Expert Systems with Applications 42 (2015) 7549–7559

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Data mining for fuzzy diagnosis systems in LTE networks

Emil J. Khatib [a], Raquel Barco [a,*], Ana Gómez-Andrades [a], Pablo Muñoz [a], Inmaculada Serrano [b]

[a] Universidad de Málaga, Communications Engineering Dept., Málaga, Spain
[b] Ericsson, PBO RA Continuous Analysis, Spain

ARTICLE INFO

Article history:
Available online 1 June 2015

Keywords:
Self-healing
Self-Organizing Networks
LTE
Data mining
Data driven learning
Supervised learning
Fault management
Fuzzy systems
Big Data

ABSTRACT

The recent developments in cellular networks, along with the increase in services, users and the demand of high quality have raised the Operational Expenditure (OPEX). Self-Organizing Networks (SON) are the solution to reduce these costs. Within SON, self-healing is the functionality that aims to automatically solve problems in the radio access network, at the same time reducing the downtime and the impact on the user experience. Self-healing comprises four main functions: fault detection, root cause analysis, fault compensation and recovery. To perform the root cause analysis (also known as diagnosis), Knowledge-Based Systems (KBS) are commonly used, such as fuzzy logic. In this paper, a novel method for extracting the Knowledge Base for a KBS from solved troubleshooting cases is proposed. This method is based on data mining techniques as opposed to the manual techniques currently used. The data mining problem of extracting knowledge out of LTE troubleshooting information can be considered a Big Data problem. Therefore, the proposed method has been designed so it can be easily scaled up to process a large volume of data with relatively low resources, as opposed to other existing algorithms. Tests show the feasibility and good results obtained by the diagnosis system created by the proposed methodology in LTE networks.

1. Introduction

In recent years, mobile communications have grown both in traffic volume and offered services. This has increased the expectations for quality of service. In this scenario, service providers are pressed to increase their competitiveness, and therefore to increase quality and reduce costs in the maintenance of their networks. This task can only be achieved by increasing the degree of automation of the network. The Next Generation Mobile Networks (NGMN) Alliance (NGMN, 2006) defined *Self-Organizing Networks (SON)* as a set of principles and concepts to add automation to mobile networks. Recently, these ideas were applied to *Long Term Evolution (LTE)* networks by the 3rd Generation Partnership Project (3GPP) (3GPP, 2012) in the form of use cases and specific SON functionalities. The SON concept is composed by three fields: *self-configuration*, *self-optimization* and *self-healing*. This paper is focused on self-healing, which includes all the functionality targeted towards automating troubleshooting in the radio access network (RAN). Currently, the task of RAN troubleshooting is manually done, so the ability of automating it is a great competitive advantage for operators. The benefits of self-healing are numerous, since it will offload troubleshooting experts of repetitive maintenance work and let them focus on improving the network. It also reduces the downtime of the network, therefore increasing the quality perceived by the users.

Self-healing is composed of four main tasks: fault detection, root cause analysis (diagnosis), compensation and fault recovery (Barco, Lázaro, & Muñoz, 2012). Currently there are no commercial implementations of these functions, although some research effort has been recently made.

The reason of this shortage of implementations, according to the COMMUNE project (COMMUNE, 2012) is the high degree of uncertainty in diagnosis in the RAN of mobile networks. The COMMUNE project uses a case based reasoning algorithm (Szilagyi & Novaczki, 2012) where a vector of Performance Indicators (PIs) is compared against a database of known problems. The cause will be the same as the case of the nearest stored problem.

In Barco, Díez, Wille, and Lázaro (2009) a Bayesian Network (BN) is used to do the diagnosis. To implement the system, it is required that an expert sets the parameters of the BN, so a Knowledge Acquisition Tool is also presented in Barco, Lázaro, Wille, Díez, and Patel (2009).

* Corresponding author.
E-mail addresses: emil@uma.es (E.J. Khatib), rbarco@uma.es (R. Barco), aga@ic.uma.es (A. Gómez-Andrades), pabloml@ic.uma.es (P. Muñoz), inmaculada.serrano@ericsson.com (I. Serrano).

http://dx.doi.org/10.1016/j.eswa.2015.05.031
0957-4174/

The UniverSelf project (Univerself, 2012) combines BNs with case-based reasoning (Bennacer, Ciavaglia, Chibani, Amirat, & Mellouk, 2012; Hounkonnou & Fabre, 2012) for the diagnosis process. Apart from the previous references in diagnosis, research in self-healing has also been extended to detection (Asghar, Hamalainen, & Ristaniemi, 2012; Barreto, Mota, Souza, Frota, & Aguayo, 2005) and compensation (Eckhardt et al., 2011; Razavi, 2012), which are out of the scope of this paper.

This paper proposes a method for learning troubleshooting rules for diagnosis methods based on Fuzzy Logic Controllers (FLCs) (Lee, 1990). FLCs use fuzzy logic (Zadeh, 1965) to assign fuzzy values to numerical variables, and by applying fuzzy rules, they obtain the value of an output variable (e.g. a parameter value) or a given action. FLCs have been used for diagnosis in other fields, such as industrial processes (Serdio, Lughofer, Pichler, Buchegger, & Efendic, 2014), machinery operation (Serdio et al., 2014; Lughofer & Guardiola, 2008) or medical diagnosis(Innocent, John, & Garibaldi, 2004). Although FLCs have been used in mobile networks for self-optimization (Muñoz, Barco, & de la Bandera, 2013a; Muñoz, Barco, & de la Bandera, 2013b), there are no previous references proposing its application to self-healing.

The implementation of the diagnosis process is done by Knowledge-Based Systems (KBS) (Akerkar & Sajja, 2010; Triantaphyllou & Felici, 2006) such as BNs and FLCs. KBS are composed of two main parts: the Inference Engine (IE) and the Knowledge Base (KB). This separation permits the algorithm to be used in a variety of situations by changing only the KB. Still, the construction of the KB, or Knowledge Acquisition (KA) (Triantaphyllou & Felici, 2006; Maier, 2007) is a major research topic. This is where the previous KA proposals in literature for diagnosis in wireless networks fail to deliver convincing results. The KA process involves the troubleshooting experts in a time consuming process (Studer, Benjamins, & Fensel, 1998; Ruiz-Aviles, Luna-Ramírez, Toril, & Ruiz, 2012; Chung, Chang, & Wang, 2012; Barco et al., 2009). It is based on the fact that the knowledge is contained in the experience of the expert. An alternative approach (Triantaphyllou & Felici, 2006) considers that the knowledge applied by the experts in the troubleshooting process will also be contained in its results. Every pair composed of the PIs and the fault cause and/or action(s) to be taken provided by the expert contains information about what aspects are observed by the expert and how they are related to each other. Therefore, a problem database (Hatamura, Ilno, Tsuchlya, & Hamaguchi, 2003) can be created, where each problem is saved along with its diagnosis; and this database will hold the expert's knowledge implicitly.

Data mining (DM) consists of the discovery of patterns in large data sets through the application of specific algorithms (Kantardzic, 2011; Witten, Frank, & Mark, 2011; Han, Kamber, & Pei, 2012). The result of DM is a model of the studied system. DM is used to process information from sensor networks (Papadimitriou, Brockwell, & Faloutsos, 2003), in scientific data collection (Ball & Brunner, 2010) or computer science (Ektefa, Memar, Sidi, & Affendey, 2010). DM is also used in commercial applications to find marketing trends (Linoff & Berry, 2011). Modern data collection and monitoring systems produce large amounts of data containing valuable knowledge, but due to the huge amount of information, this knowledge is hidden and needs to be extracted and easily visualized. In fact, the traditional DM techniques are often insufficient or ineffective for the large amount of available data. This leads to the development of Big Data techniques (Russom, 2011), which deal with databases that have special requirements due to one or more of the following factors, known as the *3 Vs of Big Data*:

- Volume: the amount of data generated may require new transmission, storage and processing techniques. In the case of mobile networks, data is produced by each network element (e.g. eNodeB) over all the network, but also by each connected terminal.
- Velocity: usually there are constraints on the time when the analysis results must be available, so that the information can be used in real time. In mobile networks, part of the data is produced in streams, that is, continuously as new events happen while the users access the services. In order to reduce processing times, heavy parallelization is very often the best solution.
- Variety: data is not always normalized and clearly structured. Also, formats vary depending on the nature of the data and the element that produces it.

In this work, a DM algorithm for obtaining fuzzy rules in the "*if ... then ...*" form for the diagnosis of the RAN of LTE networks is proposed, which relates certain behaviors in the PIs of a sector at a given time to the possible problem that is present in it. The rules reflect the knowledge of the experts that is implicitly contained in the results of their work (a database containing the solved problems). Other learning algorithms have been proposed for fuzzy rules in other fields, but none of them have been adapted to mobile communication networks (Wang & Mendel, 1992; Cordon, Herrera, & Villar, 2011; Lughofer & Kindermann, 2010; Pratama, Anavatti, Angelov, & Lughofer, 2014; Chen, 2013; Babuska, 1998). This algorithm has been designed to be easily parallelizable in order to fit the velocity requirements described earlier and to minimize the memory footprint so a large volume of data can be processed.

The remaining of this paper is organized as follows. In Section 2 the traditional processes of troubleshooting is presented, along with the use of KBS for automating it. In Section 3 the proposed DM algorithm for automating the KA process is presented. Afterwards, in Section 4, the system is tested and its performance is compared with another learning algorithm. Finally, the conclusions are discussed in Section 5.

2. Problem formulation

2.1. Troubleshooting in LTE networks

The process of troubleshooting is made up of four main tasks:

- Detection: determining that there is a problem in the network and isolating its origin, that is, the sectors that are degrading the performance of the whole network.
- Compensation: reconfiguration of the neighboring sectors to cover the affected users.
- Diagnosis: determining the cause of the problem.
- Recovery: actions to restore the affected sector to a normal state.

This process is usually manually done. Experts monitor a reduced set of Key Performance Indicators (KPIs), aggregated over the whole network. KPIs are composite variables that contain the information of several low-level PIs and reflect the general behavior of the sector. When one (or several) of those KPIs is degraded (e.g. it is lower than a certain threshold), a list of the worst offenders is obtained, showing the sectors that more strongly degrade the KPI. Those sectors are then diagnosed by observing further symptoms (i.e. abnormal values of PIs and other low-level metrics, such as counters, measurements and alarms). The relations among these symptoms, described by heuristic "*if ... then ...*" rules, determine

the diagnosis. This task is a *classification* process, where each case (a set of PIs of a sector on a given time interval) is assigned a class describing a specific cause. With the result of the diagnosis, corrective actions are taken and the process is repeated if the sector is not fixed.

2.2. Knowledge based systems

The process of diagnosis, as explained above, can be replicated with an FLC that applies fuzzy rules over a subset of PIs of the sector in order to classify the case. Since FLCs are KBS, they have a separation between the IE and the KB. The KB is divided in two parts:

- Data Base (DB): information about the variables that the system works with. In the case of mobile network troubleshooting, it contains the descriptions of the PIs and the diagnosed causes.
- Rule Base (RB): information about the relations between the variables and the output. The heuristic rules used for troubleshooting compose the RB.

In this work, we will use DM only to obtain the RB. The DB is usually provided by the experts or by the specifications of the individual PIs. Typically, there are two ways for acquiring the knowledge needed in the RB from the experts:

- By interviewing them. This option involves a knowledge engineer (Maier, 2007) who has the knowledge about the expert system 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 expert system so the knowledge is directly coded by him. It is even more intrusive than the previous option and involves a steep learning curve.

These methods of KA rely heavily on the experts 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. In Section 3.2 an algorithm for extracting the knowledge from a set of solved troubleshooting cases is proposed. The output of the algorithm is a RB that can be used in an FLC to classify cases and therefore automate the diagnosis process.

2.3. Fuzzy Logic Controllers

FLCs imitate the process of human thinking, focusing on two of its main aspects:

- Humans perceive and operate with fuzzy values. Instead of numerical values, human thought tends to classify the value of a variable in a fuzzy set.
- Human thought uses heuristic "*if* ... *then* ..." rules for inference of new concepts based on existing fuzzy values.

A properly configured FLC can be easily interpretable (Casillas, 2003; Lughofer, 2013; Gacto, Alcalá, & Herrera, 2011), that is, it is given in a simple and understandable way. Interpretability of a model is important since it can be more easily used and modified when needed. Also, an interpretable FLC may offer information about the inner workings of the modelled system, helping to expand human understanding about it.

FLCs have three parts; a *fuzzification* block, that translates from numerical *crisp* values to fuzzy values, an *inference* block, that applies fuzzy rules on the fuzzified input variables to obtain a fuzzy output, and a *defuzzification* block that transforms the fuzzy output of the inference block into a crisp value or a specific action. Fig. 1 depicts the typical contents of an FLC.

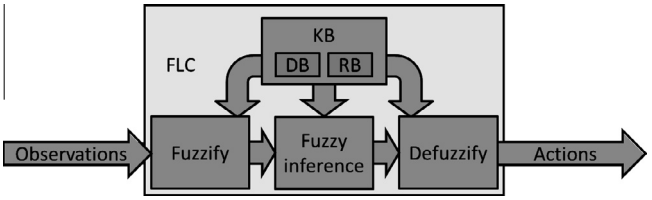


Fig. 1. Scheme of a typical FLC.

Knowledge is used in the process of fuzzification in the form of fuzzy set definitions (the DB) and in the form of rules (the RB) in the IE.

The diagnosis process in troubleshooting of LTE networks is basically a process of applying heuristic rules on PIs, alarms, counters, configuration parameters, etc. to obtain a diagnosis. Therefore, the process will start with the fuzzification of the input variables to, typically, two fuzzy sets such as *high*/*low* or *true*/*false*. The membership functions of these sets will be trapezoidal (Fig. 2), since only two thresholds are needed. Experts use this two threshold classification for the KPI values very often, so the use of trapezoidal sets is the most natural choice for the task of diagnosis. Subsequently, the fuzzy rules will use the fuzzified values to assign a validity to each of the possible causes, depending on the *degree of activation*. Finally, in the process of defuzzification the most likely cause will be selected. The validity assigned in the previous step indicates the confidence that the chosen diagnosis is correct.

The fuzzy rules are composed of two parts: the *antecedent* (the "*if* ..." part) and the consequent (the "*then* ..." part). The antecedent is made up of individual assertions about the fuzzy variables (such as "*accessibility is low*"). Each assertion has a *degree of truth* equal to the degree of membership of the value of the variable to the assigned fuzzy set. Assertions are joined by *AND* (*t-norm*) or *OR* (*s-norm*) operators. In the scope of this study, only *AND* (Klement, Mesiar, & Pap, 2010) operators are used, since they provide the basic functionality for pattern recognition on the PI observations, whereas the *OR* operator can be omitted safely. The only case where the *OR* operator would be needed is where two different combinations of PI values in the antecedent identify the same problem. In that case, two separate antecedents can be created, thus sparing the use of *OR* operators and simplifying the learning process. With the *AND* operator, the degree of truth of the whole antecedent is either the minimum (in this work, this option is chosen) or the product of the degrees of truth of its assertions (in the case of *OR* operators, it would be the maximum). The consequent assigns the degree of truth of the antecedent to an assertion about the diagnosis belonging to a certain class (such as "*diagnosis is overload*").

A full example of the application of several rules on an input vector of KPIs is shown in Fig. 2. An input vector with four variables is introduced in the FLC. In this example, each variable takes values between 0 and 1. The variables are then fuzzified, each of them in two sets, either *High* or *Low*. The membership functions defined on the variables are the same for simplicity in this example. Once the vector has been fuzzified, the rules are applied. Note that for each rule, the degree of activation is given by the minimum value of the membership degrees in the antecedent. The second rule has the biggest degree of activation, therefore, its consequent will be the diagnosis. The degree of activation expresses the confidence of the diagnosis; therefore, the higher the degree of activation, the lower the uncertainty of the result. For simplicity, only the diagnosis with the highest score is used, although in cases where more than one root cause is present, several rules should have the same

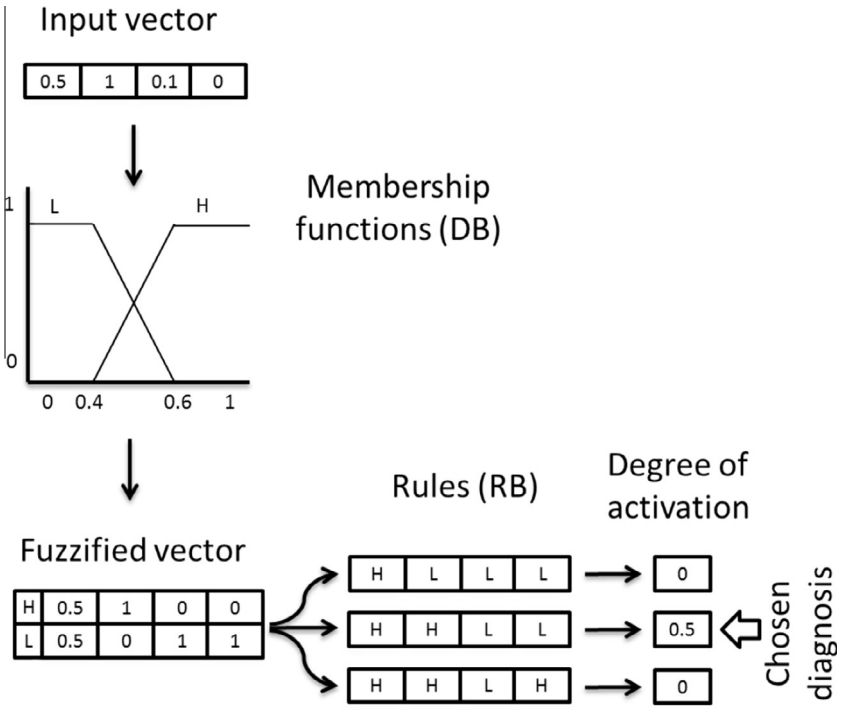


Fig. 2. Rule application process.

score. Even if only one diagnosis has the maximum score, it is still useful to provide a list of all the activated diagnoses ordered by descending score so the experts have a broader picture of the state of the sector. Another alternative to the scoring approach used in this study is weighted voting (Lughofer & Buchtala, 2013), where each diagnosis is compared individually with each of the other diagnoses through a single rule with two possible outcomes. Each rule will then *vote* for one of two possible diagnoses.

3. Data mining in LTE troubleshooting databases

3.1. Data mining

DM is the process that extracts a model out of a data set by exploring the underlying patterns. Knowledge Discovery and Data Mining (KDD) (Maimon & Last, 2001) is the larger process extending from the collection of the relevant data saved into a data base to the interpretation of the knowledge contained in it.

The DM algorithm proposed in this work obtains fuzzy rules from a set of solved cases (*training set*). These rules will take a set of PIs as input and produce a diagnosis (they will *classify* the cases). Therefore, the DM algorithm will solve a *classification* problem. The *learned* model (the fuzzy rule set) is a *classifier*. Ideally, the learned rules will be similar to those used by the experts. Each entry in the training set (case) is 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 (in this study, this is the diagnosis). Since the training set contains the class label, the DM algorithm is a *supervised learning* algorithm.

To obtain the required vectors, a previous process of *data reduction* (Khatib, Barco, Serrano, & Muñoz, 2014) must be carried out. The data extracted from the network will be a set of time-dependent arrays representing time series for each PI. This time series covers the time interval where the problem occurred, but also the surrounding time intervals where the fault did not show up in the data. Therefore, two steps must be achieved:

- Isolating the problem in time, so the data reflects only the influence of the root cause.
- Converting the set of time series to a vector with one value per PI. Once the problem is isolated, this can be done with an average value of the PI for the affected time interval.

The issue of data reduction is out of the scope of this article, but it is a necessary prior step to the data driven learning process.

3.2. Data driven learning in troubleshooting databases

In this study, a novel supervised data driven learning algorithm for wireless networks based on the WM (Wang & Mendel, 1992) method is described. It introduces some modifications 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 processes arbitrarily without loss of information. 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. Other learning algorithms have been proposed (Cordón, Herrera, Gomide, Hoffmann, & Magdalena, 2001; Khatib, Barco, Góomez-Andrades, & Serrano, in press; Angelov, Lughofer, & Zhou, 2008; Ishibuchi & Nakashima, 2001), but the WM method was found the best for parallelization and for its deterministic behavior, which helps understanding the results of the learning process.

The algorithm obtains the RB of an FLC from a training set composed of labeled cases. The learning cases are tuples $C = (\overline{k}, d)$ composed of a vector $\overline{k} = \{k_1, k_2 \ldots k_N\}$ as the attribute vector representing the values of PIs $\overline{PI} = \{PI_1, PI_2 \ldots PI_N\}$ and a label d as the class label among possible root causes $\overline{RC} = \{RC_1, RC_2 \ldots RC_M\}$. 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 $k_{F_n} = T_n | \mu_{T_n}(k_n) = max(\mu_{T_{1n}}(k_n), \mu_{T_{2n}}(k_n))$ 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 $RC | \mu_{RC}(d) = max(\mu_{RC_m}(d)) \forall RC_m \in \overline{RC}$ representing the root cause. This process is depicted with an example in Fig. 3. 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 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*. 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 $B = 1 - (1/(1 + \alpha n_c))$, where n_c is the number of cases covered by the antecedent 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 $S = n_s/n_c$. 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 $D = B \cdot S$. 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 $B = \sum_{w=1}^{W} B_w$ where 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*. In troubleshooting of LTE networks, it is common that the experts use incomplete rules, that is, rules that only focus on a subset of available PIs and ignore the values of the rest. Incomplete rules are also more comprehensive and therefore more interpretable. To obtain these types of 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 process is depicted in an example in Fig. 4, where variable PI_1 has two sets (T_{11} and T_{12}). As both rules are exactly equal with the exception of the values assigned to PI_1, this variable can be safely ignored. This step 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 Fig. 5 the flowchart of the algorithm is depicted.

3.3. Parameters

The algorithm has three parameters:

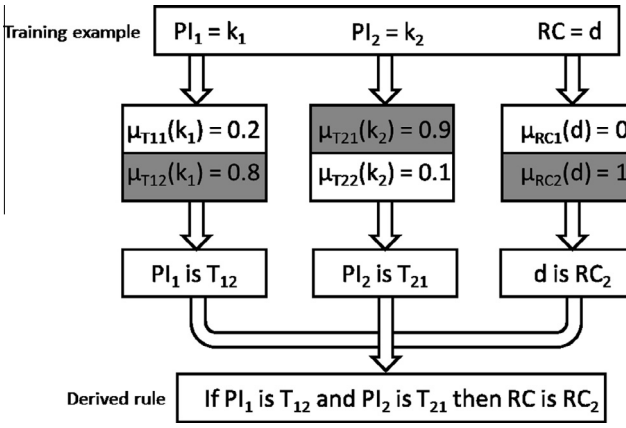


Fig. 3. Rule creation process. A training set with two input variables (x and y) and one output variable (z) is depicted. These variables take values x_1, y_1 and z_1 respectively. The truth degree for each fuzzy set is calculated for these values, and the input variables are assigned the set with the highest truth degree. With these sets, a rule is created.

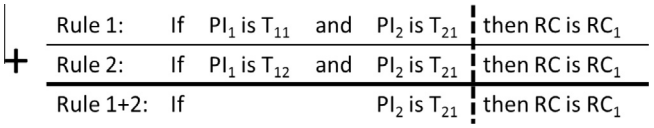


Fig. 4. Example of fusion of two rules.

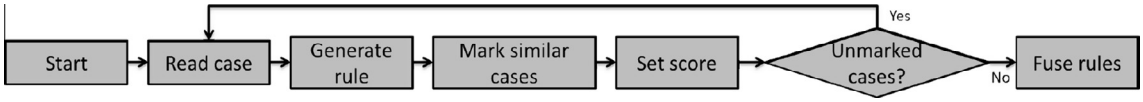


Fig. 5. Flowchart of the algorithm.

- *Uncommon case sensitivity* (α) $\in (0, +\infty)$: it adjusts the sensitivity of the algorithm to rare cases. A low value will make the algorithm more robust to spurious training errors, but may also make it ignore relevant (but uncommon) cases, therefore producing an output RB that will fail to detect these situations.
- *Minimum inclusion score (MIS)* $\in (0, 1)$: minimum score for a rule to enter the final fusion step and be part of the RB (as a contribution to a general rule).
- *Minimum degree of activation (MDA)* $\in (0, 1)$: minimum degree of activation of a rule on a case to consider it covered, that is, that the rule antecedent is activated by the case.

4. Evaluation

4.1. Case study

Although the principle of a fault database (Hatamura et al., 2003) is simple, the troubleshooting process carried out by LTE experts currently does not include a problem collection step where solved cases are saved along with their solution. Therefore, newly developed diagnosis systems need to somehow generate these cases in order to validate their operation.

In this work, the training cases are generated by a network emulator based on the knowledge of troubleshooting experts. The emulator provides the required cases (PI vectors and fault cause). The advantage of this method is that it can provide as many cases as necessary, while being close enough to what a real case extracted from the network would be.

To define the emulator, troubleshooting experts were asked to define the most common problem categories in LTE, and their related PI values. The frequency of occurrence of each type of problem was defined and the Probability Density Function (PDF) of each PI conditioned to the presence of each cause was modeled. With the information of the probability of occurrence of each case and the PDF of each PI, the LTE emulator creates two sets of solved cases: the training cases and the testing cases that will be used to evaluate the accuracy of the algorithm.

The fault categories defined by the experts, and their probability of occurrence (given that there is a problem), are the following:

- **SW problem (13%)**: The root cause is located in the software of the site. Performance is degraded despite normal radio conditions.
- **Coverage (25%)**: Some regions in the planned coverage area do not receive enough signal level from the site.
- **Quality (34%)**: There is interference in either the uplink or the downlink signal.
- **Mobility (28%)**: Problem with handovers to/from neighboring sectors.

These fault categories are complemented by an additional **normal** state that models the behavior of the network when there is no problem present. The probability of normal behavior for the study will be 70%, versus 30% of probability for the occurrence of a problem. The PIs that the emulator creates are the following:

- **Accessibility**: inverse of the blocking rate. It is defined as the ability to establish a connection in the network. It is considered high for a value greater than 99% and low for values under 98%. The *high* and *low* fuzzy sets are modeled by trapezoidal functions, as depicted in Fig. 2.
- **Retainability**: inverse of the dropped call rate. It is defined as the ability to end a call correctly. The fuzzy set membership functions have the same parameters as Accessibility.
- **Handover Success Rate (HOSR)**: percentage of initiated handovers that end successfully. Values under 95% are considered low, and over 98.5%, high.
- **95 percentile of Reference Signal Received Power (RSRP)**: value of RSRP under which 95% of the samples fall. Values below −130 dBm are low and over −100 dBm, they are high.
- **95 percentile of Reference Signal Received Quality (RSRQ)**: value of RSRQ under which 95% of the samples fall. Values under −30 dB are considered low and over −12 dBm high.

The thresholds for Accessibility, Retainability and HOSR are obtained from commercial requirements, whereas the RSRP and RSRQ are obtained from experts. The PDFs of each PI conditioned to the occurrence of a root cause are described in Table 1. The PDFs of bounded PIs (Accessibility, Retainability and HOSR) are modeled as beta distributions, with two shape parameters, $\alpha, \beta > 0$. Unbounded PIs (RSRP and RSRQ), on the other hand, are modeled by gaussian distributions, and they are parametrized by the average and the standard deviation (σ).

4.2. Experimental design

The tests carried out on the algorithm evaluate the influence of its main parameters (Table 2). The tests determine the diagnosis accuracy of the FLC that uses the obtained RB.

For this purpose, three measurements are assessed:

- **Diagnosis Error Rate** (E_d): number of incorrect diagnosis over the total number of problems. This measurement indicates the accuracy of the classifier. It is given by $E_d = N_{de}/N_p$, where N_{de} is the number of problems wrongly diagnosed (that is, cases where the FLC provides a diagnosis that does not match the original diagnosis given in the training set) and N_p the total

Table 1
PDF of each PI conditioned to an existing problem.

PI	Type		Parameters/Cause				
			Nor	Cap	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
RSRP	norm	avg	−6.5	−6	−10	−13	−11
		σ	1.1	2	5	2	3

Table 2
Parameter values.

Test	Variable	Default value	Tested values
1	MDA	0.5	0:0.1:1
2	α	0.04	0.01:0.01:0.2
3	MIS	0.1	0.1:0.1:0.9

number of problems in the validation set. This measurement does not count the problems that are diagnosed as normal (false negatives) nor the false positives.

- **Undetected Rate** (E_u): number of problems that are not diagnosed at all over the total number of problems. This measurement indicates the reliability of the FLC, that is, its ability to detect a problem. It is given by $E_u = N_{un}/N_p$, where N_{un} is the number of problematic cases that are classified as normal.
- **False Positive Rate** (E_{fp}): number of normal cases that are diagnosed as a problem over the number of normal cases. A high False Positive Rate reduces the performance of the FLC, since there is a high chance of false alarms. It is given by $E_{fp} = N_{fp}/N_n$, where N_{fp} is the number of normal cases diagnosed as problematic and N_n is the total number of normal cases. Note that even a relatively low E_{fp} can be translated into a high absolute number of false positives in the output if the number of normal cases is much higher than the number of problems, which is the most likely scenario in a production network.

The meaning and importance of these errors vary depending on if the detection phase (determining if a specified case is normal or problematic) relies on the same FLC that will do the diagnosis. If the FLC also does the detection, the main objective should be to minimize the Undetected Rate. This is done usually at the cost of an increase in the False Positive Rate, because loosing the conditions for detecting a problem will cause an increase in the number of non-problematic cases that are wrongly detected.

The increased number of scenarios that the FLC must detect will also cause more normal cases to match them. If there is a previous detection step, independent from the diagnosis system, then the Undetected Rate represents the proportion of cases that cannot be diagnosed by the system, but still are detected and can be diagnosed manually. These measurements can be used to obtain other representative errors:

- Total error rate (E_p): probability that a problematic case in the input of the troubleshooting system produces a wrong diagnosis in the output. It is given by $E_p = E_d + E_u$.
- Overall error (E): probability that a specific diagnosis is wrong. It is given by $E = P_n \cdot E_{fp} + P_p \cdot E_p$, where P_n and P_p are respectively the proportion of normal and problematic cases in the validation set.
- Complementary of the Positive Predictive Value (P_{fp}): probability that a given positive diagnosis is a false positive, given by:

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

This probability shows the importance of a low False Positive Rate. A high P_{fp} will render the system unreliable, because far too many of the diagnosed problems are not real. Again, this problem would be solved if a previous detection phase is used, since it will discard all non-problematic cases, leaving only the problematic cases for diagnosis.

The tests have been carried out with two sets of cases generated by the emulator described in Section 4.1. A training set of 2000 cases (containing 600 problems and 1400 normal cases) and a testing set of 5000 cases (1500 problems and 3500 normal) have been generated. The algorithm proposed in this paper is first trained with the training cases, and afterwards, it is evaluated with the testing cases. By default, the normal cases are not used in the training stage, except for a specific test to find the impact of using them.

These results are compared with the performance of a supervised learning process for creating a Bayesian Network, using the software package GeNIe (Genie, 2014). The BN is trained and validated with the same set of emulated cases used for the data driven method. Two training sets are used for the BN training, one including the normal cases and one excluding them.

Finally, the algorithm will be tested with a reduced set of real cases to demonstrate its validity for real life scenarios. Since the number of available cases is small, a cross validation process will be used.

4.3. Results

Test 1: 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. Fig. 6 depicts the results.

It is observed that the three errors remain stagnant for values of MDA between 0.1 and 0.8. The minimum for the Undetected Rate is 0 for values of MDA of up to 0.5. Between 0.6 and 1, the value of the Undetected Rate increases slightly, up to 0.2%. The False Positive Rate has a constant value of 34.5% except for MDA = 1, where it decreases to 32.7%. The Diagnosis Error Rate reaches a local maximum of 27.5% at MDA = 0 and a minimum of 4.9% at MDA = 0.4. For MDA values above 0.8, the diagnostic error rate grows to a maximum of 18.5%. 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.

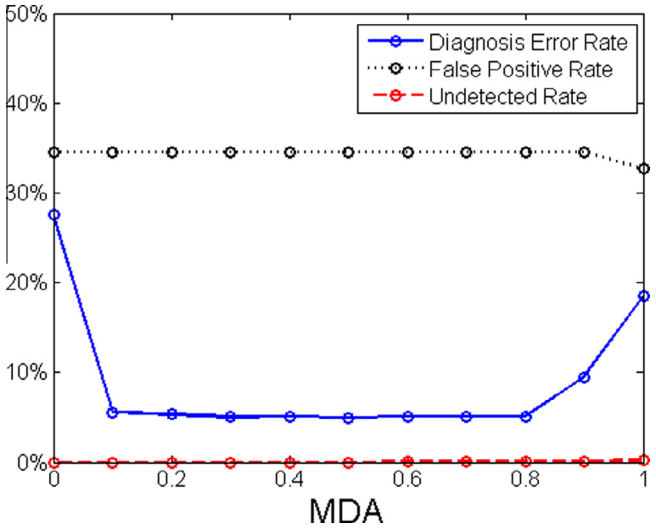


Fig. 6. Error rates for parameter MDA.

they cover cases that should not be covered) are created and included in the final rule set, therefore increasing the classification error.

The results show that the algorithm is insensitive to values of MDA in a wide range of values. For extreme values of MDA, the Diagnosis Error Rate increases. Therefore, the recommended values for this parameter are $0.1 \leqslant MDA \leqslant 0.8$. Specifically, for $MDA \leqslant 0.5$, the Undetected Rate reaches the minimum value of 0, so the optimal range of MDA is $0.1 \leqslant MDA \leqslant 0.5$.

Test 2: 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, undiagnosed rate and False Positive Rate are depicted in Fig. 7.

The Diagnosis Error Rate and Undetected Rate decrease as α increases. Both errors are maximum for $\alpha = 0.01$, 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 performance of the resulting FLC. To illustrate the relation between the False Positive Rate and the performance, by using Eq. (1) 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 $\alpha \geqslant 0.06$, since for these values, the Undetected Rate is 0, and the gain of using values of α between 0.02 and 0.05 in terms of false positives is small (in both cases, the False Positive Rate is too high and the system needs a previous detection stage).

Test 3: MIS

The MIS parameter determines the minimum score that a rule must have to be fused with other rules to produce more general rules. This process was described in Section 3.2. The results are shown in Fig. 8.

The Diagnosis Error Rate shows a stagnant behavior as MIS increases. For MIS = 1, the Diagnosis Error Rate is 0, but as observed in the Undetected Rate, there are no diagnosis in that case. MIS determines the minimum score that a rule must have to be included in the final RB, so a high value of MIS reduces the diversity of the RB. Since no rule reaches a score of 1, there is no output RB for MIS = 1.

As expected, since an increase in MIS 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 MIS increases, and the False Positive Rate decreases. The best interval for this parameter is $0.1 \leqslant MIS \leqslant 0.4$, because the Undetected Rate is minimum.

Test 4: Inclusion of 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 Fig. 9.

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 Eq. (1) 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 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.

4.4. Comparison with other technique

Bayesian Networks have also been previously proposed for troubleshooting in mobile networks (Barco et al., 2009). In this

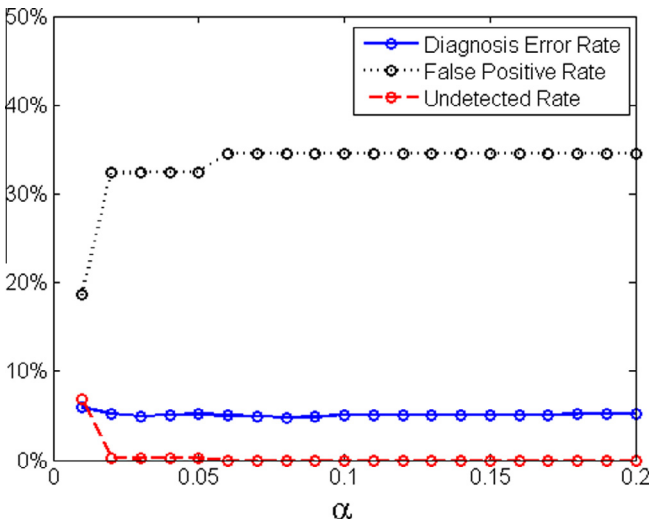


Fig. 7. Error rates for variable α.

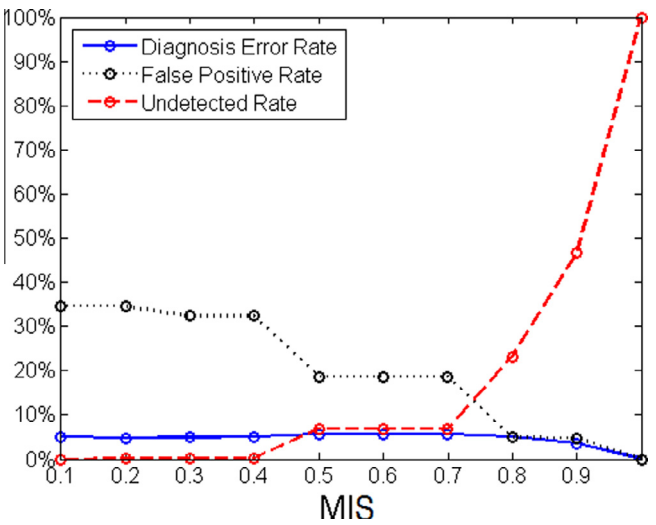
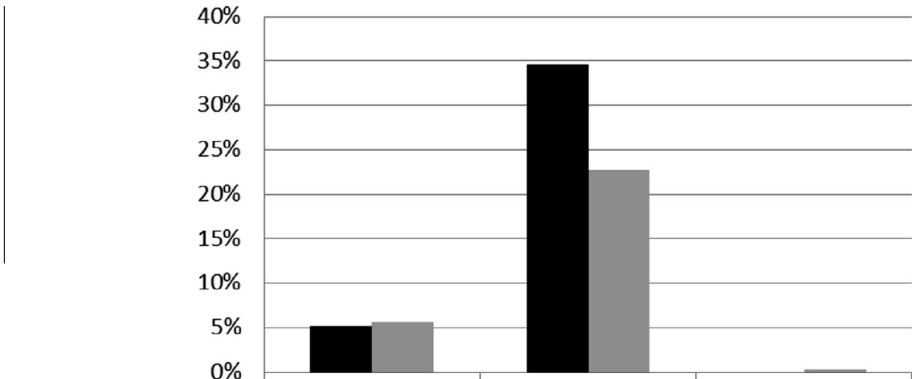


Fig. 8. Results for variable MIS.



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
Without normal cases	5.13%	34.51%	0.00%
With normal cases	5.67%	22.71%	0.27%

Fig. 9. Diagnosis error, undetected rate and false positive rate.

Section, the cases used in the data driven algorithm will be used for training a BN. Fig. 10 depicts the BN equivalent to the fuzzy system trained by the proposed data driven algorithm.

The BN is designed, trained and validated in the GeNIe software package (Genie, 2014). 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 4.1. The discretized cases are then used for supervised learning.

The results are shown in Fig. 11. Two different tests are done, one excluding normal 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 data driven method produces a more accurate diagnosis system, with a Diagnosis Error Rate of 5.1% versus 21.9%, with a lower False Positive Rate (34.5% versus 100%). The BN can achieve a lower False Positive Rate than the data driven algorithm with the cost of increasing its Diagnosis Error Rate and Undetected Rate. In addition, the use of fuzzy logic has other side advantages over BN, such as producing understandable rules and simplifying the process of integration of learned and manually elicited rules.

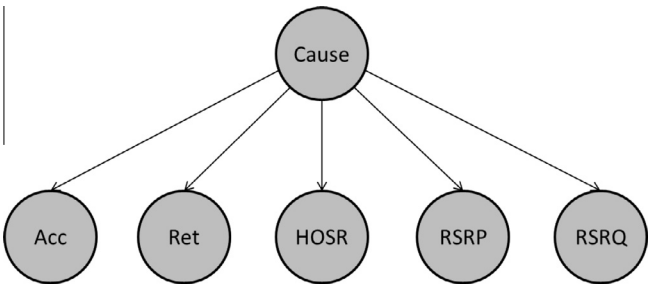


Fig. 10. BN used for troubleshooting.

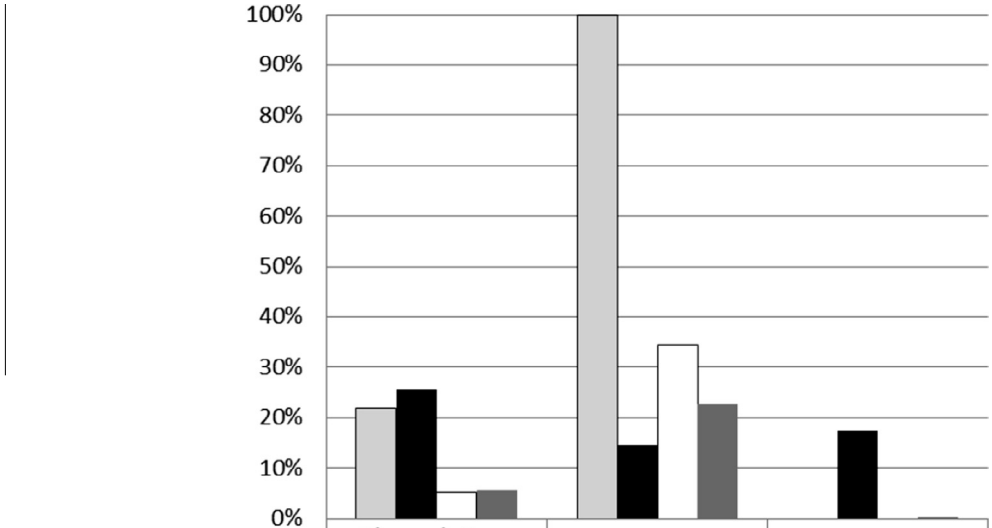
4.5. Results in live LTE network

In order to demonstrate the validity of the algorithm for real world applications, we will use it to learn rules on a reduced set of cases from a real LTE network.

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 modified in order to better understand the behavior 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). One of these cases is depicted in Fig. 12. The time series of the first four PIs is shown, with the interval where the CPU overload indicator is active marked in gray. In this case a traffic surge caused a high CPU usage, leading to dropped calls (low retainability) and rejected connections (low accessibility). Handovers to other sectors were not affected. The RSSI also grew due to the large number of users.

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.

With the algorithm parameters adjusted to the default values indicated in Table 2 (MDA = 0.4, α = 0.14, MIS = 0.1), the obtained Diagnosis Error Rate is 0, the Undetected Rate 14.3% and the False Positive Rate 0. The Diagnosis Error Rate shows that the system is very accurate when a diagnosis is produced, given that the case is not a normal case. The reason of the relatively high (compared to the emulated cases tests) Undetected Rate is 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*



	Diagnosis Error Rate	False Positive Rate	Undetected Rate
BN without normal	21,80%	100%	0,00%
BN with normal	25,50%	14,51%	17,40%
Data driven without normal	5,13%	34,51%	0,00%
Data driven with normal	5,67%	22,71%	0,27%

Fig. 11. Results for BNs trained with supervised learning compared to FLCs trained with the data driven method.

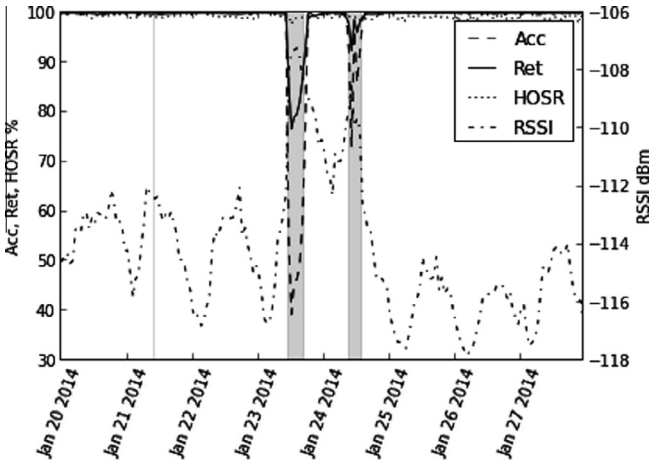


Fig. 12. Evolution of PIs over time in a sector with high CPU usage.

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 MIS 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 MIS settings, the Diagnosis Error Rate is still 0, the Undetected Rate is slightly reduced to 10% and the False Positive Rate remains 0.

5. Conclusions

A novel DM algorithm for obtaining the RB of FLCs from a set of solved cases for mobile network troubleshooting has been presented. The proposed method is a supervised, data driven learning algorithm, that uses vectors of PI values, alarms and configuration parameters labeled with the problem present at that time, as diagnosed by experts. There are currently no learning algorithms devoted to extracting troubleshooting rules from real troubleshooting cases. In the case of LTE mobile networks, the task of extracting troubleshooting rules is a Big Data problem, due to a high volume of data, a high speed of data generation and the high variability in the format and values of the variables. The presented algorithm is designed to be easily parallelizable, so it can perform the learning process over a large dataset in a limited time.

The described learning algorithm must be embedded in a data processing pipe that includes a prior step of data reduction and formatting, since the inputs to the algorithm do not follow any standard data format that is used in PI collecting systems.

Experiments that cover the three main configuration parameters of the algorithm and the presence or absence of normal cases in the training set have been performed. The results show that the parameters should allow diversity in the RB; that is, they must not filter rare cases or cases that loosely follow a rule. If the parameters are too restrictive, the output RB is unreliable, in the sense that it will often be unable to diagnose (or even detect, if the FLC is assigned this task) a proportion of the problems. On the other hand, the increase in reliability comes at a cost; the fuzzy rule

set will detect some normal cases as problems. This problem is solved if there is a detection stage, which is normally the case, that filters out normal cases prior to its analysis by the diagnosis stage. In addition, it has been demonstrated that in a wide range results are quite insensitive to the selection of the parameters, so the algorithm is fairly robust and does not require a very complex fine tuning. The algorithm has also been compared with another supervised learning algorithm based on BN, showing that the proposed method outperforms that based on BN. Finally, the algorithm has been tested in a live LTE network, showing promising results.

Acknowledgment

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

References

3GPP, (2012). Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2, version 11.4.0. Next Generation Mobile Networks (NGMN) Alliance (ts 36.300 ed.).

Akerkar, R., & Sajja, P. (2010). *Knowledge-based systems*. Jones & Bartlett Publishers.

Angelov, P., Lughofer, E., & Zhou, X. (2008). Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems, 159*, 3160–3182.

Asghar, M., Hamalainen, S., & Ristaniemi, T., (2012). Self-healing framework for LTE networks. In *IEEE 17th international workshop on computer aided modeling and design of communication links and networks (CAMAD)*.

Babuska, R. (1998). *Fuzzy modeling for control*. Kluwer Academic Publishers.

Ball, N. M., & Brunner, R. J. (2010). Data mining and machine learning in astronomy. *International Journal of Modern Physics D, 19*(07), 1049–1106.

Barco, R., Díez, L., Wille, V., & Lázaro, P. (2009). Automatic diagnosis of mobile communication networks under imprecise parameters. *Expert systems with Applications, 36*(1), 489–500.

Barco, R., Lázaro, P., & Muñoz, P. (2012). A unified framework for self-healing in wireless networks. *IEEE Communications Magazine, 50*, 134–142.

Barco, R., Lázaro, P., Wille, V., Díez, L., & Patel, S. (2009). Knowledge acquisition for diagnosis model in wireless networks. *Expert systems with Applications, 36*(3, Part 1), 4745–4752.

Barreto, G. A., Mota, J. C. M., Souza, L. G. M., Frota, R. A., & Aguayo, L. (2005). Condition monitoring of 3g cellular networks through competitive neural models. *IEEE Transactions on Neural Networks, 16*(5), 1064–1075.

Bennacer, L., Ciavaglia, L., Chibani, A., Amirat, Y., & Mellouk, A., (2012). Optimization of fault diagnosis based on the combination of Bayesian networks and case-based reasoning. In *IEEE network operations and management symposium (NOMS)*.

Casillas, J. (2003). *Interpretability issues in fuzzy modeling* (Vol. 128). Springer Science & Business Media.

Chen, M.-Y. (2013). A hybrid ANFIS model for business failure prediction utilizing particle swarm optimization and subtractive clustering. *Information Sciences, 220*, 180–195.

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

COMMUNE, (2012). Commune (cognitive network management under uncertainty).

Cordón, O., Herrera, F., Gomide, F., Hoffmann, F., & Magdalena, L. (2001). Ten years of genetic fuzzy systems: Current framework and new trends. *Joint 9th IFSA World Congress and 20th NAFIPS International Conference, 2001* (Vol. 3, pp. 1241–1246). IEEE.

Cordon, O., Herrera, F., & Villar, P. (2011). 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.

Eckhardt, H., Klein, S., & Gruber., M. (2011). Vertical antenna tilt optimization for LTE base stations. In *IEEE 73rd vehicular technology conference (VTC Spring)*.

Ektefa, M., Memar, S., Sidi, F., & Affendey, L., (2010). Intrusion detection using data mining techniques. In *International conference on information retrieval & knowledge management (CAMP)* (pp. 200–203).

Gacto, M. J., Alcalá, R., & Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences, 181*, 4340–4360.

Genie. (2014). *Genie modeling environment*. <http://genie.sis.pitt.edu/>.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques*. Waltham, MA: Morgan Kaufman.

Hatamura, Y., Ilno, K., Tsuchlya, K., & Hamaguchi, T. (2003). Structure of failure knowledge database and case expression. *CIRP Annals-Manufacturing Technology, 52.1*, 97–100.

Hounkonnou, C., & Fabre., E., (2012). Empowering self-diagnosis with self-modeling. In *Eighth international conference network and service management (CNSM), and 2012 workshop on systems virtualization management (SVM)*.

Innocent, P. R., John, R. I., & Garibaldi, J. M. (2004). Fuzzy methods for medical diagnosis. *Applied Artificial Intelligence, 19*, 69–98.

Ishibuchi, H., & Nakashima, T. (2001). Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems, 9*, 506–515.

Kantardzic, M. (2011). *Data mining: Concepts, models, methods, and algorithms*. John Wiley & Sons.

Khatib, E., Barco, R., Serrano, I., & Muñoz, P., (2014). LTE performance data reduction for knowledge acquisition. In *Globecom workshops (GC Wkshps)* (pp. 270–274).

Khatib, E.J., Barco, R., Góomez-Andrades, A., & Serrano, I. (in press). Diagnosis based on genetic fuzzy algorithms for LTE self-healing. *IEEE Transactions on Vehicular Technology, 12*.

Klement, E. P., Mesiar, R., & Pap, E. (2010). *Triangular Norms. Trends in Logic*. Netherlands: Springer.

Lee, C.-C. (1990). Fuzzy logic in control systems: fuzzy logic controller. *IEEE Transactions on I. Systems, Man and Cybernetics, 12*.

Linoff, G. S., & Berry, M. J. (2011). *Data mining techniques: For marketing, sales, and customer relationship management*. John Wiley & Sons.

Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues. *Information Sciences, 251*, 22–46.

Lughofer, E., & Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Transactions on Fuzzy Systems, 21*, 625–641.

Lughofer, E., & Guardiola, C. (2008). On-line fault detection with data-driven evolving fuzzy models. *Control and Intelligent Systems, 36*, 307.

Lughofer, E., & Kindermann, S. (2010). Sparsefis: Data-driven learning of fuzzy systems with sparsity constraints. *IEEE Transactions on Fuzzy Systems, 18*, 396–411.

Maier, R. (2007). *Knowledge management systems: Information and communication technologies for knowledge management*. Springer.

Maimon, O., & Last, M. (2001). *Knowledge discovery and data mining*. Academic Publishers.

Muñoz, P., Barco, R., & de la Bandera, I. (2013a). On the potential of handover parameter optimization for self-organizing networks. *IEEE Transactions on Vehicular Technology, 62*(5), 1895–1905.

Muñoz, P., Barco, R., & de la Bandera, I. (2013b). Optimization of load balancing using fuzzy q-learning for next generation wireless networks. *Expert Systems With Applications, 40*(4), 984–994.

NGMN, (2006). White Paper Next Generation Mobile Networks Beyond HSPA & EVDO, Version 3.0. Next Generation Mobile Networks (NGMN) Alliance. <www.ngmn.org>.

Papadimitriou, S., Brockwell, A., & Faloutsos, C., (2003). Adaptive, hands-off stream mining. In *Proceedings of the 29th international conference on very large data bases* (Vol. 29, pp. 560–571).

Pratama, M., Anavatti, S. G., Angelov, P. P., & Lughofer, E. (2014). PANFIS: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems, 25*, 55–68.

Razavi, R., (2012). Self-optimisation of antenna beam tilting in LTE networks. In *IEEE 75th vehicular technology conference (VTC spring)*.

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

Russom, P. (2011). *Big data analytics*. TDWI Best Practices Report, Fourth Quarter.

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

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

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering, 25*, 161–197.

Szilagyi, P., & Novaczki, S. (2012). An automatic detection and diagnosis framework for mobile communication systems. *IEEE Transactions on Network and Service Management, 9*(2), 184–197.

Triantaphyllou, E., & Felici, G. (2006). *Data mining and knowledge discovery approaches based on rule induction techniques*. Springer.

Univerself. (2012). *Univerself project*. <http://www.univerself-project.eu/>.

Wang, L.-X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics, 22.6*, 1414–1427.

Witten, I. H., Frank, E. H., & Mark, A. (2011). *Data mining: Practical machine learning tools and techniques/...*. Burlington, MA: Morgan Kaufman.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.