A Minimum Spanning Tree-Based Approach for Reducing Verification Collisions in Self-Organizing Networks

Tsvetko Tsvetkov*, Janne Ali-Tolppa†, Henning Sanneck†, and Georg Carle*
*Department of Computer Science, Technische Universität München
Email: {tsvetko.tsvetkov, carle}@in.tum.de
†Nokia, Munich, Germany
Email:{janne.ali-tolppa, henning.sanneck}@nokia.com

***Abstract*—The verification of Configuration Management (CM) changes has become an important step in the operation of a mobile Self-Organizing Network (SON). Typically, a verification mechanism operates in three phases. At first, it partitions the network into verification areas, then it triggers an anomaly detection algorithm for those areas, and finally generates CM undo requests for the abnormally performing ones. Those requests set the CM parameters to a previous stable state. However, verification areas may overlap and share anomalous cells which results in a verification collision. As a consequence, the verification mechanism is not able to simultaneously deploy the undo requests since there is an uncertainty which to execute and which to potentially omit. In such a case, it has to serialize the deployment process and resolve the collisions. This procedure, though, can be negatively impacted if unnecessary collisions are processed, since they might delay the execution of the queued CM undo requests.**

To overcome this issue, we propose an approach for changing the size of the verification areas with respect to the detected collisions. We achieve our goal by using a Minimum Spanning Tree (MST)-based clustering approach that is able to group similarly behaving cells together. Based on the group they have been assigned to, we remove cells from a verification area and prevent false positive collisions from being further processed. Furthermore, we evaluate the proposed solution in two different scenarios. First, we highlight its benefits by applying it on CM and Performance Management (PM) data collected from a real Long Term Evolution (LTE) network. Second, in a simulation study we show how it positively affects the network performance after eliminating the false positives.

I. Introduction

Nowadays, operators of mobile communication networks need to find a convenient way of configuring and optimizing their network. Due to the fact that we have a wide adoption of mobile communication services by users, the requirements to efficiently manage deployed Network Elements (NEs) and guarantee a high degree of reliability increase as well. As a consequence, the Self-Organizing Network (SON) concept has been introduced to cope with the complex nature of standards like Long Term Evolution (LTE), LTE-Advanced [1], and even the upcoming fifth generation of mobile communications (5G) [2]. Today, SON consists of features that automatically auto-configure newly deployed NEs, optimize the network, and deal with fault detection and resolution. Those features are implemented as SON functions, which typically resemble control loops that monitor Performance Management (PM) and Fault Management (FM) data collected from the network and, based on the type of tasks they are designed for, deploy sets of Configuration Management (CM) changes. An example is the Automatic Neighbor Relation (ANR) function which can create X2 connections between relay nodes and Evolved NodeBs (eNBs) as part of establishing neighbor relations between adjacent cells [1].

978-1-5090-0223-8/16/$31.00

In addition to the field of self-configuration and self-optimization, there is another research area that has become of particular interest, namely SON verification. This area concentrates on the performance analysis of all the deployed CM changes and the rollback of those having a negative impact on the network performance. Nowadays, a SON verification approach [3]–[7] is seen as a special type of anomaly detection, also known as the *verification process*. It operates in three phases: (1) it partitions the network into verification (observation) areas according to the CM changes, (2) runs an anomaly detection algorithm for each area, and (3) marks the changes for an undo that are most likely responsible for causing an abnormal behavior. Finally, it generates *CM undo requests* for the negatively impacted areas and observes whether the network's performance has improved afterwards.

One common fact about verification approaches is that verification areas are composed solely based on the neighbor relations between cells. Typically, they include the reconfigured cell as well as those neighbors that might be impacted by the reconfiguration. Hence, verification areas can overlap which hinders the verification mechanism to simultaneously deploy certain undo requests. If two verification areas share anomalous cells, there is an uncertainty which change to rollback since there might such not harming the performance. In literature, this type of conflict is known as a *verification collision* and is resolved by serializing the execution of conflicting undo requests. Based on the observations the verification mechanism has made, it has to deploy those requests at first that have the highest probability of restoring the performance.

However, finding those requests can be quite difficult if a high number of false positive collisions enter the verification process. This can happen since only the neighbor relations are decisive for collision detection, i.e., we have a too strict verification collision definition. For instance, a collision emerges if

one cell is optimized for coverage, another for handover, and a shared neighbor experiences coverage problems. Obviously, this particular collision is unnecessary and its presence may effectively delay the undo request execution since it needs to be resolved by the verification mechanism. To provide a solution, we propose a verification collision reduction approach that is based on graph theory. We use a Minimum Spanning Tree (MST) to depict the performance of the cells with respect to each other and to group the similarly behaving ones. Based on the outcome of the grouping process, we convert some areas to weak verification areas by reducing their size. Then, we use those areas for the elimination of unnecessary collisions.

Our paper is structured as follows. In Section II we give a detailed description of the problem as well as an overview of the verification process including examples how verification is realized in a mobile communication network. In Section III we present our concept. Section IV is devoted to the evaluation based on real as well as simulated network data. Our paper concludes with the related work in Section V and a summary in Section VI.

II. Background

A. The verification process

1) Composition of verification areas: A verification area consists of the reconfigured cell, also called the *target cell*, and a *target extension set* that includes the possibly impacted cells by the CM change(s). The composition itself is based on the neighbor relations between cells. One possible way of doing that is to take the reconfigured cell and all of its direct neighbors to which a handover of User Equipments (UEs) can be established. In addition, if we have SON function activities in the network, we may define the verification area as the impact area [8] of the SON function that has been recently active. Furthermore, we may enlarge a verification area based on its location [9], e.g., more cells are added if they are part of dense traffic and known trouble spots.

2) Detecting anomalies: In general, an anomaly is understood as something that significantly deviates from what we would typically expect. In this paper, however, we focus on detecting abnormal cell behavior, i.e., the performance of a cell has notably degraded. It is done by profiling [7] the network behavior, that is, analyzing the network performance indicators and specifying how the network should usually behave. An example of performing anomaly detection is presented in [10], where performance indicator normalization is used to determine whether cells are showing an expected behavior. Furthermore, the impact of different faults on the performance indicators can be learned as well.

3) Generating CM undo requests: For each degraded area an undo request is generated. It consists of three data fields: one that uniquely identifies the target cell in the network, another that consists of the identifiers of the impacted cells, and a list containing CM parameter values of the target cell. The latter one can be a complete snapshot consisting of all CM settings a cell has had at some earlier point in time. However, it can also be a partial list, e.g., if we rollback changes made by SON functions.

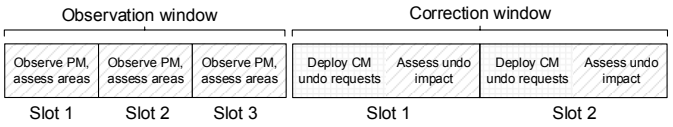


Figure 1. Example of a 3-slot observation and 2-slot correction window

B. Time windows and verification collisions

The verification process requires two time windows, each divided into one or more time slots, as depicted in Figure 1. The first one is known as the *observation window*, during which the performance impact of deployed CM changes is monitored. That is, PM data is collected, verification areas are generated, and the anomaly detection algorithm is triggered. Its length depends on the PM granularity periods [11], i.e., the PM data collection frequency. The second window, also referred to as the *correction window*, is used for the deployment of undo requests and for assessing their impact on the network performance. Its length is reliant on the time it takes to deploy the changes, also known as the CM granularity. In addition, it is subject to the environment, e.g., in a highly populated area we might have a short correction window since the restoration of the network performance has to be carried out as fast as possible.

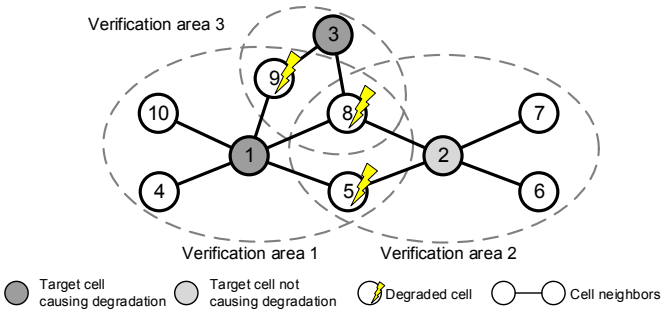
Furthermore, the correction window is used to resolve verification collisions. A collision takes place when two or more areas share anomalous cells, i.e., the corresponding undo requests address two overlapping sets of cells. If we denote the set of all cells as V, the set of all anomalous cells as V_a, where $V_a \subseteq V$, and the set of all verification areas as Ψ, two verification areas $\psi_i, \psi_j \in \Psi$ are said to be in a collision when $f_e(\psi_i) \cap f_e(\psi_j) \subseteq V_a$. Note that f_e is an extraction function that returns the cells of a verification area, i.e., $f_e \colon \Psi \to \mathcal{P}(V) \setminus \{\varnothing\}$. As a result, we have an uncertainty which change to rollback and which to omit.

To eliminate this uncertainty, the verification process creates a deployment plan which assigns each undo request to a correction window time slot. The plan itself must have the properties of being (1) collision-free and (2) degradation-aware. The first property implies that undo requests assigned to the same time slot are not in collision with each other. The second property makes sure that the deployment order maximizes the probability of returning the network performance to the expected state. That is, undo requests for CM changes that are most likely causing a degradation are allocated to the first time slot.

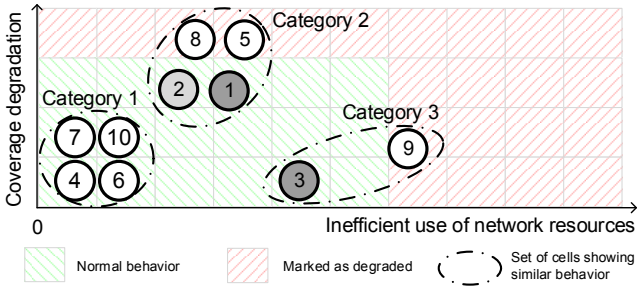
Nevertheless, it might not be always possible to guarantee that the plan is collision-free due to the lack of a sufficient high number of correction window time slots. Therefore, the verification process has to group undo requests while minimizing the probability of rolling back changes that did not harm the performance.

C. Problem description

The presence of *false positive verification collisions* can negatively impact the verification process, as shown in the following example. Suppose that we have 10 cells, as given in Figure 2(a), three of which (cell 1, 2, and 3) have been

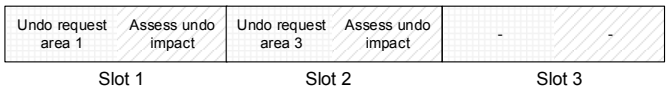


(a) Verification area selection based on the neighbor relations



(b) Performance behavior of the cells

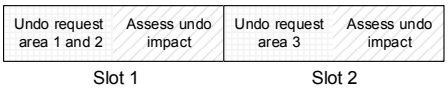
Figure 2. Example of a verification collision



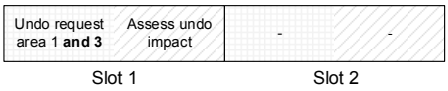
(a) Slot allocation without eliminating false positive collisions

(b) Slot allocation after eliminating false positive collisions

Figure 3. Correction window of 3 slots



(a) Slot allocation without eliminating false positive collisions



(b) Slot allocation after eliminating false positive collisions

Figure 4. Correction window of 2 slots

reconfigured and another three (cell 5, 8, and 9) have degraded. In addition, let us assume that the verification mechanism is unaware that cell 2 is not responsible for any degradation. As a result, we have three undo requests, one being generated for each verification area. Note that an area covers the first degree neighbors of the target cell and takes its identifier. Hence, the verification mechanism assigns the undo request for area 1 to the first slot since it includes the largest number of degraded cells. Should after the deployment cell 5 and 8 stop showing an anomalous behavior, it will omit the undo request for area 2 and process the one for area 3, as shown in Figure 3(a).

As a next step, let us observe the coverage degradation reports of the cells as well as how they utilize network resources. A prominent example for inefficient use of network resources are unnecessary handovers, also called ping-pongs [1]. They are repeated between two cells within a short time, leading to reduced user throughput. Figure 2(b) gives us an exemplary position of each cell in the $\mathbb{R}^2$ space, based on those reports. As shown, the cells can be grouped into three categories: (1) cells showing normal behavior, (2) cells experiencing coverage degradation, and (3) cells inefficiently using network resources. Although verification area 3 initially shares anomalous cells with area 1 and 2, it should not be in collision with them since neither cell 1 has been assigned to the group of cell 3 and 9, nor has cell 3 been grouped together with cell 2 and 8. Cell 1, 2, 5, and 8 are clearly experiencing coverage degradation whereas cell 3 and 9 an inefficient use of resources. Hence, we have two false positive collisions: one between area 1 and 3 and another between 2 and 3. If we had omitted those collisions, though, we could have simultaneously deployed the undo requests for area 1 and 3, as depicted in Figure 3(b). In other words, we could have restored the performance of the network much faster.

This can get even worse in case we have a short correction window, as shown in Figure 4. If we decrease the number of correction window time slots to two, the verification mechanism will be forced to group some undo requests. A possible outcome is to combine those for area 1 and 2 since together they include the largest number of cells. Consequently, the verification mechanism will assign those two requests to the first slot and delay the one for area 3, as Figure 4(a) outlines. Obviously, if we had eliminated the unnecessary collisions we could have simultaneously deployed the requests for area 1 and 3, as presented in Figure 4(b).

D. Causes for verification collisions

Verification collisions tend to occur when a *large number* of overlapping verification areas *simultaneously enter the verification process*. The first cause for this to happen is the duration of the PM granularity period. As stated in [11], due to the relative network and processing overhead of getting the data, the lower bound is set to 15 minutes. However, long granularity periods increase the probability of SON functions becoming simultaneously active. Thereby, a lot of verification areas might get processed at once which increases the likelihood of verification collisions.

The second cause is the location of the verification mechanism. Since it needs to have a wide view on the ongoing network changes, it is typically placed at the Domain Management (DM) or Network Management (NM) level of the Operation, Administration and Management (OAM) architecture [1]. Being at that level, though, means that is not in a position to instantly verify all ongoing reconfigurations. Instead, it

has a coarse-granular view on the CM data, which increases the potential many verification areas to enter the verification process.

The third cause is the heterogeneous nature of mobile SONs. Today, they are comprised of macro cells, but may also include many small cells that provide hot-spot coverage. As a result, the neighborship degree increases which affects the verification area size. In addition, the upcoming 5G standard is supposed to include even more heterogeneous networks, different types of access technologies, as well as a high number of small cells densely clustered together [12]. These factors create a potential for more verification collisions to emerge.

The fourth cause is the delay until we get statistically relevant PM data. A significant change in Key Performance Indicators (KPIs) like the Channel Quality Indicator (CQI) or the Handover Success Rate (HOSR) can be only seen when we have enough users that actively use the network, e.g., during the peak hours of a weekday. Consequently, a verification mechanism is able to assess certain CM changes only when such data is available, which may lead more areas to simultaneously enter the verification process.

III. Concept

Our approach is split into three phases. During the first one, we compose a behavior graph that depicts how the cells perform in the network. During the second one, we transform the behavior graph into an MST which is used to produce a set of cell clusters. Those clusters group similarly behaving cells together. In addition, the generated clusters play a crucial role during the third phase where the resizing of the verification areas and elimination of false positive collisions takes place.

A. Behavior graph composition

The behavior of each cell is specified by an anomaly vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ that is element of $\mathbb{R}^n$. The value of n equals $|K|$, where K is a set of KPIs that allow us to determine the cell performance status. Each element $a_k \in \mathbb{R}$, where $k \in [1, n]$, is called a *KPI anomaly level* and its purpose is to represent the deviation of the given KPI from the expected value.

Then, we define a function δ (Equation 1) that gives us the distance between $\mathbf{a}_i, \mathbf{a}_j$ for all i, j.

$$\delta \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \tag{1}$$

Based on the outcome, we construct an undirected edge-weighted graph $G = (V, E)$, where V is the set of vertexes and E the set of edges $V \times V$. Each vertex $v_i \in V$ is represented by an anomaly vector $\mathbf{a}_i$ in the space $\mathbb{R}^n$, i.e., there is a function $m \colon V \to \mathbb{R}^n$ that maps a vertex to a point in $\mathbb{R}^n$. In addition, the weight of every edge $(v_i, v_j) \in E$, denoted as $w(v_i, v_j)$, is computed by applying δ.

An example of how this process may look like is given in Figure 5. The network consists of 12 cells and 18 cell adjacencies. In addition, we have five CM changes (cell 1, 2, 6 , 8, and 11), and four degradations (cell 3, 4, 7, and 10). Furthermore, a verification area consist of the direct neighbors of the target cell and is identified by its ID. This leads the following verification areas to be in collision: (1, 2), (2, 6), (6, 8), and (8, 11). One possible composition of the behavior graph in the $\mathbb{R}^2$ space is given in the same figure. Note that for simplicity reasons, some vertexes are completely overlapping, i.e., they represent cells that are showing exactly the same behavior. As a consequence, the edges connecting those vertexes are omitted from the graph.

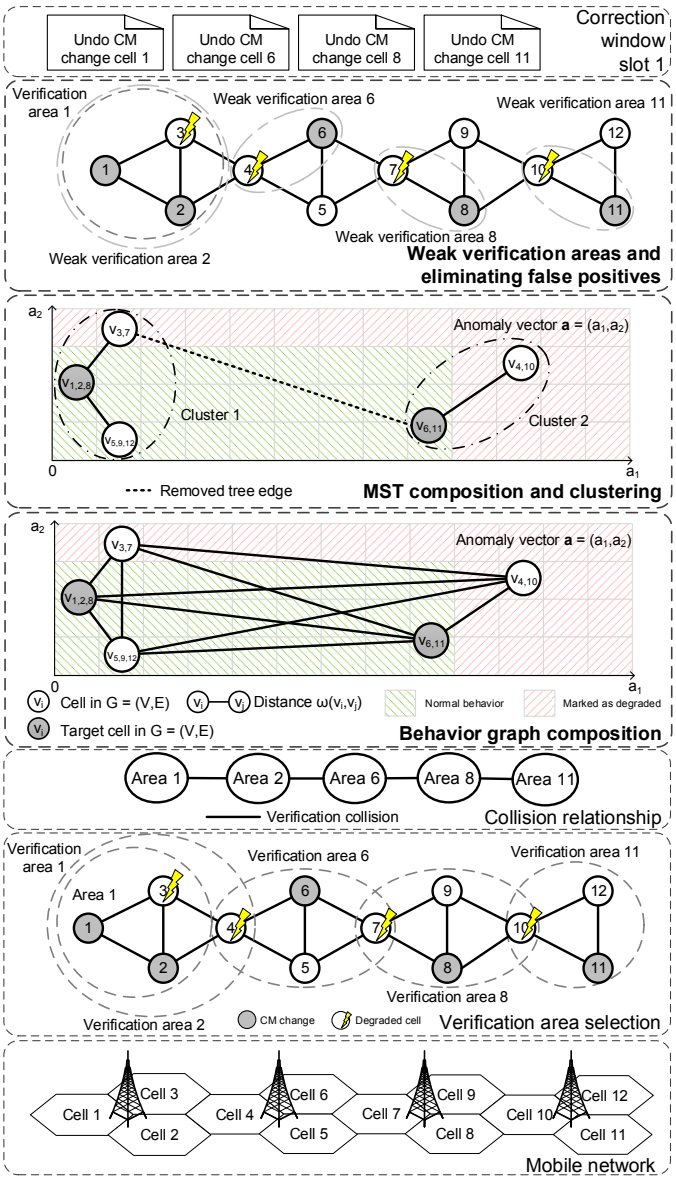


Figure 5. Concept example

B. Minimum spanning tree composition and clustering

Next, we form an MST by searching for an undirected subgraph $T = (V, E_t)$, where $E_t \subseteq E$, that minimizes $\sum_{v_i, v_j \in G} w(v_i, v_j)$. In particular, we are making use of Kruskal's algorithm, also characterized as being a greedy algorithm that continuously takes edges from a sorted list and adds those to the graph without forming a loop [13].

After constructing the MST, we divide the vertexes into κ groups so that the minimum distance between vertexes of different groups is maximized. To do so, we form a forest F,

i.e., an undirected graph whose connected components are trees, each being disjoint with the other. If we denote those trees as $\{T'_1, T'_2, \ldots, T'_\kappa\}$, the following condition must hold: $\forall i, j : V'_i \cap V'_j = \{\varnothing\}$, where V'_i and V'_j are the sets of vertexes of T'_i and T'_j, respectively. This procedure, also known as MST clustering [14], is carried out by removing a certain number of edges from T by starting from the longest one. For instance, the removal of the two longest edges results into a forest of three trees, i.e., a group of three clusters.

The decision whether to remove an edge from T depends on the edge removal threshold. The latter one is calculated by an edge removal function, as defined in Equation 2.

$$r \colon T \to \mathbb{R} \tag{2}$$

If we recall the example from before, the cells may form an MST, as shown in Figure 5. The tree consists of five vertexes and the following edges: $(v_{1,2,8}, v_{5,9,12})$, $(v_{1,2,8}, v_{3,7})$, $(v_{6,11}, v_{4,10})$, and $(v_{3,7}, v_{6,11})$. The removal of the latter edge leads to the formation of two clusters, namely ($v_{1,2,8}$, $v_{5,9,12}$, $v_{3,7}$) and ($v_{4,10}$, $v_{6,11}$).

C. Weak verification areas and eliminating false positives

After forming the MST, we can distinguish between the following three outcomes:

- $\kappa = 1$
- $\kappa = |V|$
- $\kappa = [2, |V|)$

In the first case, the forest F has only one tree, i.e., we have only one cluster to which all cells have been assigned to. Therefore, they are seen as behaving in the same manner which impedes the removal of any collision. In the second case, the forest F has $|V|$ trees each consisting of one vertex, i.e., each cell is in a different cluster. Consequently, all collisions are marked as being unnecessary and are, therefore, removed. In the third case, we start transforming some areas to *weak verification areas* by reducing their size. Cells from a target extension set that are not assigned to the cluster of the target cell are excluded from the given area. In contrast, areas whose cells are all located within the same cluster remain unchanged. The verification collisions that have disappeared after this step are marked as false positives and are omitted.

If we apply this step in our example from Figure 5, four of the verification areas are converted to weak ones: area 2, 6, 8, and 11. Hence, only the collision between area 1 and 2 remains. One permissible outcome is the assignment of the undo requests for cell 1, 6, 8, and 11 to the first correction window time slot.

IV. Evaluation

A. Parameter selection

1) Anomaly vector: To compute a KPI anomaly level, i.e., an element a_k of an anomaly vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, we define a training phase during which we collect KPI samples $p_1 \ldots p_t$, where t marks a training period. During this phase the network has to show an expected behavior. Then, we measure the current value of the KPI, denoted as p_{t+1}. The collected data, i.e., $p_1 \ldots p_t, p_{t+1}$, is standardized by computing the z-score of each data point. The KPI anomaly level is the z-score of p_{t+1}. Note that the duration of a training period is subject to the PM granularity period.

To give an example suppose that a cell has reported a HOSR of 99.9 %, 99.9 %, 99.1 %, 99.7 %, 99.8 %, and 99.6 % during the training phase. Moreover, let us assume that 90.2 % is the current HOSR. After normalizing the samples, we get the following outcome: 0.44, 0.44, 0.21, 0.38, 0.41, 0.35, and -2.26. The cell HOSR anomaly level is -2.26, which is the z-score of last data point.

After computing the KPI anomaly levels, each verification area is tested for degradation. An area is marked as such when one KPI anomaly level of one cell falls in the range $(-\infty; -2.0]$ for success KPIs (e.g., HOSR) or $[2.0; \infty)$ for failure KPIs (e.g., handover ping-pong rate, call drop rate). Note that the KPI selection differs for each study.

2) Distance computation: The distance between two anomaly vectors $\mathbf{a} = (a_1, a_2, \ldots a_n) \in \mathbb{R}^n$ and $\mathbf{a}' = (a'_1, a'_2, \ldots a'_n) \in \mathbb{R}^n$ is computed by applying the Pythagorean formula from Equation 3.

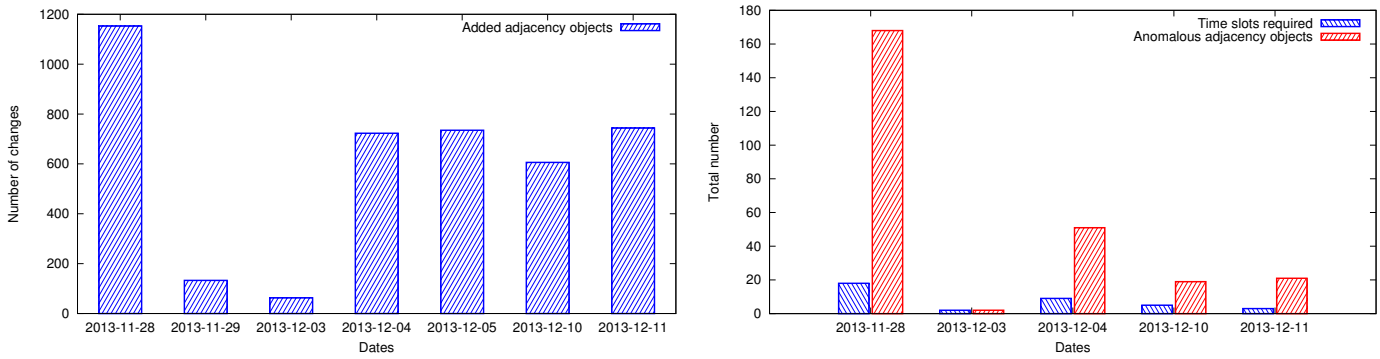
$$\delta(\mathbf{a}, \mathbf{a}') = \delta(\mathbf{a}', \mathbf{a}) = \sqrt{\sum_{k=1}^{n} (a'_k - a_k)^2} \tag{3}$$

Hence, T from Section III is an Euclidean MST, in which each edge weight $w(v_i, v_j)$ equals the Euclidean distance between $v_i \in V$ and $v_j \in V$. The edge removal function r, based on which the forest F is formed, differs for each study.

3) Deployment of undo request: The strategy for deploying undo requests has been introduced in [15]. To assemble a plan that is collision-free and degradation-aware, we make use of constraint optimization [16]. We assign each request a variable which may accept values between one and the maximum number of correction window time slots. In addition, we add a constraint for each collision that prevents two variables from receiving the same value. We also scale the variables with a so-called performance rating value (the higher the better), calculated for the corresponding verification areas. Finally, we sum up those variables and maximize the resulting equation. The requests associated with the variable receiving the lowest value are deployed at first. Furthermore, if we cannot assemble a deployment plan because of a short correction window, we start using soft constraints. They are able to provide us with a solution that minimizes the total constraint violation.

B. Real data study

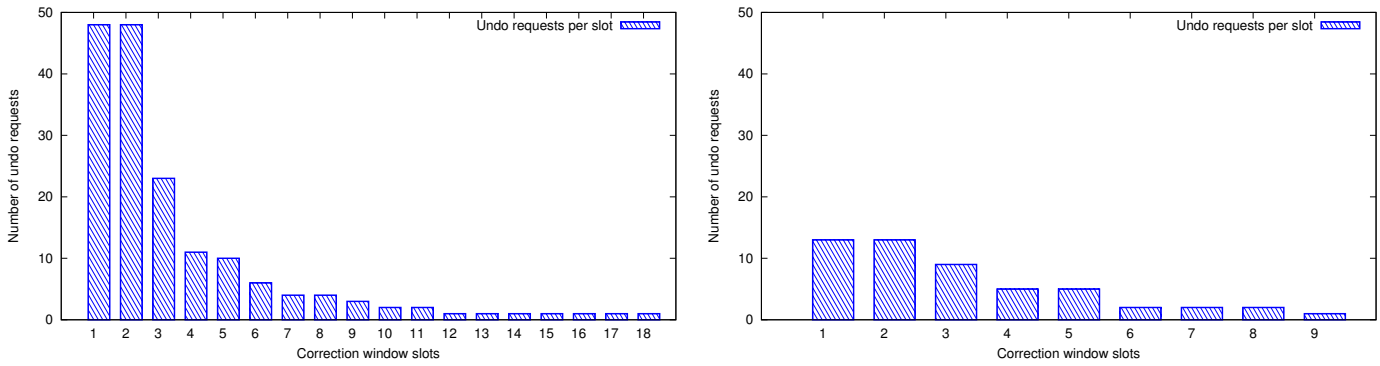
1) Environment: Our real data study is based on PM and CM data dumps generated by an LTE network. As outlined in Figure 6(a), we found seven days during which a high number of adjacency managed objects have been added. Moreover, we know that two SON functions have been active: the ANR and the Physical Cell Identity (PCI) allocation function, as defined in [1]. The list of KPIs has been exported on hourly basis, as stated in [6]. The CM changes have been obtained from the history database once every day.



(a) Cell adjacency changes

(b) Performance analysis

Figure 6. Analysis of the LTE network



(a) 28^{th} of November

(b) 4^{th} of December

Figure 7. Distribution of the undo requests

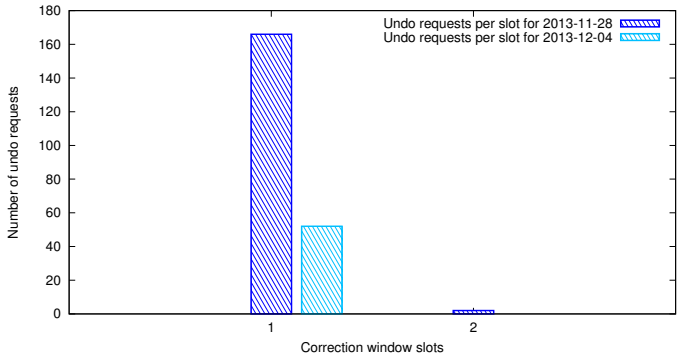


Figure 8. Distribution of the undo requests after eliminating the unnecessary collisions

As for the verification area composition, we select the cells of the reconfigured adjacency. Furthermore, we take the following KPIs for the calculation of the anomaly vectors:

- `EUTRAN_RLC_PDU_RETR_R_DL`: retransmission rate for Radio Link Control (RLC) Protocol Data Units (PDUs) in downlink direction.
- `EUTRAN_RLC_PDU_RETR_R_UL`: retransmission rate for RLC PDUs in uplink direction.

2) Results: At first, our goal is to find CM changes followed by an abnormal KPI value. In particular, we are interested in the number of adjacency objects that have negatively impacted one of the two KPIs for two consecutive hours and, therefore, have been processed by the verification mechanism. Figure 6(b) gives us the result of this observation as well as the number of time slots we need if we attempt to rollback the changes.

Second, we observe how the undo requests are distributed over the correction window time slots. In Figure 7 we have outlined the slot assignment for the days during which we managed to spot a high number of anomalies. As shown, the attempt of rolling back the changes for the 28^{th} of November requires 18 time slots, whereas for the 4^{th} of December 9 time slots. The question that arises here is how the distribution changes in case we apply our verification collision reduction approach. If we remove the edges of T whose weight exceeds the 99^{th} percentile of all edge weights, we get a slot allocation

as presented in Figure 8. The verification mechanism would require two slots for the 28^{th} of November and one for the 4^{th} of December, which is a significant decrease compared to the original slot allocation.

C. Simulation study

1) Environment: The simulation environment, called the SON Simulation System (S3), consists of an LTE radio network simulator, a set of SON functions, and a SON function coordinator, as presented in [3].

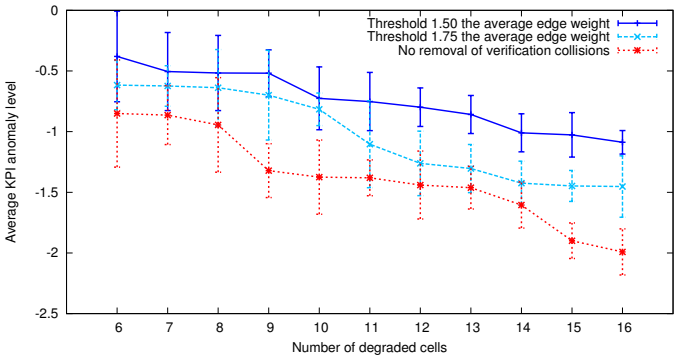
The LTE simulator, as part of the SON simulator/emulator suite [17], performs continuous simulation by tracking the changes in the network over time. The time itself is divided into time slices, called simulation rounds, each corresponding to 100 minutes in real time. At the beginning of a round, the simulator configures the network as defined by the CM parameter setup. During a round, 1500 uniformly distributed users follow a random walk mobility model (speed of 6 km/h) and actively use the mobile network. At the end of a round, PM data is exported for every cell and used as input by every SON function. Furthermore, a handover occurs immediately when a UE crosses the hysteresis threshold of 2.0 dB. A radio link failure happens based on a signal-to-interference-plus-noise ratio comparison to a threshold of -6.0 dB. The simulated area is an LTE macro cell network, consisting of 32 LTE macro cells spread over an area of 50 km^2. The network carrier frequency is set to 2000 MHz whereas the channel bandwidth is set to 20 MHz. The used SON functions are Mobility Robustness Optimization (MRO), Remote Electrical Tilt (RET), Transmission Power (TXP), as specified in [1], as well as a verification function implementing the concept introduced in Section III. The SON function coordinator we use follows the design principles outlined in [11].

The training period t of the verification mechanism is set to a simulation round. The training data is collected during a separate test run, lasting 70 rounds, during which optimal network settings are used. In addition, two success KPIs are of particular interest, namely the HOSR and CQI, as well as one failure KPI, the handover ping-pong rate [1]. It should be noted that the CQI is computed as the weighted harmonic mean of the CQI channel efficiency. The efficiency values are listed in [18].

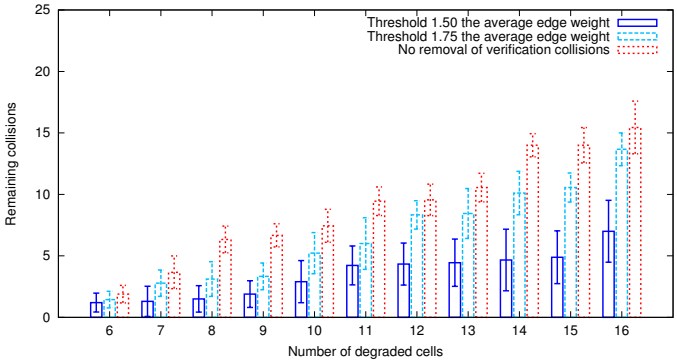
Furthermore, a verification area consists of the target cell and its direct neighbors. To form the forest F, we remove all edges from T whose weight exceeds 1.5 (first configuration) or 1.75 (second configuration) times the average edge weight.

2) Results: The purpose of the simulation study is to observe how the network performance behaves while turning our approach on and off. As we saw in the real data study most requests got assigned to the first time slot which is the reason why we are mostly curious about the network behavior after processing it. Furthermore, we are interested in how our approach affects the network performance as the number of degraded cells increases.

To perform the evaluation, we set a test run to last 5 simulation rounds, during which all SON functions are allowed to optimize the network. At the beginning of a test run certain cells are selected for degradation. The exact number



(a) Average KPI anomaly level of the network



(b) Number of remaining collisions

Figure 9. Results after processing the first correction window time slot

of degraded cells ranges between 6 and 16 and is selected based on a uniform distribution. The degradation itself is carried out by deploying two untypical configurations. On the one hand, the coverage of half of the cells is changed by setting their transmission power to 40 dBm. On the other hand, the handover capability of the other half is manipulated after selecting a Cell Individual Offset (CIO) of -5.0.

As a next step, we take the negative KPI anomaly levels in case of failure KPIs, and leave those of success KPIs unchanged. Then, we compute the overall network performance by averaging all KPI anomaly levels reported by all cells after processing the first correction window time slot. That is, the higher the average, the better the performance.

The results of this study are shown in Figure 9(a). Note that it depicts the 95% confidence intervals around the sample mean of 9 consecutive test runs. As the observations indicate, the elimination of unnecessary collisions manages to significantly improve the network performance already after deploying the first set of undo requests. Moreover, the performance improves when we lower the threshold to 1.5 times the average edge weight. The answer why we are able to see this improvement is given in Figure 9(b). It shows how many undo requests remain in collision after processing the first time slot. Similarly to the real data study, we are able to allocate more requests to the first slot after removing the false positives and, therefore, have less entering the verification process afterwards. However, if we completely disable our approach, we are not able to see this improvement due to the large number of collisions preventing the simultaneous deployment of some undo requests.

Despite the results, an important remark should be made

here. The edge removal function r has to be used with caution since the deletion of too many edges from T may lead us to the point where we have no verification collisions at all. As a result, the verification mechanism will start undoing too many changes, including such that did not harm the network performance. Here, we started seeing this effect as we selected a threshold below 1.5 times the average edge weight.

V. Related Work

The concept of pre-action SON coordination [8] can be seen as an alternative to SON verification. It is understood as a pessimistic strategy that specifies rules required for the detection and resolution of *known* conflicts between active SON function instances. It prevents conflicting functions from getting active, rather than assessing the network performance after deploying the CM changes. One example for such conflicts is the operation on shared CM parameters within the same physical area. A second example is when the activity of one function affects the input measurements of another one.

In [19], an anomaly detection technique for cellular networks has been introduced. It is based on the extended version of the incremental clustering algorithm Growing Neural Gas (GNG) which partitions the input data into smaller groups with a similar behavior. A method like this is able to identify unusual behavior in the time domain. An example is a cell remaining a whole week in a state that represents the weekend. The presented approach, however, does not consider the problem of unnecessary verification collisions.

In [5], an anomaly detection and diagnosis framework has been proposed. It attempts to verify the effect of CM changes by monitoring the state of the network. The framework operates in two phases: (1) it detects anomalies by using topic modeling, and (2) performs diagnosis for any anomaly by using Markov Logic Networks (MLNs). In the latter case it makes use of probabilistic rules to differentiate between different causes. Again, the presented solution does not address the problem of having false positive collisions.

Similarly, [7] presents an anomaly detection and diagnosis framework. The introduced system learns the faultless behavior of the network and stores that into profiles, which are also seen as statistical distributions of the performance indicator samples. The anomaly detection relies on a common statistical primitive, derived from the two-sample Kolmogorov-Smirnov (KS) test. Furthermore, the diagnosis part may learn the impact of different faults on the performance indicators in order to provide a corrective action. This is done by using a scoring system that rewards a given action if it has had a positive effect on the network. The presented framework does not address the issues outlined in this paper.

VI. Conclusion

The verification of Configuration Management (CM) changes in a mobile Self-Organizing Network (SON) is three step process during which we partition the network into verification areas, trigger an anomaly detection algorithm, and generate CM undo requests for those experiencing a degradation in performance. The purpose of those requests is to return the badly performing areas to a previous stable state. A verification area typically consist of the reconfigured cell, also called the target cell, and a set of potentially impacted cells. However, degraded verification areas may share anomalous cells, which means that they are in a so-called verification collision. As a consequence, there is an uncertainty which of the target cells to rollback at first, i.e., we cannot simultaneously deploy the corresponding undo requests. Typically, in such a case the deployment process is serialized, e.g., by starting executing the undo request for the mostly degraded area.

However, a problem that has so far been neglected is the presence of false positive collisions. Such collisions delay the verification process which means that the network needs more time to recover. In this paper, we have proposed a method based on graph theory that overcomes this issue. At first, we construct a graph that shows how cells behave with respect to each other. In addition, we transform the graph into a Minimum Spanning Tree (MST) which is later divided into two or more trees. As a result, we get a forest of trees which we take as a set of cell clusters. Based on where the cells have been assigned to, we reduce the size of the verification areas and remove the unnecessary collisions.

The evaluation of our approach is twofold. On the one side, we apply it on real Long Term Evolution (LTE) data and show its ability to eliminate false positives. On the other side, we evaluate our method in a simulated environment by toggling this feature. The results show that our method is able to successfully eliminate the unnecessary collisions and return the network close to the expected performance state already after the first set of deployed undo requests.

References

[1] S. Hämäläinen, H. Sanneck, and C. Sartori, Eds., *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Chichester, UK: John Wiley & Sons, Dec. 2011.

[2] Ericsson, "5G: what is it?" White paper, Oct. 2014.

[3] T. Tsvetkov, H. Sanneck, and G. Carle, "A Graph Coloring Approach for Scheduling Undo Actions in Self-Organizing Networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*, Ottawa, Canada, May 2015.

[4] G. Ciocarlie, C.-C. Cheng, C. Connolly, U. Lindqvist, S. Nováczki *et al.*, "Managing Scope Changes for Cellular Network-level Anomaly Detection," in *International Workshop on Self-Organizing Networks (IWSON 2014)*, Barcelona, Spain, Aug. 2014.

[5] G. Ciocarlie, C. Connolly, C.-C. Cheng, U. Lindqvist, S. Nováczki *et al.*, "Anomaly Detection and Diagnosis for Automatic Radio Network Verification," in *6th International Conference on Mobile Networks and Management (MONAMI 2014)*, Würzburg, Germany, Sep. 2014.

[6] T. Tsvetkov, S. Nováczki, H. Sanneck, and G. Carle, "A Configuration Management Assessment Method for SON Verification," in *International Workshop on Self-Organizing Networks (IWSON 2014)*, Barcelona, Spain, Aug. 2014.

[7] S. Nováczki, "An Improved Anomaly Detection and Diagnosis Framework for Mobile Network Operators," in *9th International Conference on Design of Reliable Communication Networks (DRCN 2013)*, Mar. 2013.

[8] T. Bandh, "Coordination of autonomic function execution in Self-Organizing Networks," PhD Thesis, Technische Universität München, Apr. 2013.

[9] Ericsson, "Transparent Network-Performance Verification For LTE Rollouts," White Paper, 284 23-3179 Uen, Sep. 2012.

[10] P. Kumpulainen, M. Särkioja, M. Kylväjä, and K. Hätönen, "Finding 3G Mobile Network Cells with Similar Radio Interface Quality Problems," in *Engineering Applications of Neural Networks, L. Iliadis and C. Jayne, Eds. Springer Berlin Heidelberg*, 2011, pp. 392–401.

[11] R. Romeikat, H. Sanneck, and T. Bandh, "Efficient , Dynamic Coordination of Request Batches in C-SON Systems," in *IEEE Veh. Technol. Conf. (VTC Spring 2013)*, Dresden, Germany, Jun. 2013.

[12] Next Generation Mobile Networks Alliance, "A Deliverable by the NGMN Alliance: NGMN 5G White Paper," Feb. 2015, final deliverable, version 1.0.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction To Algorithms*, 3rd ed. MIT Press, 2009, ISBN 0262258102.

[14] P. K. Jana and A. Naik, "An Efficient Minimum Spanning Tree based Clustering Algorithm," in *International Conference on Methods and Models in Computer Science*, Delhi, India, Dec. 2009, pp. 1–5.

[15] T. Tsvetkov, C. Frenzel, H. Sanneck, and G. Carle, "A Constraint Optimization-Based Resolution of Verification Collisions in Self-Organizing Networks," in *IEEE Global Communications Conference (GlobeCom 2015)*, San Diego, CA, USA, Dec. 2015.

[16] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.

[17] NSN, "Self-Organizing Network (SON): Introducing the Nokia Siemens Networks SON Suite - an efficient, future-proof platform for SON," White Paper, Oct. 2009.

[18] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," 3rd Generation Partnership Project (3GPP), Technical Specification 36.213 V12.1.0, Mar. 2014.

[19] B. Gajic, S. Nováczki, and S. Mwanje, "An Improved Anomaly Detection in Mobile Networks by Using Incremental Time-aware Clustering," in *IFIP/IEEE Workshop on Cognitive Network and Service Management (CogMan 2015)*, Ottawa, Canada, May 2015.