Les Cahiers du GERAD ISSN: 0711–2440

A machine learning framework for sleeping cell detection in a smart-city IoT telecommunications infrastructure

O. Manzanilla-Salazar, F. Malandra, H. Mellah, C. Wetté, B. Sansò

G–2019–70

September 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A machine learning framework for sleeping cell detection in a smart-city IoT telecommunications infrastructure

Orestes Manzanilla-Salazar [a, b]
Filippo Malandra [c]
Hakim Mellah [a, b]
Constant Wetté [d]
Brunilde Sansò [a, b]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Electrical Engineering, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

[c] *Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260, USA*

[d] *Ericsson Canada*

orestes.manzanilla@polymtl.ca
filippom@buffalo.edu
hakim.mellah@polymtl.ca
brunilde.sanso@polymtl.ca

September 2019
Les Cahiers du GERAD
G–2019–70

Abstract: The smooth operation of largely deployed Internet of Things (IoT) applications will depend, among other things, on effective infrastructure failure detection. Access failures in wireless networks Base Stations (BSs) produce a phenomenon called "Sleeping Cells", which can render a cell catatonic without triggering any alarms or provoking immediate effects on the cell's performance, making it difficult to discover. To detect this kind of failures, we propose a Machine Learning framework, based on the use of Key Performance Indicators (KPIs) statistics from the BS under study, as well as those of the neighboring BSs with propensity to have their performance affected by the failure. A simple way to classify neighbors is to use Voronoi diagrams. In this paper we propose a much more realistic approach based on the nature of the radio-propagation and the way the devices choose the BS to which they send access requests. We gather data from large-scale simulators that use real location data for BSs and IoT devices, and pose the detection problem as a supervised binary classification problem. We measure the effects caused on the detection performance, by the size of time aggregations of the data, the level of traffic and the parameters of the neighborhood definition. Extra Trees and Naive Bayes classifiers achieved Receiver Operating Characteristic (ROC) area under the curve scores of 0.996 and 0.993 respectively with False Positive Rates (FPRs) under 5%. The proposed framework holds potential for other pattern recognition tasks in smart-cities wireless infrastructures, that would enable the monitoring, prediction and improvement of the Quality of Service (QoS) experienced by IoT applications.

Keywords: wireless networks, smart cities, machine learning, IoT, failure detection, sleeping cells, M2M communications

1 Introduction

The deployment of the IoT in urban areas is enabling the creation of so-called "smart cities" where city life will be improved by using large amounts of information coming from hundreds of thousands of geographically distributed communicating devices. Such information will lead to the automation of some systems and the creation of new applications that will enhance city living. Smart parking, smart pedestrian crossing, intelligent transportation systems, and intelligent power distribution are just a few of the new types of innovations that can be put in place with the effective exchange of information between the city IoT devices. IoT-enabled data and services in smart cities rely on either (a) users interacting with the smart devices that should be connected to the Internet, or (b) users using Internet services that depend on IoT devices that have the roles of sensors or actuators [1]. In both cases, communications are essential for the IoT applications to work.

Even though there has been several telecommunication technologies proposed for the deployment of different IoT applications in cities [2] [3], the ubiquity of cellular communications is making operators and standardization entities such as the 3GPP to push for a common cellular infrastructure for smart cities based on 4G enhancements and on 5G.

Even with the use of a common communication infrastructure, there are several drawbacks for the smart-city large-scale implementation. First, it heavily depends on telecommunications reliability, as even banal failures may lead to the massive malfunctioning of key automated systems. Second, the type of telecommunication traffic produced in smart cities will mostly be produced by IoT machines inside those automated systems. The problem is that the statistical behaviour of such traffic is quite different than that produced by humans [4] and the lack of direct human interaction will make it even more difficult to detect telecommunication failures. Finally, the distributed nature of the applications and the large number of devices and connections will also hinder failure detection.

One of the most difficult type of failures to detect in cellular networks is the so-called "Sleeping Cell failure". It consists in failures that will not provide alarms even if the cell is malfunctioning. In human cellular communications a sleeping cell will make the users react to the lack of service, change location and eventually notify the operator. This takes time, in some cases even days, before the operator discovers the failure and takes the proper measures [5]. The importance of Sleeping Cell failures is greatly amplified in smart cities, where many automated systems may depend on the normal function of a particular cell. Thus, the city does not have the leisure to wait days for the cell malfunctioning to be detected. The delay constraints of essential smart-city applications might be difficult to satisfy even with fully-operational BSs, due to the massive number of devices that are expected to request access [6].

The objective of this paper is to present a Machine learning framework to detect sleeping cells in a smart-city IoT context. The framework is based on:

- the introduction of a new type of neighboring and proximity definition for a particular cell
- the use of aggregated KPIs over time intervals for different types of IoT applications.

The data used to feed our framework was extracted from a large-scale IoT infrastructure simulator that takes as input a real city database and the real locations and features of the BSs of several service providers.

2 State of the art

Failure detection of network elements is one of the main concerns of mobile network operators. Several papers in literature treat the problem using real network operator data at the BS level, [7] [8] [9] [10] [11]. Such an approach produces very accurate results for the specific networks but the solutions are not easily generalized due to the difficulty in retrieving real cellular networks data. As a consequence, most authors deal with simulated data, such as in [12], [13], [14], [15], [16], [7], [10], [17], [18], [19],

though emulations based on real data can also be used [20]. In this work, we employ simulated network data generated with a large-scale network simulator [21] (an extension on the proposed in [4]), which employs real data on the position of network elements and parameters of the communicating nodes.

Concerning the use of Machine Learning for cellular network analysis and failure detection, one approach is to identify standard traffic patterns and quickly detect deviations from normal behaviors (i.e., unsupervised learning). In particular, existing research focused on: i) anomaly detection [7,22,23], ii) KPIs [24], iii) clustering [11, 14, 16], and iv) dimensionality reduction techniques [14, 25]. Other authors exploit known properties of cellular networks to perform supervised learning and use it to detect faulty elements in the network [11, 13, 18, 19, 23, 26].

A complementary approach to ML proposed by [27] is to acquire data from troubleshooting (human) experts in mobile networks and to use their experience and knowledge to improve fault detection. In addition to the proposed techniques, fuzzy models can also be used for failure detection, as shown in [17, 20].

Finally, some authors propose to detect failures in a network element by looking at anomalies in the traffic and KPIs from neighboring cells [10, 13, 19]. This is particularly powerful when the traffic generated by a defected cell does not present remarkable anomalies in its KPIs, such as in the case of Random-Access Channel (RACH)-sleeping cells where new users cannot connect but existing users in the cell can continue to transmit regularly during the fault.

In this paper, we propose to use well-known supervised learning techniques for BS failure detection in a smart-city cellular infrastructure. In particular, for each cell, KPIs from neighboring cells are analyzed to highlight anomalies and detect defected BSs. Differently from the reviewed literature, i) we consider advanced propagation models not only based on the distance but also on other parameters, such as bandwidth, frequency, and antenna orientation, and ii) we define different neighbor categories to improve failure detection.

3 System modelling

3.1 Communication infrastructure

The cellular network model is composed of a set $\mathcal{A}$ of base stations enumerated as $\{1, \ldots, M\}$, a set of users $\mathcal{G}$, a backbone $\mathcal{C}$, and a Data Management Center (DMC). Only the access performance is considered, the core and metropolitan part of the network is modeled as a black box.

We assume a limited number of wireless channels (i.e., the Resource Blocks (RBs) in LTE) that can be used to transmit data between users and base station. This is done through dedicated control channels allocated through a random access procedure, based on the transmissions of preambles. The available preambles are limited and might collide, triggering retransmission and introducing additional delay in the communications between users and base stations.

Key parameters in this study are the collision probability and the access delay, i.e., the time required for a user packet to be received by the associated base station. In particular, high order statistics on those two parameters are used to detect sleeping cells. Further details on the methodology are provided in Section 4.

3.2 Topology definition

The framework was built with real telecommunications and urban data from the city of Montreal (see details in [4]). In Figure 1, a toy example of smart city cellular system is displayed: network users are represented by IoT devices, such as cars, buses, traffic lights, and security cameras. Details on the type of IoT devices and on their characteristics can be found in a previous work [28], where six different IoT applications were presented. The rectangle represents the geographical boundaries of a smart city, in which three base stations i, j, and k are installed and provide network access to the IoT devices.

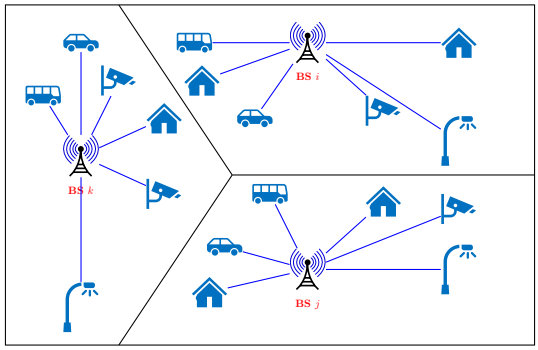

Figure 1: A sample scheme of the proposed architecture with three BSs and a large number of IoT devices.

The geographical position and other features of the BSs, such as bandwidth, transmitted power, and orientation, were retrieved from [29].

To characterize the links between users and base stations we i) define a threshold on the received power, ii) for each IoT device, compute the power received by all the BSs, iii) determine the list of BSs that cover each IoT device. A threshold of -100 dBm was considered in this study. The received power is computed according to the Cost-Hata propagation model, which allows computing the path loss based on key parameters, such as frequency, distance, and height. This propagation model is also combined with the corresponding radiation patterns for each BS. This leads to computing the Equivalent Isotropic Radiated Power (EIRP), based on the elevation, gain and inclination of the antennas, which are also available at [29]. The list of BSs covering a certain IoT device can be very large, especially in a densely populated urban scenario like Montreal, and this can lead to computational inefficiencies and large execution times. As a consequence, this list is limited to the ξ BSs with the highest received power. The list is used, as described in Section 4.1, to combine the analysis of one cell KPIs to those of neighbor cells, and ultimately to detect the sleeping cells with high accuracy.

3.3 The sleeping cell problem

A sleeping cell is usually defined as a cell which is not entirely operational and whose malfunctioning is not easily detectable by the network operator, as highlighted in [24]. This term is generally used to describe a wide variety of hardware and software failures, which degrade the QoS and Quality of Experience (QoE) and can remain hidden to the network operator for a long time (days or even weeks) [30]. In this study, we address a particular type of sleeping cells that affects the RACH in LTE networks [31]. On the one hand, this type of problem affects new users who are not able to complete the access procedure, and consequently cannot access the network. On the other hand, existing users, which were already connected to the base station when the problem manifested, continue to transmit. As a consequence, standard methods based on traffic monitoring fail in detecting the problem, because the network operator continues to monitor updated statistics coming from the *RACH-sleeping* base station.

4 A framework for sleeping cell detection

4.1 Neighborhood/closeness definition

When trying to detect if a particular BS has failed, our key idea is to include data from its "neighborhood". But *how does one determine which BSs can be considered as "neighbors"?* Though BS distance can be used, as done in [19], we now propose a richer definition: a neighboring BS is actually one whose performance KPIs are *likely* to be affected by the access failure in the BS under study. Accordingly, we base our definition on the following:

1.-Antenna priority and Received Signal Strength (RSS): when sending access requests, IoT devices will choose the active BS with the strongest RSS in its location. Figure 2 shows examples of BSs and their position in priority lists of size $\xi = 3$ for a series of locations. Note that in our implementation the 12 antennas with highest RSS are considered ($\xi = 12$). In some locations, the priority list can be shorter, as a consequence of the threshold mentioned in 3.2.

2.-Directional antennas: the strongest received signal might not come from the closest BS. Therefore, simply using distance to estimate the BS a device will choose to establish a connection, is not a valid option.

3.-KPIs availability: it is feasible to obtain aggregations of performance KPIs for all packets processed by a BS during any period of time.

The way a Sleeping Cell affects the performance of a "neighbor" can be described as follows:

- *Step 1*: The observed BS fails.
- *Step 2*: Each device usually served by the failed BS chooses the one with the second highest RSS as an alternative.
- *Step 3*: The additional traffic produced by the "new" devices requesting access induces a performance degradation in the *chosen* BS.

Note that it is desirable that the alternative BS chosen by the device in step 2 would be a "neighbor" of the one experiencing the failure. This is the reason we base our definition of *neighborhood* on the notion of *probability of experiencing a performance degradation*. Also note that even though simultaneous failures of nearby BSs may not be frequent, they cannot be ruled out. Therefore devices around the failed location may explore down their priority list until they find a non-failed BS alternative.

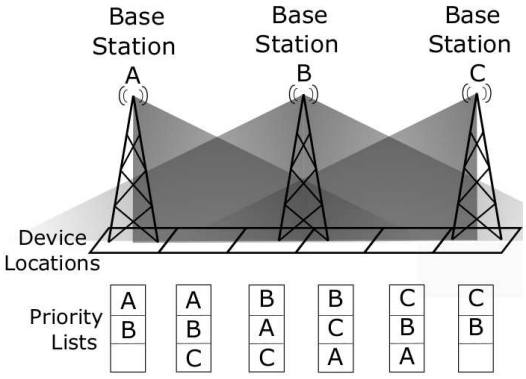

Figure 2: Priority list example.

Using probabilities to define neighborhood categories

the concept is illustrated by the following example. Let p be the probability of access failures of *any* BS during any given time interval of duration T. Let us also assume that failures in different BSs and time intervals are independent events. Let us have for the device location $g \in \mathcal{G}$, the priority list $\mathbf{s_g} = [s_g(1), s_g(2), s_g(3), s_g(4)]$ containing the 4 BSs with the highest received power in g. Given that $s_g(1)$ fails, one of the following happens:

- If $s_g(2)$ is operational it will receive all the traffic from devices in g with probability 1.
- If $s_g(2)$ is also asleep the packet of device in g will be handled by $s_g(3)$. This will happen with probability p, given that $s_g(1)$ is sleeping.
- If both $s_g(2)$ and $s_g(3)$ are asleep, then $s_g(4)$ will be the alternative BS. The probability is p^2.

This example shows the intuition behind our definition of *neighborhood of category n of BS k*. We define it as the set of BSs who have a minimum probability of p^{n-1} of receiving traffic normally served by a BS k, when BS k fails. Applying the definitions to the previous example, assuming that there is only one device in the system, in location $g \in \mathcal{G}$ and that the observed BS is $s_g(1)$, we get the following possible neighborhood sets for $s_g(1)$:

- Neighborhood category 1: $s_g(2)$.
- Neighborhood category 2: $s_g(2)$ and $s_g(3)$.
- Neighborhood category 3: $s_g(2)$, $s_g(3)$ and $s_g(4)$

Figure 3 shows a set of BSs numbered from 1 to 17. Note that BS 7 is the target of this analysis, and the BSs in the neighborhood category 1 (dark grey colored) are also part of the set of category 2 (light grey colored). In order to highlight the differences between the proposed neighboring structure and the classical geographical one, in this example, an immediate neighbor, such as BS 6, is excluded from the neighborhood of category 1, and the more distant BS 15 belongs to it. This can be caused by the fact that BS 6 has either a low power or an antenna orientation, causing its signal to be weakly received in the locations typically served by BS 7.

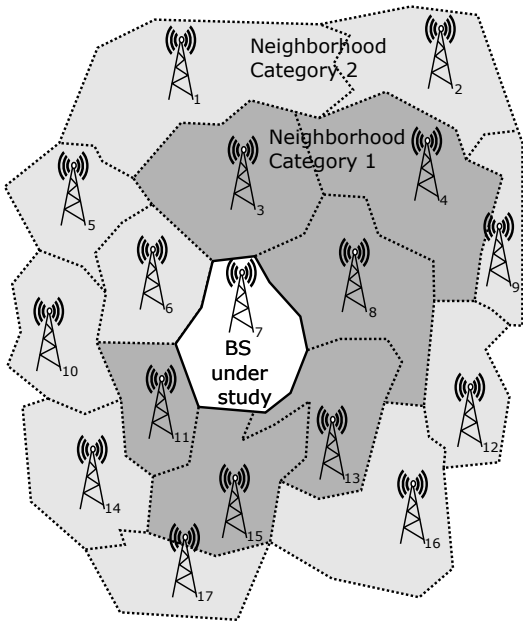

Figure 3: Example of neighborhood categories 1 and 2.

The u-v proximity

In order to compute, for each neighborhood category, the set of neighboring BSs for each target BS, we need to compute first what we have called $u - v$ proximity. To define it, we need first to obtain the priority lists of BSs for each of the locations in the geography where a device has been installed (or can reasonably assumed to be). Priority lists should contain the BSs in decreasing order of RSS. BSs can exist in the list more than once, if they have more than one antenna. Two BSs *have* $u - v$ proximity *if* there exists at least one device location in whose priority list the positions occupied by the two BSs are between position u^{th} and v^{th} (including the extremes).

Based on this definition, multiple $u - v$ proximities can be defined for a single pair of BS if there is more than one location whose list contains antennas from both BS. This occurs because for a pair of BS, when considering different device locations containing both BSs in their priority lists, the positions they occupy migh be very different. In location well positioned to receive signals from both BSs, both

might occupy the first two positions of the list. In a location far from both BSs, they might occupy the two last positions of the list.

The existance of multiple $u-v$ proximities for the same pair of BSs is not necessarily a problem. Their value becomes evident, when we consider that for a single pair of BSs, it is feasible for them to lack $1-2$ proximity, for example, but to have $2-3$ proximity. This allows us to say that this pair of BSs do not belong to each other's neighborhood category 1, but they belong to each other's neighborhood category 2. No device normally connected to one of them will have as a first choice the other BS in case of an access failure, unless there are two simultaneous failures.

Formally, given $\mathcal{A}$ the set of all BSs and $\mathcal{G}$ the set of all geographical locations in the considered smart city scenario, the *BS priority* list $\mathbf{s_g}$ can be defined as $\mathbf{s_g} = \{s_g(k)\}$, where $k = 1, \ldots, \xi$, $g \in G$, and $s_g(k) \in \mathcal{A}$. Please note that BSs in $\mathbf{s_g}$ are ordered by decreasing received power.

Let us also define the *u-v range* of priorities as $\mathcal{I}_{u-v} = \{u, u+1, ..., v\}$, where $\xi \geq v > u \geq 1$.

Let us also define that two BSs $i, j \in \mathcal{A}$ have *u-v proximity* if the following two conditions apply: i) $\exists g \in G:\ i = s_g(q)$, $j = s_g(z)$, and ii) $q, z \in I_{u-v}$.

In Figure 4, in order to visually show the $u-v$ proximity concept, the following are displayed: location $g \in \mathcal{G}$; the a set of possible machines installed in g; some of the ξ BSs in the priority list s_g. Note that, in this example, u, i, j, and v all belong to the range $\mathcal{I}_{u-v}$. Therefore, BSs $A = s_g(i)$ and $B = s_g(j)$ have $u-v$ proximity.

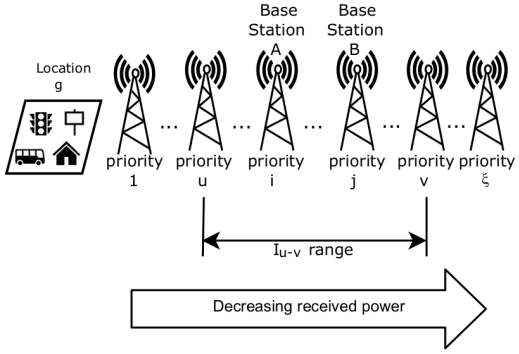

Figure 4: Range u-v of proximity between BSs (i,j).

Then, we compute a *neighborhood matrix* for each *neighborhood of category n*, such that there is one row and one column for each BS of the system. The element (i, j) of the matrix is the value of the $u-v$ proximity associated to the neighborhood category. We define neighborhoods of categories from 1 to 11, such that for category n, the element (i, j) is computed as the $1-(n+1)$ proximity. As an example, for neighborhood category 3, each element (i, j) is computed as $1-4$ proximity for the pair of BSs (i, j). We obtain one neighborhood matrix for each of the neighborhood categories. Because the u, for the $u-v$ ranges, is always 1, the matrices are built in a way such that if node i is a neighbor of node j according to the neighborhood matrix of category n, they will be also neighbors according to the matrices of any category l if $l > n$. Incrementing the category produces a neighborhood that is additive to the neighborhood of the previous category.

The neighborhood matrices are used to aggregate the KPIs of the neighborhood for each of the BSs, as part of the construction of the "feature vectors" that allow the use of Machine Learning for the detection of the failures. The specifics on how this is done are detailed in Section 4.3.

4.2 Network simulation

We use an LTE simulator[1] similar to the one described in [4]. The way IoT devices gain access to the BSs is based on computing priority lists ($\xi = 12$) for each device connecting to the mobile infrastructure.

As this model encompasses the uplink RACH procedure and the transmission until the reception at EPC, its output is composed of counters and statistics related to both phases:

- Count of packets created.
- Count of packets transmitted.
- Count of RACH collisions.
- Count of RACH attempts.
- Minima, maxima and average of the RACH delay.
- Minima, maxima and average of the transmission time (from the RACH completion to the reception at EPC).

The priority lists are computed considering as options the antennae, instead of the BSs. In the preprocessing to compute the neighborhood matrices the antennae identification code in these lists are replaced by the identification of the BS where the antennas are installed.

The RACH failures are modeled as affecting simultaneously all the antennae of a particular BS. As in general the BSs do not have the same number of antennae, most of the times the probabilities to the positions in the priority lists may have values higher than the theoretical minimum described in 4.1 for a particular neighborhood category. We purposely chose omitting the implementation of countermeasures in the preprocessing to address the "noise" introduced by it, as the effect on the results does not hinder the methodology.

4.2.1 Simulated scenarios

In Table 1 we show the locations for devices used, the levels of traffic, duration of the simulation, total BSs and number of failing BSs when generating the data.

Table 1: Simulation scenarios.

Locations of devices	smart meters, surveillance cameras, bus stops, traffic lights, parking lots, microPMUs*
Levels of traffic	High / low
Simulation duration (h)	12
Total BSs	479
Failing BSs	50

* Micro-phasor measurement units, devices required in locations where alternative sources of power are used, such as solar panels, wind power stations, etc., to monitor the distribution power grid.

We considered two scenarios in our simulations: high traffic and low traffic. The parameters used in each of the IoT applications in both scenarios are shown in Table 2.

Table 2: Applications table.

Applicationname	Machines location	Packet size (bytes)	Traffic generation distribution	Avg. inter-arrival time (s) (high traffic / low traffic)
Smart meters	Postal codes	200	Poisson	600 / 300
Smart parking	Parking slots	250	Poisson	1,800 / 900
Smart public transportation	Bus stops	300	According to schedule	(N/A)
Public safety	Surveillance cameras	2,000,000	Poisson	60 / 30
Smart car congestion	Traffic lights	500	Poisson	30 / 15
microPMUs	1,000 random locations	200	periodic	0.020
Smart fire alarms	Fire alarms	200	Predetermined	(N/A)

[1]Smart cities M2M: https://www.trafficm2modelling.com/

4.2.2 RACH failure generation

In a random sample of 50 BSs, total RACH failures were parametrized to initiate at the begining of each 1-hour simulation, with a duration of 30 minutes. This process was repeated 12 times (with different random seeds), including 10 minutes of initialization whose data was omitted from the analysis.

4.3 A ML framework

4.3.1 Preprocessing

The output of the simulator is aggregated in three ways: time intervals (according to the size of the time aggregations), across the antennae of each BS, and across the IoT devices. As a result, the data set contains the statistics at BS level and considers generic traffic (without distinction among the traffic generated by the different devices/applications). The results were preprocessed aggregating the data at BS level in time intervals of 5, 10, 15 and 30 minutes. This allowed us to study the effect the aggregation size on detection performance.

A fundamental part of the preprocessing, is the computation of the neighborhood matrices for each of the neighborhood categories. This process involved the following steps:

- Analyzing the entry file used by the simulator to obtain the priority lists for the location of each of the IoT devices connecting to the mobile infrastructure.
- Processing the priority lists to obtain the $u - v$ proximities between each pair of BSs.
- Using the $u - v$ proximities according to each of the neighborhood categories, to obtain the neighborhood matrices.

Our aggregating procedure consisted on computing the following statistics: average, variance, skewness, kurtosis, percentiles (5, 25, 50, 75, 95), minimum, maximum and range.

After the data was aggregated for each aggregation interval/BS, a normalization step was used to force the values to lie within the range $(0, 1)$.

The feature vectors to be used in the Machine Learning algorithms were built concatenating for each aggregation interval/BS two vectors:

- The vector of aggregation statistics of the "target" BS.
- An aggregation of the statistics of all the BSs that can be considered *neighbors*, according to the neighborhood matrix of the category that is being studied.

In order to perform supervised classification, the data set was completed by associating each vector with a category label or "target value" that indicates whether the "target" BS was experiencing a RACH failure or not in the particular aggregation interval. This procedure was repeated for the same data, for each time aggregation size and neighborhood category considered.

4.3.2 Models and training strategy

The binary classifiers used in our experiments are:

1. Naive Bayes
2. Logistic Regression
3. Linear, Quadratic, Cubic and RBF Support Vector Machines
4. Decision Trees
5. Extra trees
6. Bagged Decision Trees
7. Random Forest
8. Shallow (single hidden layer) Neural Networks

For each of the simulation scenarios and preprocessing strategies, the data set was randomly split into *training* (70%) and *testing* (30%) sets. Parameter tuning for each of the classification models was performed via 10-fold cross-validation (within the data from the *training* set). The detection (classification) performance reported in this paper are those obtained from the predictions obtained when evaluating data from the *testing* set in the trained classifiers (with the hyperparameters chosen via cross-validation).

The detection (classification) performance is mainly evaluated via the ROC Area Under the Curve (AUC) score. However, failure investigation activities associated to a false alarm represent a considerable operational cost for the telco providers. Consequently, we also computed FPRs as a performance index.

5 Numerical results

According to their classification performance along all of our experiments, we can identify two groups of supervised classifiers:

Table 3: Minimum and average ROC AUC per classifier.

	ROC AUC	
Classification model	**Average**	**Minimum**
Extra Trees	**0.997**	0.971
Random Forests	0.985	0.921
Decision Trees	0.984	0.897
Naive Bayes	0.983	0.938
Bagged Decision Trees	0.981	0.879
Linear Support Vector Machine (SVM)	0.959	0.912
Shallow Neural Network	0.955	0.898
Quadratic SVM	0.954	0.891
Logistic Regression	0.954	0.877
Radial Basis Function (RBF) SVM	0.953	0.902
Cubic SVM	0.952	0.875

- *Group 1*: consisting on all the SVM classifiers, along with Logistic Regressions and the Shallow Neural Networks, which achieve in average an AUC score below 0.97.
- *Group 2*: consisting on the ensemble learners (Bagged Decision Trees, Random Forests and Extra Trees), Decision Trees and Naive Bayes, whose average AUC is higher than 0.97. The Extra Trees classifiers in particular, in its worst performance, achieved an AUC higher than 0.97, and its average score was higher than 0.97.

These values correspond to the task of determining whether the aggregated KPIs of one BS and its neighborhood were captured during an interval where a RACH failure was taking place or not, without any information regarding the particularities of the BS (number of devices, applications, or the identification of the BS). We can also observe in Figure 5 that FPRs for classifiers of Group 2 were also better than those of Group 1. Naive Bayes classifiers have a similar average performance to that of Extra Trees (outperforming all the SVMs and the Shallow Neural Networks) while being extremely simple models.

Among all the experiments, the average *effect of increasing the traffic intensity* is mild (never higher than 1%), though in most classifiers the effect is slightly negative. Bagged Decision Trees and Extra Trees are an exception, showing an average reaction of performance improvement.

When averaging results with the *different aggregation levels* for the KPIs, we find that increasing aggregation size from 5 to 10 minutes has effects that range from mild (case of the classifiers of Group 2, with AUC values higher than 0.97), to clearly positive (case of the classifiers of Group 1), as can be seen in Figure 6. With the exception of Bagged Trees classifiers, all models in Group 2 experience a slight improvement when the aggregation level is increased to 15 minutes. Interestingly, aggregations of 30 minutes resulted in a decrease on the performance of classifiers of Group 2, while most models

in Group 1 experienced a slight improvement. Extra Trees in particular, obtained the highest AUC score averages along all aggregation sizes.

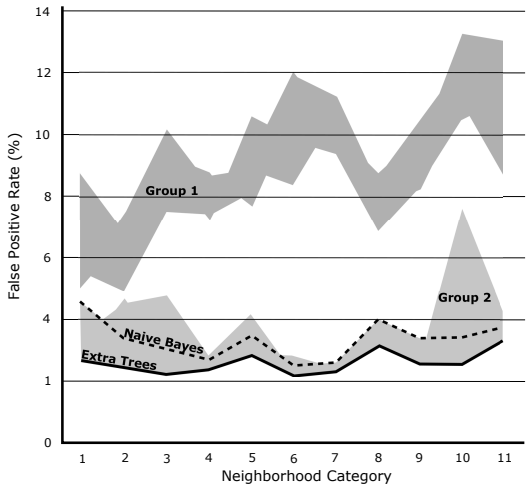

Figure 5: Effect of the size of the proximity range on false positive rate per classifier.

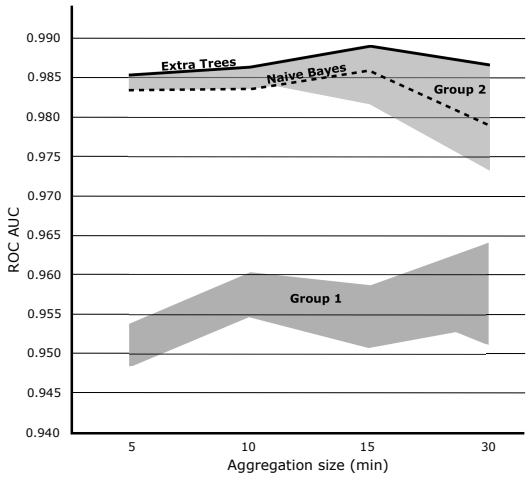

Figure 6: Effect of aggregation on AUC per classifier.

When observing the average *response to changes in the neighborhood category* (which affects which BSs' KPIs are included to detect the failure), we keep observing a clear distinction in the behavior of classifiers of Groups 1 and 2. Group 2, although its performance does not exhibit a clear response pattern, achieves better results (AUC) than Group 1 (see Figure 7). On the other hand, the performance of classifiers from Group 1 show progressively worst performance as the neighborhood category increases. There is no evidence, under our experimental conditions, that justifies the use of neighborhoods of categories higher than 2. The same is true when looking at the effect on FPRs (see Figure 5).

Figures 8 and 9 show the average AUC scores for the two best models: Extra Trees and Naive Bayes, respectively. In these figures, in order to analyze the *joint effect of time aggregation size and neighborhood category*, we created a heatmap, in which lighter colors represent higher AUC scores and consequently better detection performance. In the aforementioned figures, it can be noticed that both methods produced AUC scores close to 1. However, neighborhood categories 2, 3 and 4 obtained the best scores, specially when the size of the time aggregations were of 10, 15 and 30. In particular, best results were achieved with 15 minutes of aggregation and neighborhood category 2, allowing Extra Trees classifiers to achieve an AUC score of 0.996, and the Naive Bayes classifiers a score of 0.993.

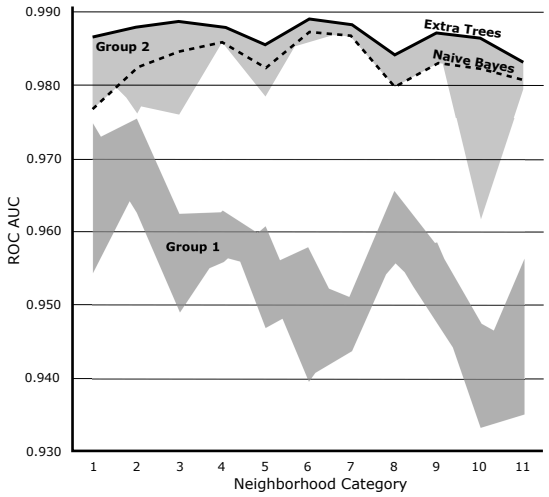

Figure 7: Effect of the size of the proximity range on AUC per classifier.

Neighborhood Category	Time Aggregation: 5 min	10 min	15 min	30 min
1	0.989	0.984	0.986	0.987
2	0.984	0.988	**0.996**	0.984
3	0.986	0.981	0.991	0.997
4	0.983	0.985	0.992	0.992
5	0.984	0.987	0.987	0.984
6	0.987	0.994	0.988	0.987
7	0.990	0.983	0.989	0.991
8	0.982	0.986	0.988	0.981
9	0.982	0.988	0.989	0.988
10	0.988	0.984	0.988	0.986
11	0.984	0.989	0.984	0.976

Figure 8: Joint effect of proximity and aggregation levels on Extra Trees AUC score.

Neighborhood Category	Time Aggregation: 5 min	10 min	15 min	30 min
1	0.985	0.982	0.986	0.953
2	0.982	0.987	**0.993**	0.967
3	0.985	0.981	0.981	0.992
4	0.982	0.981	0.990	0.992
5	0.982	0.984	0.981	0.984
6	0.986	0.992	0.984	0.987
7	0.987	0.981	0.988	0.991
8	0.981	0.984	0.987	0.967
9	0.981	0.984	0.988	0.979
10	0.983	0.980	0.985	0.982
11	0.983	0.984	0.981	0.976

Figure 9: Joint effect of proximity and aggregation levels on Naive Bayes AUC score.

6 Conclusion and future work

In this paper, we have proposed a supervised learning framework to detect RACH-related *sleeping* cells in a smart city cellular infrastructure. We used well-known binary classification techniques to detect network elements at fault, based on the analysis of aggregated KPIs, such as the RACH collision probability and the delay.

RACH-related sleeping cells are difficult to detect, due to the lack of evidence in the KPIs from a faulty cell. In order to overcome this problem, we have proposed to jointly consider the KPIs of one cell with those from the neighboring cells. We have also proposed a novel definition for neighbors of a cell, not choosing the nodes geographically closer to a cell but those that would be more likely impacted by its failure.

We used data obtained with a large-scale IoT network simulator, that employs real data on the telecommunication infrastructure and on the position of IoT nodes in a smart city environment. Although LTE was chosen to obtain numerical results, the proposed framework can be easily adapted to other cellular technologies, such as 5G.

Different levels of time aggregation intervals for KPIs were tested: 15 minutes resulted the aggregation interval that permitted achieving the highest AUC. Such an aggregation level permits to heavily reduce the amount of data to be analyzed by a network operator to detect faulty elements, resulting in large potential savings. Numerical results also proved Extra Trees and Naive Bayes to be the most effective binary classification techniques, among the ones considered in this work.

References

[1] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, An information framework for creating a smart city through internet of things, IEEE Internet of Things journal, 1(2) 112–121, 2014.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, Internet of things for smart cities, IEEE Internet of Things journal, 1(2) 22–32, 2014.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications, IEEE Internet of Things Journal, 4(5) 1125–1142, 2017.

[4] F. Malandra, L. Chiquette, L.-P. Lafontaine-Bédard, and B. Sansò, Traffic characterization and LTE performance analysis for M2M communications in smart cities, Pervasive and Mobile Computing, 48 59–68, 2018.

[5] R. Barco, P. Lazaro, and P. Munoz, A unified framework for self-healing in wireless networks, IEEE Communications Magazine, 50(12), 2012.

[6] M. Polese, M. Centenaro, A. Zanella, and M. Zorzi, M2m massive access in lte: Rach performance evaluation in a smart city scenario, in 2016 IEEE International Conference on Communications (ICC). IEEE, 2016, pp. 1–6.

[7] A. Coluccia, A. D'Alconzo, and F. Ricciato, Distribution-based anomaly detection via generalized likelihood ratio test: A general maximum entropy approach, Computer Networks, 57(17) 3446–3462, 2013.

[8] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, A first look at cellular machine-to-machine traffic: large scale measurement and characterization, in ACM SIGMETRICS Performance Evaluation Review, vol. 40. ACM, 2012, Conference Proceedings, pp. 65–76.

[9] ——, Large-scale measurement and characterization of cellular machine-to-machine traffic, Networking, IEEE/ACM Transactions on, 21(6) 1960–1973, 2013.

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

[11] S. Rezaei, H. Radmanesh, P. Alavizadeh, H. Nikoofar, and F. Lahouti, Automatic fault detection and diagnosis in cellular networks using operations support systems data, in Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP. IEEE, 2016, pp. 468–473.

[12] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lazaro, Automated diagnosis for umts networks using bayesian network approach, IEEE Transactions on vehicular technology, 57(4) 2451–2461, 2008.

[13] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, A cell outage detection algorithm using neighbor cell list reports, in International Workshop on Self-Organizing Systems. Springer, 2008, pp. 218–229.

[14] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch, Detection of sleeping cells in lte networks using diffusion maps, in Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. IEEE, 2011, pp. 1–5.

[15] M. Panda and P. M. Khilar, Distributed soft fault detection algorithm in wireless sensor networks using statistical test, in 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing. IEEE, 2012, pp. 195–198.

[16] Y. Ma, M. Peng, W. Xue, and X. Ji, A dynamic affinity propagation clustering algorithm for cell outage detection in self-healing networks, in Wireless Communications and Networking Conference (WCNC), 2013 IEEE. IEEE, 2013, pp. 2266–2270.

[17] A. Gómez-Andrades, P. Muñoz, E. J. Khatib, I. de-la Bandera, I. Serrano, and R. Barco, Methodology for the design and evaluation of self-healing lte networks, IEEE Transactions on Vehicular Technology, 65(8) 6468–6486, 2016.

[18] M. Sun, H. Qian, K. Zhu, D. Guan, and R. Wang, Ensemble learning and smote based fault diagnosis system in self-organizing cellular networks, in GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017, pp. 1–6.

[19] O. Manzanilla-Salazar, F. Malandra, and B. Sansò, enodeb failure detection from aggregated performance kpis in smart-city LTE infrastructures, in 15th International Conference on the Design of Reliable Communication Networks, DRCN 2019, Coimbra, Portugal, March 19–21, 2019, 2019, pp. 51–58.

[20] E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, and I. Serrano, Data mining for fuzzy diagnosis systems in lte networks, Expert Systems with Applications, 42(21) 7549–7559, 2015.

[21] Smart cities M2M traffic characterization and performance analysis, https://www.trafficm2modelling.com/home, accessed: 2019-02-01.

[22] B. Cheung, S. Fishkin, G. Kumar, and S. Rao, Method of monitoring wireless network performance, Mar. 23 2006, uS Patent App. 10/946,255.

[23] Q. Liao, M. Wiczanowski, and S. Stańczak, Toward cell outage detection with composite hypothesis testing, in Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012, pp. 4883–4887.

[24] F. Chernogorov, S. Chernov, K. Brigatti, and T. Ristaniemi, Sequence-based detection of sleeping cell failures in mobile networks, Wireless Networks, 22(6) 2029–2048, 2016.

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

[26] X. Liu, G. Chuai, W. Gao, and K. Zhang, Ga-adaboostsvm classifier empowered wireless network diagnosis, EURASIP Journal on Wireless Communications and Networking, 2018(1) 77, 2018.

[27] E. J. Khatib, R. Barco, P. Muñoz, and I. Serrano, Knowledge acquisition for fault management in lte networks, Wireless Personal Communications, 95(3) 2895–2914, 2017.

[28] F. Malandra, P. Potvin, S. Rochefort, and B. Sansò, A case study for M2M traffic characterization in a smart city environment, in International Conference on Internet of Things and Machine Learning (IML 2017), Liverpool, UK, Oct. 2017, pp. 1–9.

[29] Spectrum management system data, https://sms-sgs.ic.gc.ca/eic/site/sms-sgs-prod.nsf/eng/h_00010.html, accessed: 2019-09-09.

[30] S. Hämäläinen, H. Sanneck, and C. Sartori, LTE self-organising networks (SON): network management automation for operational efficiency. John Wiley & Sons, 2012.

[31] M. Amirijoo, R. Litjens, K. Spaey, M. Döttling, T. Jansen, N. Scully, and U. Türke, Use cases, requirements and assessment criteria for future self-organising radio access networks, in International Workshop on Self-Organizing Systems. Springer, 2008, pp. 275–280.