Degradation Detection Algorithm for LTE Root Cause Analysis

***Abstract*—Self-Organizing Networks (SONs) aim to automate network Operation & Maintenance (O&M) tasks. SONs comprise Self-configuration, Self-optimization and Self-healing. Within Self-healing, Root Cause Analysis (RCA), i.e. diagnosis of the cause of the network problems, is one of the most difficult tasks. To automate diagnosis, Data Mining (DM) algorithms over sets of solved troubleshooting cases can be applied in order to use Knowledge Based Systems. Data reduction is part of the DM process, where large time-dependent matrices are transformed into vectors of values. In this work, an algorithm for data reduction is proposed, which is based on detecting degradations. Once the degraded period is isolated, time-dependent data is aggregated and a vector is obtained.**

I. Introduction

In the last decade, the demand for mobile data connections has greatly increased. To cater for the growing bandwidths and quality requirements of mobile networks, new communication technologies have been developed. Currently, operators are using Long Term Evolution (LTE) as the latest Radio Access Network technology. But in the current commercial scenario, competition among providers for offering a better service is tough. Users not only seek high bandwidths, but also a reliable service. Therefore, a key aspect of Quality of Service in current mobile networks is the minimization of downtimes. Usually, network problems or maintenance work are the cause of downtimes.

In order to reduce downtimes, automation in the Operation and Maintenance (O&M) of LTE networks is required. For this purpose, Self Organizing Networks (SONs) [1] were proposed, making heavy use of Artificial Intelligence (AI) techniques. SON functions cover three main aspects of O&M: Self-configuration (easy addition of new infrastructure to the network), Self-optimization (network parameters are automatically adjusted to offer the best possible service) and Self-healing (automatic detection, diagnosis, compensation and recovery from problems).

Self-healing [2][3] reduces the downtime due to malfunctions in the network. It speeds up the process of manual troubleshooting, by doing an early detection, a quick root cause analysis, automated recovery actions and compensation procedures in order to provide service while the problem is being solved.

This paper is focused on the diagnosis phase, which, due to its importance, has been covered by several studies; the Commune [4] project uses Case Based Reasoning for the diagnosis of problems; Bayesian Networks (BNs) are used in [5] and a combination of BNs and Case Based Reasoning is proposed in the UniverSelf project [6].

All of these proposals use AI algorithms, specifically Knowledge Based Systems (KBS), that is, systems that use expert knowledge to solve problems. In order to obtain that knowledge from field experts, a Knowledge Acquisition (KA) process is required. Traditionally, KA is performed manually, either with interviews or asking the field experts to program the KBS. But this process may be intrusive in their workflow, so an automated approach is required, such as the learning algorithm proposed in [7].

In order to apply Data Mining (DM), this type of learning algorithm needs a database of solved troubleshooting cases. These cases must contain a representative vector of values of the parameters that are observed to diagnose the problem, such as Key Performance Indicators (KPIs), and a label indicating the problem. All previously proposed algorithms [4][5][6][7] for diagnosis in cellular networks either assume that these databases are available, which is never the case, or, alternatively, the inputs to those algorithms are obtained with simple methods, e.g. calculating the average of the KPIs in a time period, such as a day.

In real networks, solved cases could be fed into a database as they are diagnosed by experts. But the format that these cases take is invalid for DM algorithms. Indeed, in the O&M system each KPI is given as a time series covering the entire observed time interval whereas DM algorithms require one single value per KPI. In addition, a root cause may be present during the whole interval, but it might only manifest itself as a degradation in short bursts, which makes algorithms that simply calculate average values not valid in practice. Up to our knowledge, the only method proposed in literature to convert time-dependent real data from cellular networks into the single value per KPI required by most DM algorithms is the data reduction algorithm in [8]. Although there are detection methods that do have time into account [9][10], they do not focus on producing simplified vectors to be used in DM algorithms. The algorithm in [8] uses one representative KPI to isolate the periods of time where a degradation occurs. Each KPI time series will then be aggregated over that period of time. However, experts do not usually rely on a single KPI in order to detect degradations, since different root causes may degrade different KPIs. In this paper, an algorithm for data reduction that takes into account multiple KPIs is proposed and evaluated on real network data. It is based on detecting intervals where the network performance is degraded and calculating the average of the KPIs on those degraded intervals.

The rest of this paper is organized as follows. In Section

II, the nature of degradations is discussed, and in Section III the algorithm to detect them is presented. In Section IV, the algorithm is tested and compared to detection with a single KPI. Finally, in Section V, the conclusions of the study are discussed.

II. Degradations in LTE

In current networks, to perform the task of *detection*, that is, to determine that there is a problem in a cell and to locate which network element is affected, experts monitor a small set of KPIs. When the value of one or more KPIs is abnormal for a group of cells, a list of *worst offenders* is obtained, that is, those cells that are contributing the most for the bad value of the KPIs. Once the problematic cells are identified, the experts observe the time series of the KPIs of each cell to find further clues for the diagnosis.

A KPI is *degraded* when it takes a value that is too high or too low as compared with its value under normal circumstances. For instance, the *Accessibility* KPI indicates the proportion of requested connections that are processed correctly. When it falls below a certain threshold, it is considered degraded. On the other hand, the *Dropped Call Rate* (the proportion of connections that end abnormally) is degraded when it grows too much. A *Degraded Interval* (DI) is defined as the period between the first occurrence of an abnormal value for a KPI and the moment where its behavior is again normal [8]. Degradation thresholds are approached in a fuzzy manner, since often there is not a sharp value that separates normal from abnormal behavior, so usually, the values will be referred to as being *high* or *low*.

A DI is indicative that a root cause is present; but the reverse is not always true; a problem may be present although there is no degradation observable. This is due to the fact that degradations happen when several factors enter into play. For instance, an eNodeB that has a low capacity for the area it is serving may not show a degradation at non-rush hours, although the capacity problem is still present. Therefore, in order to convert the time series for each KPI into a single value with the aim of applying DM algorithms, it would not make sense to aggregate each KPI over the full time interval where a root cause is present (for instance by taking the average), since the degraded values may be masked by the normal values.

Therefore, there is a need for isolating the time intervals where the degradation is observable. This will result in more representative aggregated values. This can be manually done by troubleshooting experts, but it is an invasive task that is far from their workflow. Automating this process has the advantage of reducing the need of human intervention and it also represents a considerable speed-up of the process of labelling problems.

III. Degraded Interval detection

Let *good* and *bad* be two thresholds that define three behaviors for a KPI: *normal behavior*, *degraded behavior* and a middleground *affected behavior*. As example, Figure 1 shows these thresholds on a time series of a KPI. Let $S = \{S_1, S_2, ..., S_N\}$ be the set of driving KPIs, that is, those KPIs that are important to determine whether a cell is degraded.

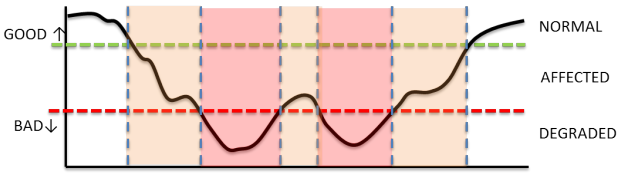

Figure 1. *Good* and *Bad* thresholds and defined states.

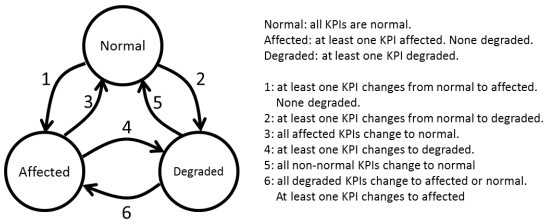

Figure 2. State machine of the detection algorithm.

To define if the overall behavior of a cell is degraded, a state machine (Figure 2) is proposed. Three states are defined: *Normal* (all the KPIs in S are in the *normal behavior* state), *Affected* (at least one KPI in S is in the *affected behavior* state, but there is none in the *degraded behavior* state) and *Degraded* (at least one KPI is in the *degraded behavior* state). The algorithm will consider a DI any sequence of states that starts with a *degraded* overall state and contains only *degraded* or *affected* states. The DI will then start with the first *degraded* state of the sequence and finish when the last *degraded* state ends.

By using several KPIs, the algorithm will have a higher accuracy than using just one KPI, because a DI may be determined by degradations on several KPIs that are not overlapping in time. Also, by using several KPIs, a broader range of problems may be detected, because a single KPI may be unaffected by certain root causes.

IV. Results

The proposed algorithm has been applied over a database of 360 problematic cases collected over a time span of six months from a real network of 5366 eNodeBs located in a major urban area. A case is defined as the data obtained from a problematic cell along with a label identifying the root cause. These cases have been manually diagnosed by experts, using the *Accessibility* and *Retainability* KPIs for detection, which indicate, respectively the proportion of successful connection attempts and the proportion of connections that end correctly. Experts use the same *good* (99%) and *bad* (98%) thresholds for both KPIs, basing this choice on experience. The choice of KPIs and thresholds can be done using specific algorithms [11] that are out

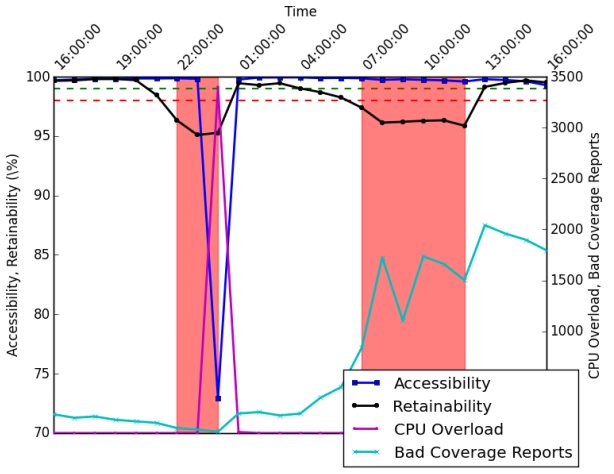

Figure 3. Example of two detected DIs.

of the scope of this article. Other KPIs that are often used for detection are *Throughput* (that measures the data transmission rate, and has several sets of thresholds), *Handover Success Rate (HOSR)* (that measures the proportion of successful handovers, and is cosidered *bad* below 95% and *good* above 98.5%) and *Inter-Radio Access Technology (IRAT) Rate* (that reflects the proportion of connections that start in LTE and are redirected to 2/3G and is cosidered *good* below 1% and *bad* above 2%). These KPIs are not used for detection in this data set, but they are nevertheless available and can be used to better explore the data.

Figure 3 shows a specific case where two different degradations occur. The first one is due to a CPU overload (due to a software glitch) that causes bad values of both *Retainability* and *Accessibility* (due to internal traffic control processes). When the software glitch causes a high number of *CPU Overload Alarms* (that fire when the CPU load reaches a certain threshold), it is detected and the eNodeB is reset. Once restarted, a parameter misadjustment causes coverage problems. This can both be observed in the *Number of Bad Coverage Reports* (i.e. the number of user terminals that report back to the eNodeB a received power or SNR that is lower than a certain threshold) and the *Retainability* KPIs (but not in *Accessibility*).

Table I compares the averages of the KPIs using the daily average based detection (which is often the first value that is observed when performing a top-down analysis), the average over DIs detected using both *Accessibility* and *Retainability* and the averages of the non degraded time intervals. It can be seen that although there is a degradation between 22:00 Day 1 and 00:00 Day 2, nothing abnormal shows up in the daily average of Day 1. A minor degradation is visible for Day 2. On the other hand, when using DIs, the degradations clearly stand out. On the lower level KPIs (*Number of CPU Overload Alarms* and *Number of Bad Coverage Reports*), degradations also are clearly visible. Moreover, when using the DIs, these KPIs

Table I
COMPARISON OF KPI AGGREGATION USING DAILY AND DI PERIODS

	Acc.	Ret.	CPU	Bad Cov.
Normal	99.88	99.86	0.61	140.63
Day 1	99.87	99.43	0.0	96.08
Day 2	98.68	98.51	141.92	1024.54
DI 1	90.88	95.57	1132.0	32.0
DI 2	99.74	96.37	0.0	1428.17

provide a clear picture of what is the root cause in each case. In DI 1, the *Number of CPU Overload Alarms* is very high compared to the normal average, whereas the *Number of Bad Coverage Reports* is not degraded; clearly implying problems with the CPU (that further analysis can trace to the software glitch). In DI 2, the situation is reversed; there are no CPU alarms, but the *Number of Bad Coverage Reports* is largely increased. When using the daily averages, Day 1 shows no degradations (even though the first DI starts at 22:00) for the lower level KPIs, and Day 2 shows degradations in both of them. Since both low level KPIs are degraded, there is no way to discriminate the two different problems. A visual inspection of the case will determine that Day 2 has two different problems, but that implies further manual actions, as opposed to DI detection, that automatically has isolated the different degradations, making the information available at first sight and also making possible the automation of the whole process.

In this test, two KPIs have been used, but if only one had been used, the results would have been different. If only *Accessibility* was used, the second DI would not be detected at all. If only *Retainability* had been used, on the other hand, the DIs would have been detected anyway; but other DIs would be missed where only *Accessibility* is degraded, such as the example shown in Figure 4, where a configuration problem produces a high number of rejections from the user terminals, causing them to reattempt the connection establishment. In this case, only *Accessibility* is degraded, whereas *Retainability* is not afected at all, because ongoing connections are not influenced by the configuration parameters that are the root cause.

With these examples in mind, we have applied the algorithm over all the 360 cases of the database. Table II shows these results. Two scenarios are compared: using daily average and using the proposed algorithms, and for each scenario, three sets of KPIs are used: only *Retainability*; *Retainability* and *Accessibility*; and *Retainability*,*Accessibility* and *HOSR*. When taking the daily average, only one threshold can be used, so the *bad* threshold is used in order to maximize the detections.

The advantage of using DI detection is not only an increased number of detections and accuracy on the values of the KPIs, but it also helps transforming the time-dependent matrices that represent all the KPIs in a certain

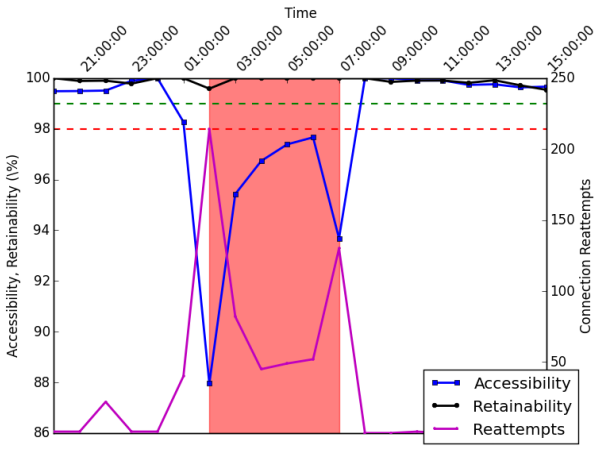

Figure 4. Example of a DI detected only with *Accessibility*.

Table II
COMPARISON OF NUMBER OF DETECTIONS

Daily average	Proposed method	Improvement
Only Ret.		
1160	1899	63.71%
Ret. & Acc.		
1350	2429	79.93%
Ret., Acc. & HOSR		
2716	4715	73.6%

period into a reduced set of vectors (by doing the average over the detected DI), each representing an occurrence of a problem. These vectors are much more easy to process by Data Mining algorithms. For each degradation, one single DI is produced by the algorithm, regardless of its duration. In the original input, each time interval (degraded or not) has an entry for each KPI. Therefore, the algorithm also has a data volume reduction effect. When applied over the whole database (with *Retainability* and *Accessibility* as driving KPIs), 2429 DIs are obtained. Considering that there are 347 KPIs in each entry of the database, that each entry collects hourly KPIs for 2 weeks and that the total number of cases is 360, the original number of individual elements is 41973120. Using the same approach with the 4738 detected DIs, the number of elements is reduced to 842863, representing a reduction of 97.99%. If the trend information is extracted too, as exposed in [8], each KPI will have a double entry (one representing its average, and one representing its slope during the period prior to the degradation), so the number of elements will be 3288172, representing a reduction of 95.98%.

V. CONCLUSIONS

There is a need for automating troubleshooting in mobile networks due to the incresingly high cost of downtimes. In order to automate root cause analysis, learning algorithms may be used over datasets of known problems in order to extract the knowledge required for AI systems. But data collected from operator databases is often in a format that cannot be directly used. First of all, the KPI data is given as time-dependent matrices, and secondly, the KPI time series normally include both operating intervals with degraded intervals.

In this work, an algorithm to isolate Degraded Intervals has been presented. It uses several predefined KPIs in order to determine when a dataset is affected by a root cause. This information may be used to perform data reduction (i.e. transformation from time-dependent matrices to time-independent vectors of KPI averages) to create a good training set for learning algorithms for diagnosis.

The proposed algorithm has been tested over a database of real cases, showing a potential capacity for data reduction. Also, more detailed inspection of a specific case shows the advantages of using DI detection over using daily averages and the advantage of using more than one KPI.

REFERENCES

[1] 3GPP, *Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements*. 3rd Generation Partnership Prject, TS 32.500 ed., September 2012.

[2] R. Barco, P. Lazaro, and P. Muñoz, "A unified framework for self-healing in wireless networks," *Communications Magazine, IEEE*, vol. 50, no. 12, pp. 134–142, 2012.

[3] 3GPP, *Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements (Rel 11)*. 3rd Generation Partnership Prject.

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

[5] R. Barco, L. Díez, V. Wille, and P. Lázaro., "Automatic diagnosis of mobile communication networks under imprecise parameters," *Expert systems with Applications.*, vol. Vol.36 (1), pp. 489–500, January 2009.

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

[7] E. J. Khatib, R. Barco, A. Gómez-Andrades, and I. Serrano, "Diagnosis based on genetic fuzzy algorithms for LTE Self-Healing," *IEEE Transactions on Vehicular Technology, In press*, 2015.

[8] E. Khatib, R. Barco, I. Serrano, and P. Muñoz, "LTE performance data reduction for knowledge acquisition," in *Globecom 2014*, pp. 270–274, Dec 2014.

[9] S. Novczki and B. Gajic, *Fixed-Resolution Growing Neural Gas for Clustering the Mobile Networks Data*, vol. 517 of *Communications in Computer and Information Science*. Springer International Publishing, 2015.

[10] B. Gajic, S. Novaczki, and S. Mwanje, "An improved anomaly detection in mobile networks by using incremental time-aware clustering," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 1286–1291, May 2015.

[11] T. Mehmood, K. H. Liland, L. Snipen, and S. Sæbø, "A review of variable selection methods in partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 118, pp. 62 – 69, 2012.