two executions is taken as the secondary threshold. For the output variable, that is, the diagnosis, the learning process involves the creation of a list of the possible root causes (those present in the training set). When diagnosis is performed, each cause will receive a score representing its confidence.

2. *Rule Base Learning*: The Rule Base (RB) in an FLC contains the definitions of the fuzzy rules. While the DB represents the behavior of each variable independently, the RB describes the relations among variables. Fuzzy rules take the form of *"if ... then ..."* propositions. For the data mining to take place, an appropriate representation for the rules must be created, so that the learning algorithm can easily work with them. Each rule is represented with a $(\bar{f}, L_P)$ tuple containing a vector $\bar{f}$ of fixed length and a class label L_P representing the root cause identified in the consequent. Each element of vector $\bar{f}$ represents the assertion about a PI x that has an assigned fuzzy label y (the individual *"PI x is y"* expressions). Some rules may only use a subset of the input variables, so their $\bar{f}$ vector only contains fuzzy values in the elements corresponding to that subset; the other elements containing an empty value. In this work, it is considered that the individual assertions are all joined by *"AND"* operators. Using the $(\bar{f}, L_P)$ representation of the rules and the $(\bar{x}, L_P)$ tuples of the training set, the DM is implemented as a data driven learning algorithm [11] based on the WM algorithm [20]. This algorithm translates directly the $(\bar{x}, L_P)$ training tuples into $(\bar{f}, L_P)$ representations of rules. The $\bar{f}$ vector is the antecedent of the rule (the *"If ... and ..."* part), and L_P is the consequent (the *"then ..."* part). In the case of diagnosis, the consequent is the diagnostic. Each created rule is given a score based on the evaluation of its success rate over the training set. A set of rules is obtained from this process. In this set, there may be rules that contradict each other (i.e. they have the same antecedent $\bar{f}$, but a different consequent L_P). The score will decide which rule is the chosen. Also, rules with the same consequent where the difference in the antecedent is only found in one element are fused into one rule that has no requirement over that element, since the rule must be activated regardless of the value of that PI. The resulting rule base is a reduced representation of the training set that can be used in an FLC to diagnose new cases.

4 Evaluation

In this Section, the KA process described in this work is applied to a real LTE Network.

4.1 Experimental setup

The full process is applied over a set of 47 problematic cases (distributed as shown in Figure 7) and 100 normal cases. These cases have been collected from a real urban area network. When experts report a problem and the diagnosis, the data is collected for a prior period of two weeks for each problem, for 330 PIs. Nevertheless, only a reduced set of PIs is selected for this study:

- *Average CQI*: Average of the CQI (Channel Quality Indicator) reported by the user terminals during a time interval.
- *Average number of active terminals*: Number of active users in the Downlink.
- *Average UL RSSI*: Average RSSI (Received Signal Strength Indicator) in the Uplink.
- *Handover Success Rate (HOSR)*: Proportion of handovers that are correctly finalized.
- *Inter-Frequency Handover Preparation Rate*: Proportion of handovers that are done between cells in different carriers.
- *Intra-Frequency Handover Preparation Rate*: Proportion of handovers that are done between cells in the same carrier.
- *IRAT Handovers*: Proportion of handovers that are done between cells of different technologies.
- *Number of Bad Coverage Reports*: Number of Bad Coverage reports collected from the users of the cell.
- *Number of calls transferred to 2G/3G (CS fallback)*: Some calls are transferred to 2G or 3G for offloading the LTE network or for special needs of the call.
- *Number of CPU Overload Alarms*: Number of alarms that indicate a high activity in the CPU.
- *Number of ERAB Attempts*: Number of E-UTRAN Radio Access Bearer Connection Requests. It reflects the offered traffic.
- *Retainability*: Proportion (percentage) of connections that end normally. Retainability is the inverse of the *Dropped Call Rate*. In this study, it is used to detect the degradations in the data reduction stage (Section 3.3).
- *DL traffic volume*: Quantity (GB) of transferred data in the Downlink.
- *UL traffic volume*: Quantity (GB) of transferred data in the Uplink.

The collected cases belong to 4 types of problems:

- High Traffic (14 cases): The cell may be having problems due to an abnormally high number of users.

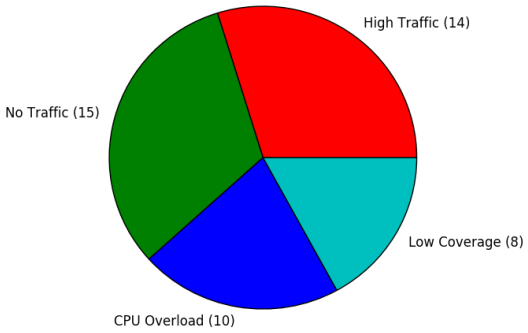


Fig. 7 Distribution of the collected faults among the problem classes

- No Traffic (15 cases): The cell may be having configuration problems that are causing an abnormally low number of users.
- High CPU utilization (10 cases): The eNodeB's CPU is heavily used and it is not due to a high number of users. Normally this is due to a SW problem.
- Low Coverage (8 cases): The radio signal of the network is not powerful enough in a specific location.

The first step will be to clean (Section 3.2) and reduce (Section 3.3) the data. To better understand this process, a single case will be shown in full detail. After these processes, data balancing (Section 3.4) is applied in order to finish the data preparation stage. For the evaluation of the DM stage, the learning algorithms described in Section 3.5 are applied over the reduced dataset and the measurements of Section 2.2 are taken over the result. To test the impact of the cleaning and reducing stages on DM, two additional tests are carried out. In the first test, the effect of data balancing is evaluated. For that, the DM algorithm is tested with a data set containing the whole data collected for each problematic case. Afterwards, a second test is performed that removes the effects of both data reduction and balancing, removing both processes from the training set.

4.2 Data preparation

The first step after data collection is data cleaning. To test the accuracy of the proposed method, it has been applied over 100 different time series of the same KPI (*Number of ERAB 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 (two

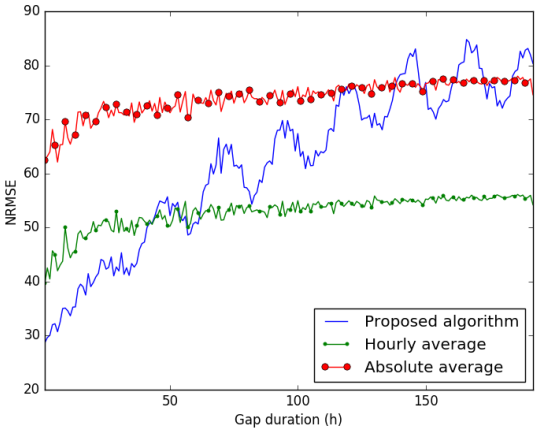


Fig. 8 NRMSE of the data filling method compared with filling with a constant (the absolute average) and the hourly average.

Table 1 Data Reduction Results

	Before	After	Variation
Number of samples	47 cases	359 degradations	↑ 663.83%
Data volume	5211360	118470	↓ 97.73%

commonly used techniques for missing data). The results are shown in Fig. **??**. 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.

Once data is clean, the data reduction stage is executed over the data set. The results of data reduction are summarized in Table 1. As stated in Section 4.1, the collected data consists of 47 problematic cases. Each has information on 330 PIs on an hourly level for 2 weeks (that is, 336 hours). That sums up a total of 5211360 elements. When the data reduction algorithm is applied over these cases, it detects 359 degradations. Since each degration is a vector of equal size to each hourly vector of the original database, the new total number of elements is 118470. This represents a volume reduction of 97.73%. The distribution of the resulting degradations among the four classes is shown in Fig. 9. An increase in the imbalance among the cases is observed, most notably a disproportionate multiplication in *No traffic* degradations. The number of detected degradations depends highly on the nature of the root cause and the specific case. Since the class with the lowest number of samples is *CPU Overload* (19 samples), random samples of 19 entries of the other classes are taken and used for training the DM algorithm (data balancing).

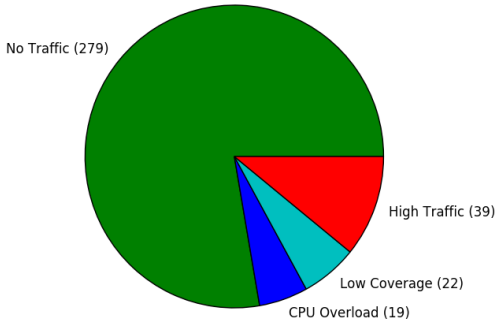


Fig. 9 Distribution of the detected degradations among the problem classes

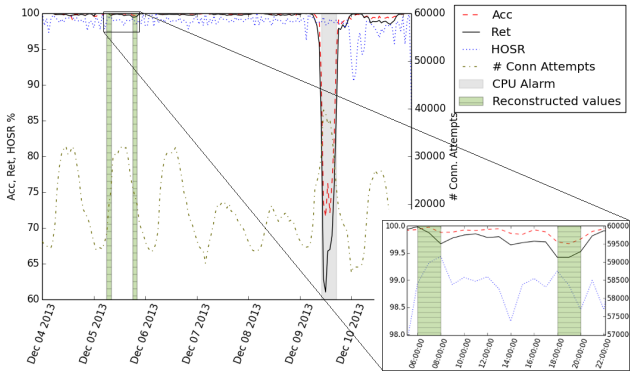


Fig. 10 Traffic overload example

To better understand the behavior of the DI detection algorithm, a specific case of the database is shown in detail. In the studied case, the site has a surge of traffic that rapidly causes an increase in the CPU load, raising an alarm (the *CPU load alarm* PI). This causes rejection in connection attempts (observed in the reduced *Accessibility*) and drops in established connections (reduced *Retainability*). The limiting factor in this case is not the radio capacity, but the *software capacity* of the site. Once the traffic is restored to a normal value, the behavior of the site returns to normal. The *CPU load alarm* may indicate several different root causes (such as software problems, traffic overload in current or cosite sectors, UE problems, etc). In this case, since the traffic rises at the same time as the alarm occurs, the case is diagnosed as having a traffic overload. Fig. 10 shows the time series of PIs selected for illustrating this problem. The problem occurs in Dec 9.

Table 2 Data Preparation Result

	Acc.	Ret.	CPU alarm	HOSR	Num ERAB Att.
Average in degraded period	79.69	71.557	On	98.804	33783.625
Average/standard deviation in normal period	99.804 / 0.215	99.737 / 0.318	Off	99.232 / 1.465	18127.306 / 6071.044

Table 3 l and h thresholds extracted from the data

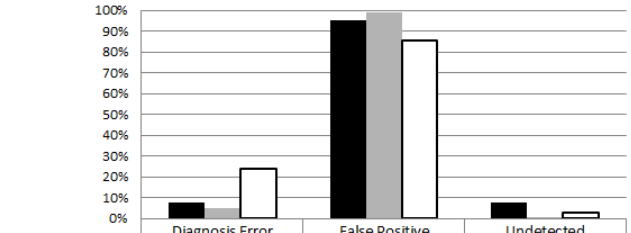
Average CQI	5.52	10.57
Average RSSI	-119.11	-109.14
CS Fallback Rate	0.0	4.03
Average Number of Active UEs	0.0	0.81
DL Traffic Volume	0.05	1.17
Handover Success Rate	20.61	99.72
Interfreq HO Preaparation Rate	0.0	99.97
Intrafreq HO Preaparation Rate	39.07	99.32
iRAT rate	0.12	58.49
Number of ERAB Attempts	14.0	7180.33
Number of CPU Overload Alarms	0.21	59.1
UL Traffic	0.0	0.16
Number of Bad Coverage Reports	1.5	384.89

The first step of the algorithm once the data has been collected is data cleaning. In Figure 4, there are two small intervals where data are missing and must therefore be reconstructed. Since the gaps are both longer than one hour (Dec 5, between 6:00 and 8:00 and later that day between 18:00 and 20:00), they are filled using the daily averages at the hour of the missing data for each PI.

The next step is data reduction. First, the degradation is detected. The result of the detection matches the time interval where the CPU overload alarm is active. In this case only one isolated period is detected. Next, the averages of the PIs in the degraded interval are calculated and a summary vector is generated. Table 2 shows the PI averages during the occurrence of the problem, along with the averages out of the degraded intervals. It can be observed how the values of the PIs in the degraded intervals clearly differ from the normal periods. Nevertheless, this deviation is not observed for *HOSR*, since the *CPU overload* problem does not affect handovers.

4.3 Data Mining

After the data preparation stages, the Data Mining stage is tested. First, the Data Base is extracted from the data using the EMD algorithm. The resulting l and h thresholds are shown in Table 3.



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
Proposed System	7.66%	95.00%	7.52%
No balancing	5.01%	99.00%	0.28%
No reduction	23.96%	85.50%	3.06%

Fig. 11 Results of the DM stage testing

With these thresholds and the dataset from data preparation, the Rule Base is extracted using the algorithm in [11]. Four rules are extracted (one per problem) and shown in Table 4.

Table 4 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	L	H						H	High Traffic
	L	L			L		L	L	L	L	L	L	No Traffic
			H	H		H	L		H		L		CPU Overload
				H		H	L	H	L		L		Low Coverage

These rules are applied over the cases in order to extract the error figures introduced in Section 2.2. The results are shown in Fig. 11, along with the error figures for the original dataset without data reduction (i.e., using each time stamped vector as one occurrence of the problem) and the reduced database without applying the balancing stage.

The results seem to contradict what would be expected (that the data preparation has improved the error figures). Nevertheless, a deeper analysis using the confusion matrices (Tables 5, 6 and 7) shows that the details are somehow more complex. The rule base extracted from the raw dataset without reduction misses two classes completely. As shown un Table 6, it can only diagnose *High Traffic* and *No Traffic* cases, but it will completely miss *CPU Overload* and *Low Coverage*. The result improves when applying data reduction (Table 6). In this case, both *High Traffic* and *No Traffic* cases are diagnosed with greater accuracy, whereas *CPU Overload* and *Low Coverage* cases are now successfully diagnosed 63.16% and 45.45% of times, respectively. This

Table 5 Confusion matrix for the ruleset extracted from the raw dataset

	Normal	High Traffic	No Traffic	CPU Overload	Low Coverage
Normal	0.14	0.62	0.23	0.0	0.0
High Traffic	0.03	0.97	0.0	0.0	0.0
No Traffic	0.0	0.2	0.8	0.0	0.0
CPU Overload	0.32	0.68	0.0	0.0	0.0
Low Coverage	0.18	0.59	0.23	0.0	0.0

Table 6 Confusion matrix for the ruleset extracted from the reduced and unbalanced dataset

	Normal	High Traffic	No Traffic	CPU Overload	Low Coverage
Normal	0.01	0.34	0.28	0.0	0.38
High Traffic	0.0	1.0	0.0	0.0	0.0
No Traffic	0.0	0.0	1.0	0.0	0.0
CPU Overload	0.05	0.32	0.0	0.63	0.0
Low Coverage	0.0	0.55	0.0	0.0	0.45

Table 7 Confusion matrix extracted for the reduced and balanced dataset

	Normal	High Traffic	No Traffic	CPU Overload	Low Coverage
Normal	0.05	0.28	0.32	0.01	0.34
High Traffic	0.0	0.92	0.0	0.0	0.08
No Traffic	0.1	0.0	0.84	0.0	0.06
CPU Overload	0.0	0.16	0.11	0.68	0.05
Low Coverage	0.0	0.0	0.07	0.0	0.93

results in lower Undetected and Diagnosis Error Rates. When data balancing is added to the process, a slight degradation in the accuracy for *High Traffic* and *No Traffic* with respect to the previous scenario occurs. In exchange, the *Low Coverage* cases are diagnosed correctly 93.18% of times, and a slight improvement also happens for *CPU Overload* cases. This means that the average success rate per problematic class increases from 77.15 % in the unbalanced dataset to 84.54 % for the balanced dataset. Nevertheless, the overall figures are worse for the balanced dataset because for the largest class (*No Traffic*) the error of the balanced dataset test is slightly higher. This means that, for this specific network, the balanced dataset will perform worse, although the error is more stable among classes. On the other hand, the non-balanced dataset ruleset will generally behave better in this specific scenario, although when a non-common problem (such as *Low Coverage* in this case) the diagnosis has a higher probability of being wrong.

5 Conclusions

In this work we have proposed a novel system for Knowledge Acquisition comprising the creation of a fault database and Data Mining. The interaction with the experts is therefore simplified by asking them only for examples of troubleshooting cases, instead of detailed descriptions of the process. This will reduce the time required to complete the training of the systems and the impact on the expert's workflow, therefore increasing the degree of collaboration. With the information obtained from the experts, the data of the sector is downloaded and prepared. Each occurrence of a problem is reduced to a representative vector, that is accumulated in a training set. A DM algorithm is then used on the training set to obtain a model of the troubleshooting process consisting of fuzzy rules. This model may finally be used in an FLC to diagnose problems in the live network.

The full KA process has been tested over a real fault database collected using the UI principles described in the study. The performance of the resulting FLC is measured. To show the benefits of the data preparation phases, the DM method is also tested with the same data without having used some of these processes. The results show that the performance of the FLC increases when using the full process, although there is a compromise when deciding whether to use data balancing, since it will reduce overfitting of the data on a cost of a lower accuracy for most common problems.

Competing Interests

The authors declare that they have no competing interests.

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. 3GPP: Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements. Next Generation Mobile Networks (NGMN) Alliance, ts 32.500 edn. (2012)
2. Barco, R., Díez, L., Wille, V., Lázaro., P.: Automatic diagnosis of mobile communication networks under imprecise parameters. Expert systems with Applications. **Vol.36 (1)**, 489–500 (2009)
3. Barco, R., Lázaro, P., Muñoz., P.: A unified framework for self-healing in wireless networks. IEEE Communications Magazine **50**, 134–142 (2012)
4. Barco, R., Lázaro, P., Wille, V., Díez, L., Patel, S.: Knowledge acquisition for diagnosis model in wireless networks. Expert systems with Applications. **Vol.36, Issue 3, Part 1**, 4745–4752 (2009)

5. Barco, R., Wille, V., Díez, L., Toril, M.: Learning of model parameters for fault diagnosis in wireless networks. Wireless Networks **16**, 255–271 (2010)
6. Bennacer, L., Ciavaglia, L., Chibani, A., Amirat, Y., Mellouk, A.: Optimization of fault diagnosis based on the combination of bayesian networks and case-based reasoning. In: IEEE Network Operations and Management Symposium (NOMS) (2012)
7. Breysse, D.: Forensic engineering and collapse databases. Proceedings of the Institution of Civil Engineers-Forensic Engineering **165**(2), 63–75 (2012)
8. COMMUNE: Commune (cognitive network management under uncertainty) (2012)
9. Fayyad, U., Irani, K.B.: Multi-interval discretization of continuous valued attributes for classification learning pp. 1022–1027 (1993)
10. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI magazine **17(3)** (2003)
11. Khatib, E., Barco, R., Gómez-Andrades, A., Muñoz, P., Serrano, I.: Data mining for fuzzy diagnosis systems in {LTE} networks. Expert Systems with Applications **42**(21), 7549 – 7559 (2015). DOI http://dx.doi.org/10.1016/j.eswa.2015.05.031. URL http://www.sciencedirect.com/science/article/pii/S0957417415003590
12. Liinasuo, M., Karvonen, H., Aaltonen, I., Fuentes, B., Castro, A.: Human operator trust in autonomic functionalities. In: Proceedings of the 30th European Conference on Cognitive Ergonomics, ECCE '12, pp. 45–51. ACM (2012)
13. NGMN: Use Cases Related to Self-Organising Network, Overall Description. Next Generation Mobile Networks (NGMN) Alliance, http://www.ngmn.org (2007)
14. Ramírez-Gallego, S., García, S., Mouriño-Talín, H., Martínez-Rego, D., Bolón-Canedo, V., Alonso-Betanzos, A., Benítez, J.M., Herrera, F.: Data discretization: taxonomy and big data challenge. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **6**(1), 5–21 (2016)
15. Santos, J.P., Alheiro, R., Andrade, L., Caraguay, V., Leonardo, Á., Barona López, L.I., Sotelo Monge, M.A., Garcia Villalba, L.J., Jiang, W., Schotten, H., et al.: Selfnet framework self-healing capabilities for 5g mobile networks. Transactions on Emerging Telecommunications Technologies **27**(9), 1225–1232 (2016)
16. SELFNET: Selfnet project. https://selfnet-5g.eu/ (2015)
17. Szilagyi, P., Novaczki, S.: An automatic detection and diagnosis framework for mobile communication systems. IEEE Transactions on Network and Service Management **9, no.2**, 184–197 (2012)
18. Terwel, K., Boot, W., Nelisse, M.: Structural unsafety revealed by failure databases. Proceedings of the ICE-Forensic Engineering, 167 (1), 2014 (2014)
19. Univerself: Univerself project. http://www.univerself-project.eu/ (2012)
20. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man and Cybernetics **22.6**, 1414–1427 (1992)
21. Wu, X., Chen, H., Wu, G., Liu, J., Zheng, Q., He, X., Zhou, A., Zhao, Z.Q., Wei, B., Gao, M., Li, Y., Zhang, Q., Zhang, S., Lu, R., Zheng, N.: Knowledge engineering with big data. IEEE Intelligent Systems **30**(5), 46–55 (2015)