Verification of Configuration Management Changes in Self-Organizing Networks

Tsvetko Tsvetkov, Janne Ali-Tolppa, Henning Sanneck, and Georg Carle

***Abstract*—The verification of Configuration Management (CM) changes is an essential operation in a mobile Self-Organizing Network (SON). Usually, a verification approach operates in three steps: it divides the network into verification areas, triggers an anomaly detection algorithm for those areas, and finally generates CM undo requests for the abnormally performing ones. Those requests set CM parameters to a previous stable state. However, the successful completion of this process can be quite challenging, since there are factors that might negatively impact its outcome. For instance, if a temporal degradation occurs during the optimization of a cell, a verification mechanism may wrongly assume that it is anomalous, and interrupt the optimization process. Furthermore, a verification strategy experiences difficulties when it faces verification collisions, i.e., conflicting undo requests that cannot be simultaneously deployed. At first, it has to determine whether the collision is a false positive one. Then, it has to resolve it by finding out which requests have the highest probability of restoring the network performance. In addition, it needs to consider the time that is given for rolling back CM changes.**

In this paper, we contribute to the area of SON verification by providing a solution that addresses those problems. Our approach makes use of constraint optimization techniques to resolve collisions as well as find the appropriate order for deploying undo requests. In addition, we use a Minimum Spanning Tree (MST)-based clustering technique to eliminate false positive collisions. Further, we evaluate our solution in a simulation study in which we show its positive effect on the network performance.

***Index Terms*—Self-Organizing Network, SON, Anomaly detection, SON verification, Configuration rollback**

I. Introduction

THE Self-Organizing Network (SON) concept as we know today has been developed to deal with the complex nature of standards like Long Term Evolution (LTE) and LTE-Advanced. Its purpose is to optimize the operation of the network, supervise the configuration and auto-connectivity of newly deployed Network Elements (NEs), and enable automatic fault detection and resolution [1]. To be able to perform those tasks, though, a SON-enabled network has to be managed by a set of autonomous SON functions that are designed to perform specific network management tasks. Typically, they are implemented as control loops which monitor Performance Management (PM) and Fault Management (FM) data, and based on their goals, change Configuration Management (CM) parameters. One example is the Coverage and Capacity Optimization (CCO) function which has been developed to optimize the coverage within a cell by changing the transmission power or the antenna tilt. Another one is the Mobility Load Balancing (MLB) function that has been designed to move traffic from highly loaded cells to neighbors as far as interference and coverage allows by adjusting the Cell Individual Offset (CIO) [2].

Nevertheless, the increasing reliance on SON features to perform the correct optimization tasks creates a new set of challenges. In a SON, a function's decision to change certain CM parameters depends not only on the environment, but also on the actions taken by other active functions. For example, the outcome of the CCO optimization process impacts any upcoming decision of an MLB instance monitoring the affected cells. Consequently, any inappropriate CM change deployed to the network may lead to a set of bad decisions in the future.

For this reason, the concept of SON verification has been developed [3]–[5]. Today, it is referred to as a special type of anomaly detection, also known as a *verification process*, that operates in three phases. At first, based on the ongoing CM changes, it partitions the network into sets of cells, also called verification or observation areas. Second, it assesses the performance of each area by triggering an anomaly detection algorithm and tries to find those areas that are abnormally performing. Third, a decision is made to either accept the deployed CM changes or to revert some of them back to a previous stable state. In the latter case, the changes most likely being responsible for causing a degradation are combined into a *CM undo request*, being generated for each degraded area.

Nevertheless, the process of verifying CM changes and rolling some of them back is certainly not a straightforward procedure. On the one hand, if the network is sampled for a too short time period, the verification process does not have the opportunity to examine in detail the impact of a CM change. Therefore, it may not have enough knowledge to assess whether a change is harming the network performance. On the other hand, if it observes the network for too long, it would need to analyze the impact of a large set of CM changes which can lead to *verification collisions*. A collision is an indication for being uncertain which change to rollback as we try to simultaneously deploy two or more undo requests impacting a shared set of anomalous cells. For instance, if we change the transmission power within one cell, adjust the antenna tilt within the other, and a common neighbor of both cells start showing an abnormal behavior, we would need to find out which change to rollback and which to accept.

Even if we manage to handle a single collision, the process of resolving a large set of such can be quite challenging, especially when *false positive collisions* enter the verification

Tsvetko Tsvetkov and Georg Carle are with the Department of Computer Science, Technische Universität München, e-mail: ({tsvetko.tsvetkov, carle}@in.tum.de).

Janne Ali-Tolppa and Henning Sanneck are with Nokia Bell Labs, Munich, Germany, e-mail: ({janne.ali-tolppa, henning.sanneck}@nokia.com).

Manuscript received February 28, 2016; revised June 24, 2016.

process. For example, a verification collision occurs if one cell is optimized for coverage, another one for handover, and a shared neighbor starts experiencing coverage problems. Obviously, this collision is unnecessary and it delays the verification process by suppressing one of the undo requests.

Unfortunately, false positive collisions are not the only challenge for SON verification. Our time to resolve valid collisions, deploy the appropriate undo requests, and assess their impact on the network is *limited*. For example, we may have five undo requests each being in collision with the other, but be allowed to correct actions only twice a day.

In this paper, we contribute to the area of SON verification by providing a solution to those problems. At first, we present an adaptive observation strategy for SON verification which tries to find the optimal time for sampling the network. The proposed method is based on exponential smoothing of the deviation from the expected cell performance. Second, we propose an approach that eliminates false positive collisions. It uses a Minimum Spanning Tree (MST)-based clustering approach that eliminates unnecessary verification collisions by grouping similarly behaving cells. Third, we introduce a solution for resolving valid collisions as well as a method that handles the time limitation constraint. Both are based on constraint optimization techniques.

The reminder of this paper is structured as follows. In Section II we give a detailed description of the verification process. Section III presents the challenges of SON verification whereas Section IV the factors influencing the verification process. In Section V we present our concept. Section VI is devoted to the simulation environment whereas Section VII outlines the results of our study. The paper concludes with the related work in Section VIII and a summary in Section IX.

II. Background

A. The verification process

1) Composition of verification areas: A verification area is composed by taking the reconfigured cell, also referred to as the *target cell*, and a set of cells surrounding it, called the *target extension set*. In particular, the extension set consists of cells that are possibly impacted by the reconfiguration of the target. Typically, they are selected based on the existing neighbor relations, e.g., by taking all first degree neighbors of the reconfigured target. Furthermore, in case of SON activities, we may define the verification area as the impact area of the SON function that has been recently active. The impact area is the spatial scope within which a SON function modifies CM parameters and where it takes its measurements from [6]. In addition, the size of a verification area can be subject to the location of the cells [7]. For instance, if a cell is part of dense traffic or known trouble spots, it might get forced to join a verification area, even if it was not initially supposed to do so.

2) Detecting abnormal behavior: An anomaly is understood as "something that deviates from what is standard, normal, or expected" [8]. In this paper, though, we focus on detecting abnormal cell behavior, that is, the performance of a cell has notably degraded. To do so, we need to profile the network behavior [9], which is carried out by analyzing the network performance indicators and specifying how the network should usually behave. Once the cell performance considerably differs from all learned states of the normal network operation, it is marked as anomalous and the corresponding corrective actions have to be applied.

One possible way of learning faultless states is to make use of clustering algorithms like K-means [10] or Self-Organizing Maps [11]. Another example is the solution described in [5] which implements an extended version of the incremental clustering algorithm Growing Neural Gas (GNG) [12].

3) Generation of CM undo requests: Each verification area that is marked as being degraded has to be returned to a stable state. For this purpose, CM undo requests are generated, each consisting of three data fields. The first one uniquely identifies the target cell of the given verification area, the second one consists of all cell identifiers included in the target extension set, and the third one a list of CM parameter values. The latter one consists of either a complete snapshot of all CM settings the target cell has previously had or a partial list that includes only a CM parameter subset. Such partial list is typically created when we rollback CM changes made by SON functions. For instance, in case of the Mobility Robustness Optimization (MRO) function, we may undo only parameters influencing the handover of User Equipments (UEs) between cells.

B. Verification time windows

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. As part of the anomaly detection procedure, the observation window is used for the generation of verification areas as well as the comparison of cell profiles with the current PM reports. The length of a single slot depends on the PM granularity periods [1], i.e., the PM data collection frequency, whereas the total number of required slots depends on the anomaly detection strategy.

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. The length of a correction window slot is reliant on the delay until the impact of those requests becomes visible in the PM data. The total number of available slots, however, is subject to the environment. For instance, 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.

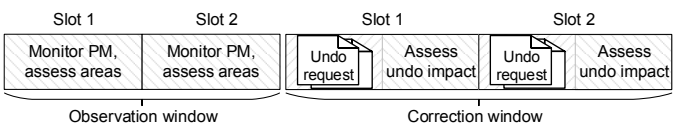


Figure 1. Example of a 2-slot observation and 2-slot correction window

III. Challenges for SON verification

A. Interruption of a SON optimization process

As stated in Section II-B, the observation window is the time frame during which a verification mechanism monitors the cell performance and decides whether an area should be further assessed by the verification process. However, in a SON a performance drop does not immediately mean that CM changes have to rolled back. Today, some SON functions require not only one, but several steps to achieve their goal. In other words, SON functions may try out different CM settings until they finally manage to optimize the network. As they do so, they may induce a temporal performance decrease. For instance, in LTE the CCO function monitors the impact of its last deployed antenna tilt or transmission power change, and corrects that if required [1].

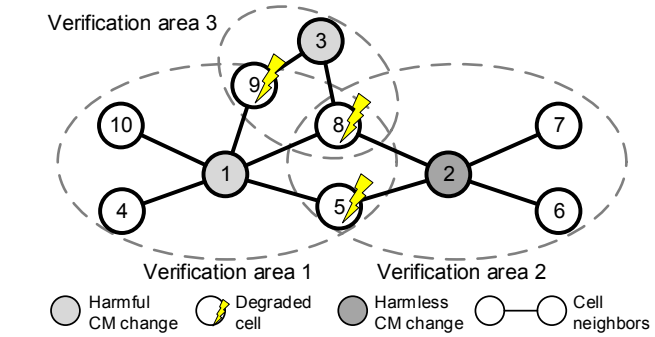
For this reason, a verification mechanism has to be able to filter out such changes. Otherwise, it may interrupt SON functions that are currently optimizing the network and prevent them from reaching their objective.

B. Verification collisions

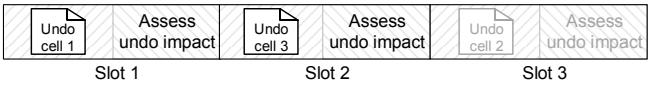
A collision takes place when two or more verification areas share anomalous cells, i.e., the corresponding undo requests address two overlapping sets of cells. If we denote the set of all cells as Σ, the set of all anomalous cells as Σ', where $\Sigma' \subseteq \Sigma$, and the set of all verification areas as Φ, two verification areas $\varphi_i, \varphi_j \in \Phi$ are said to be in a collision if and only if $\varepsilon(\varphi_i) \cap \varepsilon(\varphi_j) \subseteq \Sigma'$. Here, ε is a function that returns the cells of a verification area, i.e., $\varepsilon \colon \Phi \rightarrow \mathcal{P}(\Sigma) \setminus \{\varnothing\}$. As a result, we have an uncertainty which change to rollback and which to accept.

To give an example, let us assume that we have 10 cells, as given in Figure 2(a). Three of those cells (1, 2, and 3), have been reconfigured whereas another three (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 and each in collision with the other. Note that in this example an area includes the first degree neighbors of the target cell and is identified by the ID of that cell.

To eliminate this uncertainty, the verification process needs to create 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 one ensures 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 must be allocated to the first slot. A permissible allocation in case of the aforementioned collisions is given in Figure 2(b). The CM changes that have led to a degradation are assigned to slots 1 and 2, respectively. The rollback of cell 2 is assigned to the third slot, but is later discarded.



(a) Verification area overlaps



(b) Correction window slot allocation

Figure 2. Verification collision example

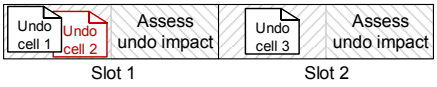


Figure 3. Example of an inefficient correction window slot allocation

C. Insufficient number of correction window slots

The time for deploying undo requests might be rather limited, i.e., the number of correction window time slots might be insufficient. As a result, it might not be always possible to guarantee that the plan is collision-free. Suppose that for the deployment of the undo requests in Figure 2(a) we have a correction window of size two. The question would be how to process the queued requests. One possible strategy that we might follow here is to violate the collision constraint by grouping some undo requests. However, grouping increases the risk of rolling back CM changes that did not harm the network performance. For instance, the undo request for cell 1 and 2 might get allocated to the same time slot, as shown in Figure 3. Hence, we will rollback the settings for cell 2 although it was not responsible for any degradation.

D. False positive collisions

Let us continue with the example from Figure 2(a) and 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 4 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 in Figure 2(a) verification area 3 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

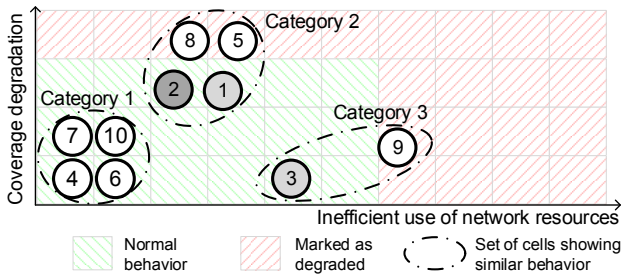
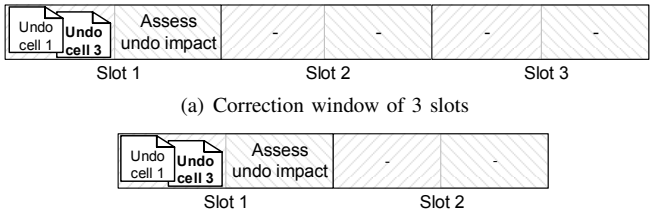


Figure 4. Performance behavior of the cells



(a) Correction window of 3 slots

(b) Correction window of 2 slots

Figure 5. Slot allocation after eliminating false positive collisions

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 false positive collisions between the following area pairs: (1,3) and (2,3). If we omit those collisions, though, we would be allowed to simultaneously deploy the undo requests for cell 1 and 3, as depicted in Figure 5(a). In other words, we can restore the performance of the network much faster.

Furthermore, if we eliminate those false positive collisions, we would lower the probability of getting an allocation like the one presented in Figure 3, where we have an insufficient number of correction window slots. Obviously, if we remove the false positives we can simultaneously rollback the CM settings of cell 1 and 3, as presented in Figure 5(b).

IV. Factors impacting SON verification

A. Verification mechanism location

In order to have a wide view of the network and all ongoing CM changes, a verification mechanism resides at the Domain Management (DM) or the Network Management (NM) level of the 3GPP Operation, Administration and Management (OAM) architecture [1]. However, being at that level hinders the verification mechanism from instantly verifying CM changes and will, therefore, have a coarse-granular view of the CM data. As a result, it will analyze a large set of CM changes at once which can complicate the verification process since it creates a potential for verification collisions to emerge.

Figure 6 shows the impact of this problem in an LTE network. The statistics originate from a real network consisting of 3028 cells [13], in which two SON functions have been actively performing changes: the Automatic Neighbor Relation (ANR) and the Physical Cell Identity (PCI) allocation function [1]. Furthermore, CM data has been gathered only once a day, which resulted in the CM change pattern shown in Figure 6(a). The figure gives us the daily number of changed adjacencies between network cells, that are required for handover of UEs. As it can be seen, a verification mechanism is required to assess seven CM change sets, five of which contain a large number of modifications.

B. PM granularity periods

The frequency of getting PM data from the network, also called a PM granularity period [1], plays a major role while verifying CM changes. As stated in [15], the lower bound is set to 15 minutes which is mainly due to the relative network and processing overhead of getting the data. However, such a long granularity period increases the probability of SON functions to become active at the same time. Thereby, numerous verification areas might be simultaneously generated and processed which increases the likelihood of verification collisions as well as the number of required correction window slots.

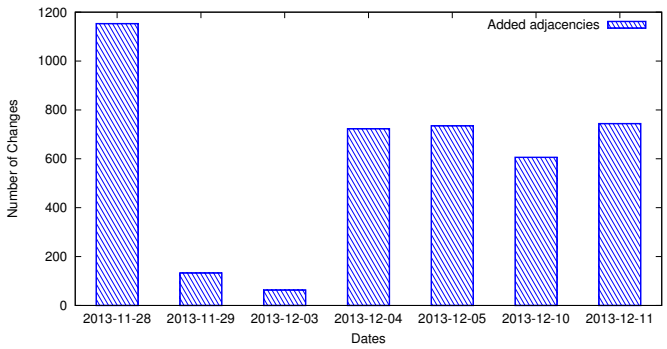
Let us recall the LTE network statistics presented in Section IV-A. PM data was exported every hour, which caused the anomaly pattern from Figure 6(b) to emerge. The figure shows the number of degraded adjacency objects as well as number of time slots we need if we attempt to rollback the changes. As it can be seen, this rather long PM granularity period results in the simultaneous rollback of a relatively large number of changes. Furthermore, the number of required correction window time slots increases as well. As Figures 6(c) and 6(d) outline, the attempt of undoing the changes for the 28^{th} of November requires 18 time slots, whereas for the 4^{th} of December 9 time slots.

C. Collection of statistically relevant PM data

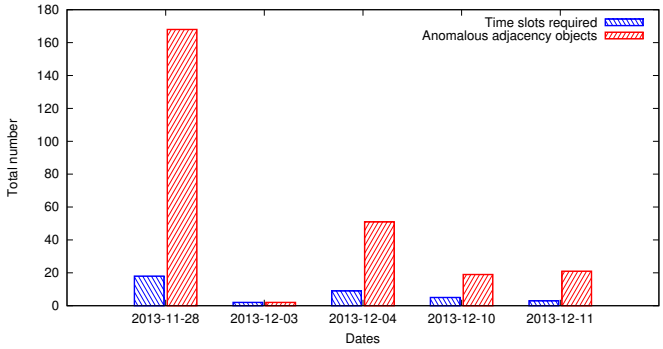
Even if assume that we can frequently get PM data from the network, the question remains of how statistically relevant the gathered data is. A significant change in Key Performance Indicators (KPIs) like the Channel Quality Indicator (CQI) or the Handover Success Rate (HOSR) [1] can be only seen when we have enough UEs that actively use the network, e.g., during peak hours of a weekday. As a result, a verification mechanism is able to assess certain CM changes only when such data is available, which may leads us to the aforementioned problems. Namely, numerous verification areas may simultaneously enter the verification process.

D. Coarse granular reconfigurations

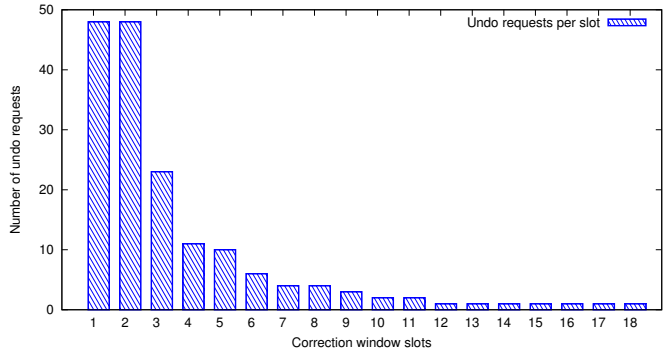
Typically, CM changes are deployed by using reconfiguration modules connected to the antennas. In the case of antenna tilt changes, those modules are referred to as Remote Electrical Tilt (RET) modules. However, more than one antenna can be connected to the same module which makes it impossible to separately correct some parameters for each antenna. As a result, we have shared modules which force multiple changes to be deployed at once. That is, a verification mechanism is required to assess the impact of more changes than initially planned, which leads us to the problems described in Section IV-A. In [16], the authors faced this problem after analyzing the changes suggested by an offline CCO algorithm. In total, 21 cells have been suggested for optimization, but 32 changes were deployed due to the shared RET module.



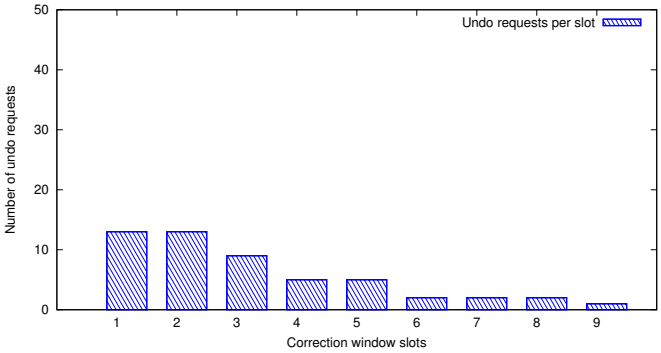
(a) Number of added cell adjacency objects



(b) Anomalous objects and required time slots



(c) Distribution of the undo requests for the 28^{th} of November



(d) Distribution of the undo requests for the 4^{th} of December

Figure 6. Performance analysis an LTE network [14]

E. Neighborship degree

The heterogeneous nature of today's mobile SONs is an essential factor that influences the process of verification. Nowadays, 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. Moreover, the upcoming fifth generation of mobile communications (5G) 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 [17], [18]. These factors create a potential for the aforementioned problems to emerge.

V. Concept

The concept, based on our work from [3], [14], [19], starts with the representation of the performance behavior of each cell in the network. Then, we describe the anomaly detection process which is followed by our solution for eliminating false positive collisions. Finally, we present the method of resolving valid collisions as well as the process of dealing with insufficient correction window slots. Note that the used symbols and notations are summarized in Table I.

A. Cell behavior model

Each cell in the network is represented by an anomaly vector $\mathbf{a} = (a_1, \ldots, a_n)$ that is element of $\mathbb{R}^n$. Each $a_1, \ldots, a_n$ is called a *KPI anomaly level* and its purpose is to represent the deviation of a given KPI from the expected value. The KPIs that are considered while calculating vector $\mathbf{a}$, are referred to as *dedicated KPIs* and their set is denoted as K, where $|K| = n$. Furthermore, for the set of all cells Σ, $A = \{\mathbf{a}_1, \ldots \mathbf{a}_s\}$ is called the anomaly level vector space, where $|A| = |\Sigma|$.

Let us give an example for computing $\mathbf{a} = (a_1, \ldots, a_n)$. For this purpose, assume that K consist of the HOSR and Call Setup Success Rate (CSSR). Here, we may compute each KPI anomaly level as the z-score [20] of a KPI, i.e., the distance between the given KPI value and the sample mean in units of the standard deviation. The samples required to compute that value can be collected separately, e.g., during a training phase. Let us assume that during this phase a cell reports a HOSR of $\{99.9\%, 99.7\%\}$ and a CSSR of $\{99.5\%, 97.4\%\}$. In addition, suppose that the current HOSR and CSSR are 90.2% and 90.0%, respectively. This leads to the following anomaly vector: $\mathbf{a} = (-1.15, -1.13)$, i.e., a HOSR anomaly level of -1.15 and CSSR anomaly level of -1.13. Since both are negative, we can state that the current KPI values are more than one standard deviation below the sample mean.

B. Detecting anomalies

Let us continue with the z-score based anomaly vector presented above. Since it consists of z-scores indicating that the current HOSR and CSSR more than one standard deviation away from the sample mean, we may consider the given cell as being anomalous. As a result, we will generate a verification area, that includes the target cell as well as cells surrounding it, and push it through the verification process. However, this area might be unnecessarily processed since we might have a temporal performance decrease induced by a SON function.

Table I
NOTATION OVERVIEW

Sets	
Σ	Set of all cells
Σ', where $\Sigma' \subseteq \Sigma$	Set of all anomalous cells
Φ	Set of all verification areas
X	Set of verification area variables
K	Set of dedicated KPIs
Θ	Set of collisions in a clique group
Vectors	
$\mathbf{a} = (a_1, \ldots, a_n)$	Anomaly vector
$a_1, \ldots, a_n$, where $n = \lvert K \rvert$	KPIs anomaly levels
$A = \{\mathbf{a}_1, \ldots, \mathbf{a}_s\}$, $\lvert A \rvert = \lvert \Sigma \rvert$	Anomaly level vector space
$\zeta_t \in \mathbb{R}$	CVSI at time t
Graphs	
$G^\Phi = (V^\Phi, E^\Phi)$	Verification collision graph
C^Φ, where $C^\Phi \subseteq V^\Phi$	A clique of the graph G^Φ
$G^\Sigma = (V^\Sigma, E^\Sigma)$	Cell behavior graph
$T^\Sigma = (V^\Sigma, E_t^\Sigma)$	Cell behavior tree
$F^\Sigma = \{T_1^\Sigma, \ldots, T_\kappa^\Sigma\}$	Set of disjoint trees (cell clusters)
$w(v_i^\Sigma, v_j^\Sigma)$	Weight of an edge $(v_i^\Sigma, v_j^\Sigma) \in E^\Sigma$
Functions	
$\varepsilon \colon \Phi \to \mathcal{P}(\Sigma) \setminus \{\varnothing\}$	Cell extraction function
$\psi \colon A \to \mathbb{R}$	Vector aggregation function
$\omega \colon V^\Phi \to X$	Variable assignment function
$\rho \colon V^\Phi \to \mathbb{R}$	Verification area priority function
$\rho' \colon \Theta \to \mathbb{R}$	Collision priority function
$\delta \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$	Cell distance function
$\xi \colon T^\Sigma \to \mathbb{R}$	Edge removal function
$r \colon X \times X \to \{0,1\}$	Reification function
Counters	
$\tau \in \mathbb{N}^+$	Available correction window slots
$t \in \mathbb{N}^+$	Time, granularity period
$\kappa \in \mathbb{N}^+$	Number of cell clusters

As discussed in Section III-A, this might happen as it tries to reach its optimization goal.

Hence, to make the detection approach more resistant against PM fluctuations, we define for each a cell a Cell Verification State Indicator (CVSI). Let us denote that indicator as ζ, where $\zeta \in \mathbb{R}$. The computation of ζ is done by applying exponential smoothing, as follows:

$$\zeta_0 = \psi(\mathbf{a})_0 \tag{1}$$

$$\zeta_t = \alpha\psi(\mathbf{a})_t + (1-\alpha)\zeta_{t-1},\ t > 0 \tag{2}$$

Here, $\alpha \in [0;1]$ is the smoothing factor, also referred to as the *state update factor*, whereas ζ_t the CVSI calculated at time t. The CVSI is a simple weighted average of the current observation, denoted as $\psi(\mathbf{a})_t$, and the previous smoothed ζ_{t-1}. Note that the current observation is an aggregation of $\mathbf{a} = (a_1, \ldots, a_n)$, i.e., $\psi \colon A \to \mathbb{R}$. For instance, $\psi(\mathbf{a})$ can be the arithmetic average of $a_1, \ldots, a_n$ or the norm of $\mathbf{a}$.

Figure 7 represents an exemplary computation of ζ over a time frame of 30 PM granularity periods. Here, $\psi(\mathbf{a})$ corresponds to the arithmetic average of $a_1, \ldots, a_n$, as defined in the example in Section V-A. As shown, the PM fluctuations

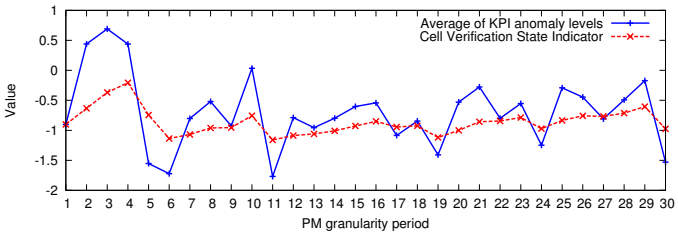


Figure 7. Exemplary computation of the CVSI ζ

have been smoothed which gives us a more realistic view on how the cell performs.

C. Eliminating false positive verification collisions

Let us go one step further and assume that rather than having just one cell, we have a mobile network consisting of 12 cells and 18 cell adjacencies, as shown in Figure 8. Now, suppose that after accessing the CM and PM database, we are able to spot the following CM changes and anomalies: cells 1, 2, 6, 8, and 11 have been reconfigured, and cells 3, 4, 7, and 10 have degraded. Based on those events, let us construct a verification area by taking the target cell as well as its direct neighbors. Furthermore, let us identify a verification area by the ID of the target cell. As Figure 9(a) outlines, we have the following four verification collisions: (1,2), (2,6), (6,8), and (8,11). The question that we aim at answering is how many of those collisions are really required and whether there are undo requests that are unnecessarily held back.

In Section V-A, we represented the behavior of a cell as a vector $\mathbf{a} = (a_1, \ldots, a_n)$. Now, we are going to group cells that behave similarly. For this purpose, we define a function δ (Equation 3), that computes $\forall i, j$ the distance between $\mathbf{a}_i$ and $\mathbf{a}_j$. A popular example is the Manhattan or the Euclidean distance function [21].

$$\delta \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \tag{3}$$

Next, we construct an undirected edge-weighted graph $G^\Sigma = (V^\Sigma, E^\Sigma)$, where V^Σ is a set of vertexes and E^Σ a set of edges $V^\Sigma \times V^\Sigma$. Each vertex $v_i^\Sigma \in V^\Sigma$ represents a cell in the $\mathbb{R}^n$ space by taking $\mathbf{a}_i$ into account. In addition, the weight of every edge $(v_i^\Sigma, v_j^\Sigma) \in E^\Sigma$, denoted as $w(v_i^\Sigma, v_j^\Sigma)$, is computed by applying δ.

An exemplary composition of G^Σ in the $\mathbb{R}^2$ space is given in Figure 9(b). Note that for simplicity reasons, some vertexes are completely overlapping, i.e., they represent cells showing exactly the same behavior. Hence, the distance between those vertexes is zero.

After constructing G^Σ, we form an MST by searching for an undirected subgraph $T^\Sigma = (V^\Sigma, E_t^\Sigma)$, where $E_t^\Sigma \subseteq E^\Sigma$, that

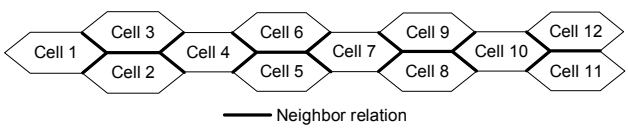
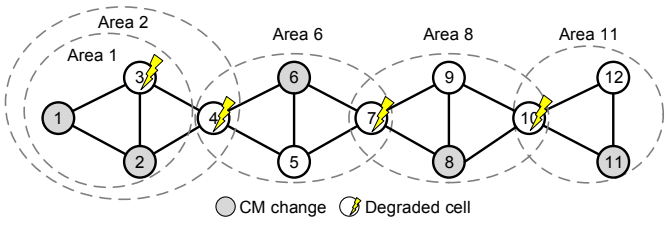
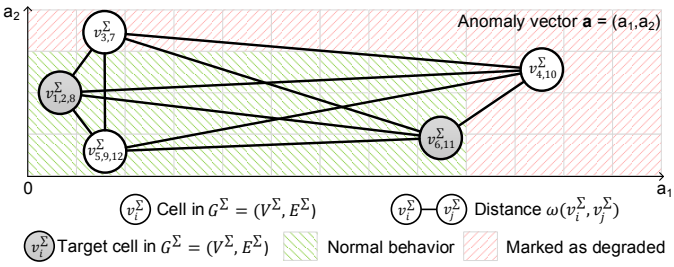


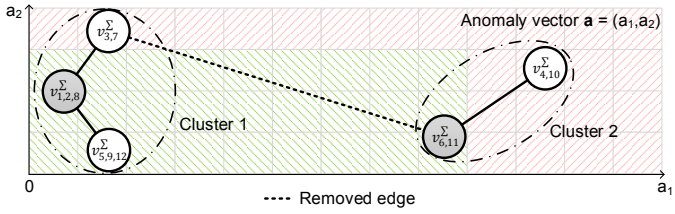
Figure 8. Example of a mobile network



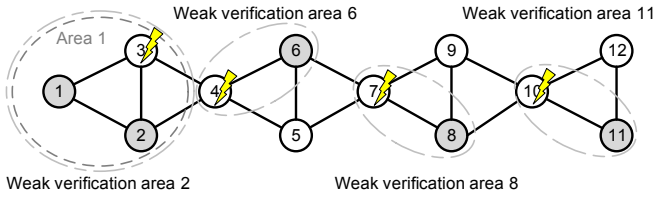
(a) Verification area selection



(b) Behavior graph construction



(c) MST construction and clustering



(d) Weak verification area selection

Figure 9. Example of eliminating false positive collisions

minimizes $\sum_{v_i^\Sigma, v_j^\Sigma \in G^\Sigma} w(v_i^\Sigma, v_j^\Sigma)$. 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 [22].

Then, 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^Σ, i.e., an undirected graph whose connected components are trees, denoted as $\{T^{\Sigma\prime}_1, \ldots, T^{\Sigma\prime}_\kappa\}$, each being disjoint with the other. That is, we have to satisfy the following condition: $\forall i, j : V^{\Sigma\prime}_i \cap V^{\Sigma\prime}_j = \{\varnothing\}$, where $V^{\Sigma\prime}_i$ and $V^{\Sigma\prime}_j$ are the vertex sets of $T^{\Sigma\prime}_i$ and $T^{\Sigma\prime}_j$, respectively. This procedure, also known as MST clustering [23], is carried out by removing a certain number of edges from T^Σ 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.

Figure 9(c) outlines an exemplary MST. It consists of five vertexes and the following edges: $(v^\Sigma_{1,2,8}, v^\Sigma_{5,9,12})$, $(v^\Sigma_{1,2,8}, v^\Sigma_{3,7})$, $(v^\Sigma_{6,11}, v^\Sigma_{4,10})$, and $(v^\Sigma_{3,7}, v^\Sigma_{6,11})$. The removal of the latter edge leads to the formation of two clusters, namely $(v^\Sigma_{1,2,8}, v^\Sigma_{5,9,12}, v^\Sigma_{3,7})$ and $(v^\Sigma_{4,10}, v^\Sigma_{6,11})$.

The decision itself when to remove an edge from T^Σ depends on an edge removal threshold. The latter one is calculated by an edge removal function, as defined in Equation 4.

$$\xi : T^\Sigma \rightarrow \mathbb{R} \tag{4}$$

Depending on how many edges we have removed, we will face one those three outcomes: (1) $\kappa = 1$, (2) $\kappa = |V^\Sigma|$, and (3) $\kappa = [2, |V^\Sigma|)$. In the first case, the forest F^Σ 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^Σ has $|V^\Sigma|$ trees each consisting of one vertex, i.e., each cell is in a different cluster. Consequently, all collisions are marked as false positive 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 positive and are eliminated.

Figure 9(d) gives us the end result this procedure. 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, while the remaining ones are removed.

D. Solving the verification collision problem

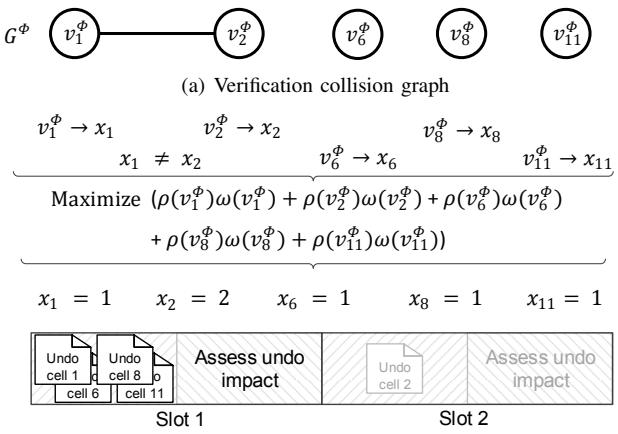
Let us continue with the example from Figure 9(d). As shown, the undo requests for cell 1 and 2 cannot be simultaneously deployed, i.e., we need to find out which of those two requests are most likely responsible for the degradation of cell 3.

To do so, we represent the verification collision problem as an undirected graph $G^\Phi = (V^\Phi, E^\Phi)$, consisting of a set V^Φ of vertexes and a set E^Φ of edges. The set V^Φ represents the abnormally performing verification areas, whereas E^Φ the set of verification collisions $V^\Phi \times V^\Phi$. That is, two vertexes are connected with each other if and only if the corresponding undo requests must not be simultaneously deployed due to shared anomalous cells. Figure 10(a) visualizes G^Φ for the given collisions. It consists of the following set of vertexes and edges: $V^\Phi = \{v^\Phi_1, v^\Phi_2, v^\Phi_6, v^\Phi_8 v^\Phi_{11}\}$ and $E^\Phi = \{(v^\Phi_1, v^\Phi_2)\}$.

As a next step, we define an assignment function (Equation 5) that maps each vertex $v^\Phi_i \in V^\Phi$ to a variable $x_i \in X$. Furthermore, each x_i takes values between 1 and the number of available correction window slots τ. Basically, what we say here is that the undo request for the target cell of the given verification area can be assigned to one of the available slots.

$$\omega : V^\Phi \rightarrow X \tag{5}$$

After carrying this assignment out, we prioritize the undo requests for the given verification areas. To do so, we define a priority function ρ, as given in Equation 6. The outcome should be interpreted in the following way: the lower the value of x_i, the more important it is to process v^Φ_i, i.e., to execute the undo request for the area associated with v^Φ_i. The priority



(a) Verification collision graph

(b) Correction window slot allocation

Figure 10. Example of resolving verification collisions

itself may depend on factors like the geographical location of the target cell, number of degraded cells within an area or the overall degradation level.

$$\rho \colon V^\Phi \to \mathbb{R} \tag{6}$$

Based on those two functions, we define the undo request deployment as a constraint optimization problem as follows:

$$\max \sum_{v^\Phi \in V^\Phi} \rho(v^\Phi)\omega(v^\Phi) \tag{7}$$

subject to

$$\forall (v_1^\Phi, v_2^\Phi) \in E^\Phi : \omega(v_1^\Phi) \neq \omega(v_2^\Phi). \tag{8}$$

In Equation 8 we say that the variables of every two vertexes connected in G^Φ must not receive the same value. Therefore, the set of edges E^Φ can be also seen as a set of hard constraints [24] that must not be violated. The objective itself is defined in Equation 7, which is a maximization of the weighted sum. The undo requests of the area associated with the variable receiving the value 1 are deployed at first.

Figure 10(b) visualizes an exemplary outcome of this process. For the given G^Φ, we have the following variables: x_1, x_1 x_2, x_6 x_8, and x_{11}. In addition, we have only one constraint, namely $x_1 \neq x_2$. Each variable x_i is multiplied with the outcome of $\rho(v_i^\Phi)$ and the resulting weighted sum is maximized. One permissible outcome is the allocation of the undo request for cell 1, 6, 8 and 11 to the first time slot.

Finally, the impact of those requests on the network performance is assessed. Should the network still show an anomalous behavior, the whole process is triggered again. However, with one notable difference, namely a decreased τ. The reason is that we have already consumed one time slot after deploying the first set of undo requests, i.e., we have one less for completing the verification process.

E. Solving an over-constrained verification problem

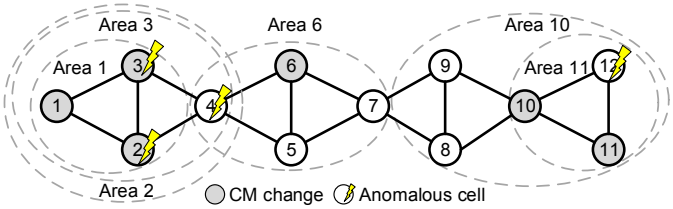
Sections V-A to V-D presented the cell behavior model, described how to prevent verification areas from being unnecessarily processed, how to eliminate false positive collisions as well as how to resolve those being considered as valid. However, providing a solution to the verification collision problem might be challenging, as shown by the following example. Suppose that our network from Figure 8 shows an activity like the one presented in Figure 11(a): cells 1, 2, 3, 6, 10 and 11 have been reconfigured, and cells 2, 3, 4, and 12 have degraded. As a result, we have the following verification collision pairs: (1,2), (1,3), (2,3), (2,6), (3,6), and (10,11). Now, assume that all collisions are valid and that τ, i.e., the number of available correction window slots, is two. Obviously, we have an *over-constrained verification collision problem*, since we cannot find a slot allocation that satisfies all given constraints. Consequently, we need to modify our verification collision resolving approach in such a way that it is able to find an *acceptable solution*, i.e., an outcome that minimizes the total constraint violation.

To find such a solution, we have to firstly identify the vertexes in the verification collision graph G^Φ that make our problem unsolvable. Cliques from graph theory [22] are able to provide us with that information. A clique C^Φ is a subset of the vertexes of the graph G^Φ (i.e., $C^\Phi \subseteq V^\Phi$) such that every two distinct vertexes $v_i^\Phi, v_j^\Phi \in C^\Phi$ are adjacent in G^Φ, i.e., $(v_i^\Phi, v_j^\Phi) \in E^\Phi$. That is, a clique induces a complete subgraph of G^Φ which consequently means that all areas associated with the vertexes within a clique are in a collision with each other. However, we are interested only in those cliques that have a certain size, namely such that have more than τ vertexes upon all maximal cliques. A maximal clique is a clique that cannot be extended by adding more adjacent vertexes. Consequently, each clique having more than τ vertexes represents an over-constrained verification collision problem.

Figure 11(b) visualizes the outcome of this procedure. The given verification collision graph G^Φ consists of the following vertexes and edges: $V^\Phi = \{v_1^\Phi, v_2^\Phi, v_3^\Phi, v_6^\Phi, v_{10}^\Phi, v_{11}^\Phi\}$ and $E^\Phi = \{(v_1^\Phi, v_2^\Phi), (v_1^\Phi, v_3^\Phi), (v_2^\Phi, v_3^\Phi), (v_2^\Phi, v_6^\Phi), (v_3^\Phi, v_6^\Phi), (v_{10}^\Phi, v_{11}^\Phi)\}$. Due to the limitation of $\tau = 2$, the maximal cliques that we are interested in must contain at least 3 vertexes. Here, two are meeting this requirement: one comprised of v_1^Φ, v_2^Φ, v_3^Φ, and another including v_2^Φ, v_3^Φ, v_6^Φ.

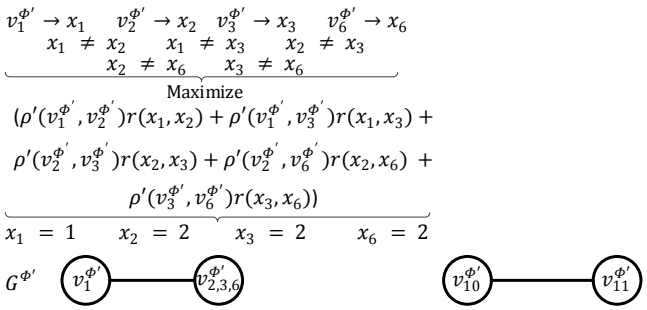
Let us denote the set of vertexes that are part of at least one clique as $V^{\Phi\prime}$, where $V^{\Phi\prime} \subseteq V^\Phi$. Considering the above-mentioned example, this set would include v_1^Φ, v_2^Φ, v_3^Φ, v_6^Φ. Now, we need to determine whether those vertexes are part of one single over-constrained problem, or whether they are part of two or more independent problems. In other words, we must find a *partition* of $V^{\Phi\prime}$, i.e., a set P of sets that does not contain the empty set, the union of all sets in P equals $V^{\Phi\prime}$, and the intersection of any two sets in P is empty. Obviously, cliques do not give us this partitioning since they may have common vertexes. However, if we unite cliques that are sharing vertexes and consider the resulting unions as one entity, we will be able to get P. In this paper, we call the union of such cliques a *clique group*. In our example, we will get the partition $\{\{v_1^\Phi, v_2^\Phi, v_3^\Phi, v_6^\Phi\}\}$, i.e., we have exactly one clique group combining the two cliques.

Next, are going to find an acceptable solution for each clique group by violating some collision constraints. To do so, we



(a) Verification area selection

(b) Verification collision graph and clique selection



(c) Reducing oversized cliques

Figure 11. Example of an over-constrained verification collision problem

reduce a clique group's size by merging some of its vertexes. This process is also known as edge contraction, i.e., for every two merged vertexes, an edge $e^{\Phi} \in E^{\Phi}$ is removed and its two incident vertexes $v_i^{\Phi}, v_j^{\Phi} \in V^{\Phi}$, are merged into a new vertex $v^{\Phi'}_k$, where the edges incident to $v^{\Phi'}_k$ each correspond to an edge incident to either v_i^{Φ} or v_j^{Φ}. This procedure gives us a new undirected graph $G^{\Phi'} = (V^{\Phi'}, E^{\Phi'})$ that does not have the edges between merged vertexes.

To find those edges, we assign each vertex within a clique group a variable x_i ranging between 1 and τ. Let us reuse ω from Section V-D for that purpose. Furthermore, let us denote all collisions $V^{\Phi} \times V^{\Phi}$ within a clique group as Θ. Those collisions are then marked as *soft constraints*, which as known from constraint optimization, are such that might be violated based on the priority they have received [24]. To compute it, we define a modified priority function ρ', as follows:

$$\rho' \colon \Theta \to \mathbb{R} \tag{9}$$

In contrast to Section V-D, it rates the verification collision instead of the verification areas. Moreover, the outcome should be interpreted as follows: the higher the value, the more important it is the constraint to be satisfied, i.e., the collision to remain.

Finally, the clique group size reduction is modeled as a constraint optimization problem defined in the following manner:

$$\max \sum_{(v_i^{\Phi}, v_j^{\Phi}) \in \Theta} \rho'(v_i^{\Phi}, v_j^{\Phi}) r(\omega(v_i^{\Phi}), \omega(v_j^{\Phi})) \tag{10}$$

subject to

$$r(x_i, x_j) = \begin{cases} 0 & \text{iff } x_i = x_j \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

As known from constraint optimization, Equation 11 gives us a reification function r that tells us whether the inequality of two variables x_i and x_j is satisfied. Equation 10 defines the objective, i.e., the maximization of the weighted sum of all soft constraints. Finally, the vertexes whose variables receive the same value are merged together.

In Figure 11(c), each vertex of the clique group is assigned to a variable x_i. In addition, the verification collisions lead us to the following constraints: $x_1 \neq x_2$, $x_1 \neq x_3$, $x_2 \neq x_3$, $x_2 \neq x_6$, and $x_3 \neq x_6$. One permissible outcome is the merge of v_2^{Φ}, v_3^{Φ}, and v_6^{Φ}, since their variables got the same value. The resulting graph $G^{\Phi'}$ consists of four vertexes $v^{\Phi'}{}_1$, $v^{\Phi'}{}_{2,3,6}$, $v^{\Phi'}{}_{10}$, and $v^{\Phi'}{}_{11}$, as well as the edges $(v^{\Phi'}{}_1, v^{\Phi'}{}_{2,3,6})$ and $(v^{\Phi'}{}_{10}, v^{\Phi'}{}_{11})$.

After completing those steps, we no longer have over-constrained verification collision problems, i.e., we can continue with the approach introduced in Section V-D.

VI. Environment

A. LTE network simulator

The used LTE radio network simulator is part of the SON simulator/emulator suite, as defined in [25]. The configuration parameters are summarized in Table II. The simulator performs continuous simulation by tracking the changes in the network over time. The time itself is divided into time slices, called simulation rounds. Those rounds correspond to the PM granularity periods, as explained in Section IV-B. At the beginning of a round, the simulator configures the network as defined by the current CM parameter settings. The set of parameters that are changed and verified during the test runs are the antenna tilt, transmission power, and CIO. The SON functions that are allowed to adjust them are the MRO, RET, and the Transmission Power (TXP) function [1]. Note that the latter two are a special type of the CCO function.

During a simulation round, 1500 uniformly distributed users follow a random walk mobility model (6 km/h) and actively use the mobile network. The simulated scenario itself covers parts of Helsinki, Finland. At the end of a round, PM data is exported for each cell, which is evaluated by the SON functions as well as the verification mechanism.

B. Verification process configuration

1) Dedicated KPIs: The dedicated KPIs are the HOSR, the handover ping-pong rate, and the CQI [1]. Note that the latter one is calculated as the weighted harmonic mean of the CQI channel efficiency. The efficiency values are listed in [27].

2) Anomaly vector: To compute a KPI anomaly level, i.e., an element a_k of an anomaly vector $\mathbf{a} = (a_1, \ldots, a_n)$, we take the z-score computation as introduced in Section V-A. Furthermore, we distinguish between success KPIs (e.g., HOSR) and failure KPIs (e.g., handover ping-pong rate, call drop rate). Since, a negative z-score is an indication for a drop in performance only in the case of success KPIs, we negate the z-score of failure KPIs in order to keep things consistent.

Table II
LTE NETWORK SIMULATOR SETUP

Parameter	Value
Network Frequency	2000 MHz
Bandwidth	20 MHz
Number of PRBs	100
Handover hysteresis threshold	2.0 dB
Radio Link Failure threshold	-6.0 dB.
Shannon gap	-1.0 dBm
Thermal noise	-114.447 dBm
Path loss model	UMTS 30.03 [26]
Downlink scheduler mode	CBR mode
Shadowing correlation distance	50.0 m
Shadowing standard deviation	8.0 dBm
Total cells	32 macro cells
Antenna heights	17-20 m
Simulation round	5400 sec
Simulated area	50 km^2
Number of users	1500 uniform
User speed	6 km/h
User movement	Random walk
Constant bit rate requirement	175 kbps
SON functions	MRO, TXP, RET [1]

3) Verification area selection: A verification area is set to include the target cell and its direct neighbors to which UEs can be handed over.

4) CVSI calculation: The observation $\psi(\mathbf{a})_t$ at time t is computed as follows:

$$\psi(\mathbf{a}) := \frac{1}{n}\sum_{i=1}^{n} a_i \tag{12}$$

Furthermore, the state update factor α is selected at time t as follows:

- $\alpha = 0.2$ if $|\psi(\mathbf{a})_t| \in [0;1)$
- $\alpha = 0.4$ if $|\psi(\mathbf{a})_t| \in [1;2)$
- $\alpha = 0.8$ if $|\psi(\mathbf{a})_t| \in [2;\infty)$

A verification area is considered as being anomalous at time t, if and only if the following condition is met:

$$\frac{1}{|\varepsilon(\varphi)|}\sum_{i=1}^{|\varepsilon(\varphi)|} \zeta_t(\sigma_i) \in [2.0,\infty) \tag{13}$$

Note that ε is a function that gives us the cells of a verification area $\varphi \in \Phi$, as defined in Section III-B.

5) Cell distance function: 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 14.

$$\delta(\mathbf{a},\mathbf{a'}) = \delta(\mathbf{a'},\mathbf{a}) = \sqrt{\sum_{k=1}^{n}(a'_k - a_k)^2} \tag{14}$$

Hence, T^{Σ} from Section V-C is an Euclidean MST, in which each edge weight $w(v_i^{\Sigma}, v_j^{\Sigma})$ equals the Euclidean distance between $v_i^{\Sigma} \in V^{\Sigma}$ and $v_j^{\Sigma} \in V^{\Sigma}$.

6) Edge removal function: To form F^{Σ}, we remove all edges from T^{Σ} whose weight exceeds 1.5 (first configuration) or 1.75 (second configuration) times the average edge weight.

7) Priority functions: The functions ρ and ρ' compute the priority by taking the degradation severity of the considered cells into account.

VII. EVALUATION

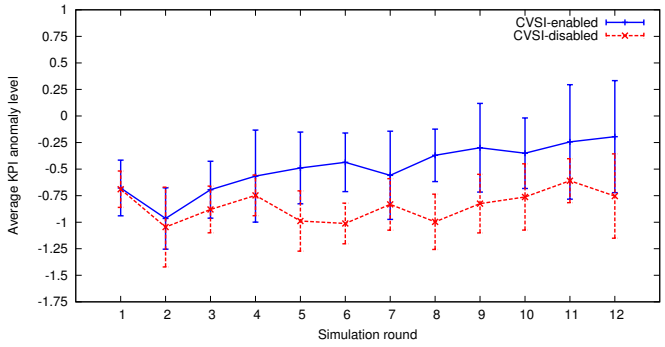
The evaluation consists of three parts and is partially based on our work from [3], [14], [19]. It has been carried out by using the simulation environment, as introduced in Section VI. First, our goal is to study the ability of our verification approach to deal with fluctuations in the PM data. Second, we are interested in the capability of detecting and eliminating false positive collisions. Third, we observe how our solution behaves when we have an over-constrained verification problem due to an insufficient number of correction window slots. Note that all figures depict the 95% confidence intervals around the sample mean.

A. Fluctuating PM data

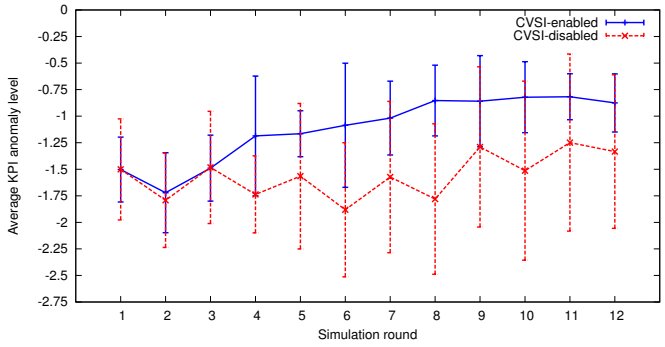
As stated in [1], the environment may change from the assumptions made when the network was initially planned and set up. Typical changes are the construction or demolition of buildings, insertion and deletion of base stations, as well as season changes. As a result, we need to trigger a function like RET or TXP to optimize the coverage. However, starting the optimization process with inaccurate assumptions may cause the SON functions to frequently try out different CM settings, which may also lead to fluctuating PM data.

To recreate this scenario, we have selected nine cells and changed their coverage by using obsolete data. In particular, four cells have a tilt degree of 0.0, two cells a degree of 1.0, one cell a degree of 3.0, and two cells a degree of -4.0. In addition, their transmission power is set to the maximum of 46 dBm. Those obsolete settings are applied before starting a test run. In total, we have seven test runs, during each of which we let the SON functions try reaching their optimization goal. In particular, the coverage optimization functions are changing the physical cell borders at first, which is later followed by an MRO adjustment of the handover parameters. Those functions, however, can be interrupted by the verification mechanism if it decides to generate an undo request, i.e., a rollback has the highest priority. Furthermore, a test run lasts 12 rounds and the verification mechanism is allowed to observe the performance impact of the changes after the third simulation round.

During the test runs, we managed to spot two cells (having the ID 2 and 6) on which the SON functions put their highest focus on. Our observations show that the SON functions tried different CM settings out. In particular, the RET function started to adjust the antenna tilt, which was followed by a CIO adjustment by MRO. Those changes, however, induced temporal performance drops which negatively impacted the verification process while the CVSI feature was turned off. In Figure 12, we show the resulting performance of the areas by calculating the average of all KPI anomaly levels. Note that the verification area of target cell 2 includes five cells whereas the other one six cells. As we can see, the average KPI anomaly level of both areas starts to fluctuate (until round 12) and never reaches zero when the CVSI was not used.



(a) Verification area of target cell 2



(b) Verification area of target cell 6

Figure 12. Average KPI anomaly level when using the CVSI feature

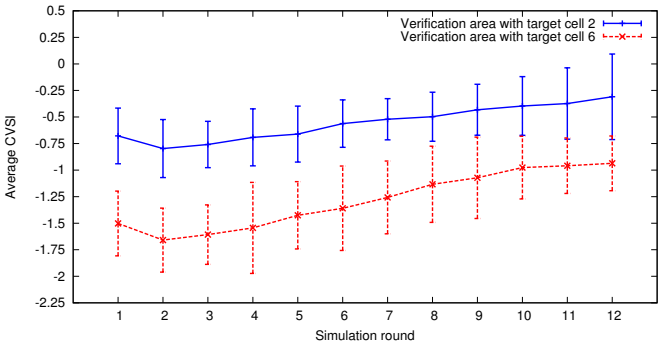


Figure 13. CVSIs of the verification areas

In contrast, having an enabled CVSI gives us a better KPI anomaly level. The reason why it manages to perform like that is given in Figure 13. After simulation round 4, the CVSI approach is able to detect that the areas have stabilized, which allows the SON functions reach their optimization goal.

B. Elimination of false positive collisions

In order to evaluate the capabilities to detect and eliminate false positive collisions, we have to change the experimental setup. We set a test run to last 5 simulation rounds, during which all SON functions are allowed perform changes. At the beginning of a test run certain cells are selected for degradation. The number 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 CIO of −5.0. Moreover, we compute the overall network performance by averaging all KPI anomaly levels reported by all cells after processing the first correction window time slot.

The results of this study are shown in Figure 14(a). 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. In addition, 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 14(b). It shows the number of undo requests remain due to a collision after processing the first time slot. As shown, we are able to allocate more requests to the first slot after removing the false positives and, therefore, have less areas entering the verification process afterwards. However, if we disable the verification collision reduction approach, we are not able to see this improvement due to the large number of collisions preventing the simultaneous deployment of some undo requests.

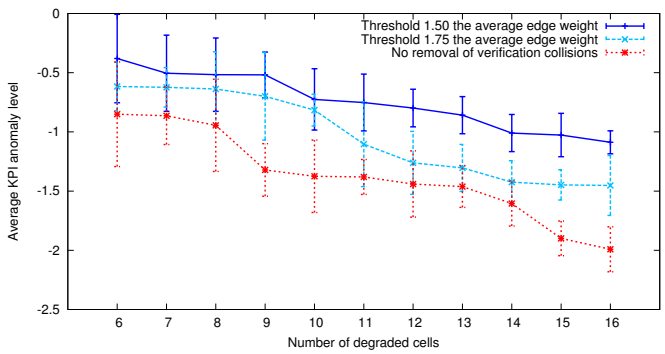
C. Solving an over-constrained verification problem

In order to observe the ability of our approach to handle an over-constrained verification problem, we set the number of available correction windows slots τ to 2. Similarly to the previous experiment, we select a certain number of cells for degradation. This time, the number of degraded cells ranges between 4 and 12. The degradation itself is done by deploying another untypical configuration: an antenna tilt of 3 degrees and a transmission power of 42 dBm. Furthermore, a single test run lasts 4 simulation rounds. During the first round we trigger the degradation, during the second and third one we deploy the undo requests. The result is obtained by computing the average of the KPI anomaly levels reported by all 32 cells from the last simulation round. Furthermore, we compare the proposed constraint softening strategy with a method that uses minimum graph coloring [28], i.e., a constraint satisfaction approach that does not perform any constraint softening [29].

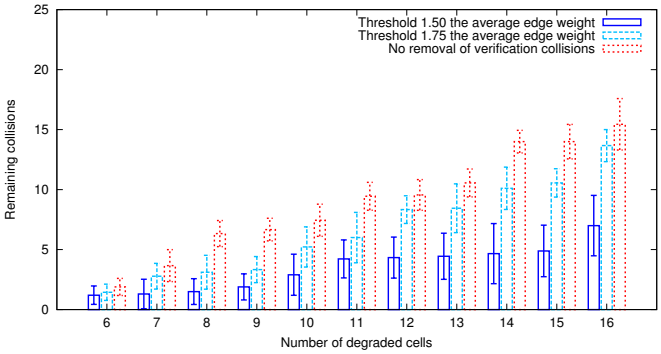
Figure 15(a) shows the outcome of this experiment. As outlined, the constraint softening approach is able to provide a better KPI anomaly level compared the minimum graph coloring strategy. Moreover, we see that in case of nine or more degraded cells, the set maximum of two time slots becomes the limiting factor for the minimum graph coloring method. This trend can be seen in Figure 15(b) which gives us the actual number of deployed undo requests. Our observations show that more verification collisions start appearing as we increase the number of degraded cells which, as a consequence, forces the constraint softening approach to start merging some undo requests. The minimum graph coloring approach on the other side does not perform any grouping, which leads the number of deployed undo requests to decrease.

D. Evaluation remarks and summary

First of all, during the clustering procedure function ξ has to be used with caution since the removal of too many edges from T^{Σ} may lead us to the point where we have no verification

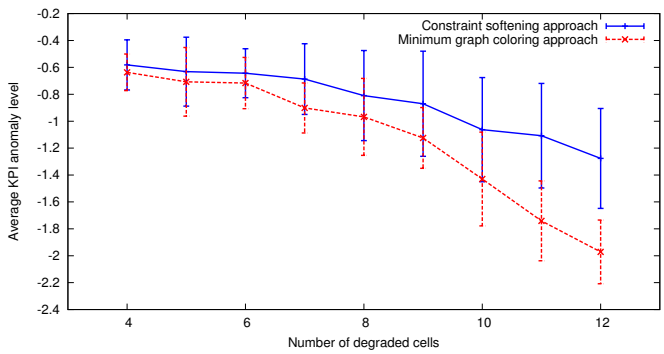


(a) Average KPI anomaly level

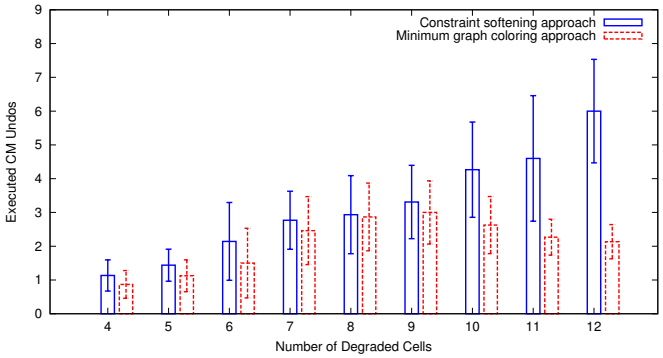


(b) Number of remaining collisions

Figure 14. Evaluation of the capability to eliminate false positive verification collisions [14]



(a) Average KPI anomaly level



(b) Number of excuted undo requests

Figure 15. Evaluation of the capability to solve an over-constrained verification problem [3]

collisions at all. As a result, the verification mechanism will start rolling back too many changes, including such that did not harm the network performance. In the experiment from Section VII-B, we saw this effect as we selected a threshold below 1.5 times the average edge weight.

Second, when verifying CM changes, the network size as well as the availability of the cells has to be taken into account. Today's SONs are not only highly interconnected, i.e., having a high number of cells and cell adjacencies, but may also be optimized for energy saving, i.e., depending on the network requirements numerous cells can be switched on or off. However, the presence of such cells creates uncertainties during the process of verification which may lead to a suboptimal sequence of undo actions or even blaming changes not harming performance. For example, switching on a cell may cause an anomaly at its neighbors since they did not expect this to happen. Similarly, the same may happen when we turn off a cell. The neighbors may expect that the cell is always available in the network. As a result, a high cell neighborship degree, accompanied by dynamic topology changes can be handled only if we manage to update the profiles accordingly.

VIII. Related Work

The concept of pre-action SON coordination [6], [30] can be seen as an alternative strategy to SON verification. It is seen as a pessimistic approach that specifies rules required for the detection and resolution of *known conflicts* between active SON function instances. That is, it prevents conflicting functions from getting active, rather than assessing the network performance after deploying CM changes. For instance, a conflict occurs when SON functions operate on shared CM parameters within the same physical area, or when the activity of one function affects the input measurements of another one.

The basic idea of verifying a CM change triggered by a SON function and rolling it back has been introduced in [15]. However, the presented solution is based only on rules defined with regards to SON coordination. In particular, the priorities of SON function instances are taken into account, i.e., a CM change made by a SON function is rolled back only if another, higher prioritized and conflicting function becomes active within the same area and at the same time.

Within the SOCRATES project [31] ideas have been developed about how undesired behavior can be detected and resolved in a SON. The authors introduce a Guard function whose purpose is to detect unexpected and undesired network performance. In general, they distinguish two types of undesired behavior: oscillations and unexpected absolute performance. Into the first category fall CM parameter oscillations, whereas the second one includes unexpected KPI combinations, like a high Random Access Channel (RACH) rate while having low traffic. The Guard function itself follows only the directive that has been defined through policies. In case of an undesired network behavior, it makes use of two helper functions: an arbitration and an activation function. The

first one is responsible for the detection and resolution of conflicting CM change requests. The second one takes care for enforcing parameter changes, rolling them back, and even suggesting SON function parameter changes. Despite the given ideas, neither a concept is provided, nor a solution is given.

An example for resolving conflicting actions has been introduced in [32]. In particular, the authors focus on how to guarantee a confusion and conflict free PCI assignment. The first property guarantees that there is no cell in the network that has two or more neighbors with identical PCIs, whereas the second one ensures that there are no two neighboring cells receiving the same PCI. The proposed solution makes use of graph coloring, where the vertexes represent the cells in the network and the set of colors the number of available PCI values. In addition, for any two neighboring cells an edge is added to the graph which fulfills the collision free requirement. In order to satisfy the confusion free requirement, an edge is added for every two neighboring cells of second degree.

In [33], an anomaly detection technique for cellular networks has been introduced. It is based on the incremental clustering algorithm GNG which partitions the input data into smaller groups. Those groups represent sets of input data that have similar characteristics. The presented method is referred to as Fixed Resolution GNG (FRGNG) and targets the problems of representing the input data as well as determining when to stop sampling PM data. However, the solution does not address problems like resolving verification collisions, eliminating false positives, as well as solving an over-constrained collision problem.

In [4], an anomaly detection and diagnosis framework has been proposed. Its purpose is to verify the effect of CM changes on the network by monitoring the PM data. The framework itself operates in two phases. First, it detects anomalies by using topic modeling. Second, it performs diagnosis for the found anomalies by using Markov Logic Networks (MLNs). In the latter case it makes use of probabilistic rules to differentiate between different causes. Nevertheless, the presented solution does not address any of the verification collisions problems discussed in Sections III-B to III-D.

In [9] another anomaly detection and diagnosis framework for mobile communication systems is proposed. It analyses performance counters reported by NEs, observes them for anomalous behavior and suggests a corrective action. Typical counters are the number of successful circuit or packet switched calls. The suggested system consists of three modules: a profile learning, an anomaly detection and a diagnosis module. The first module analyzes historical data and learns how the network should usually behave. The second module monitors the current network performance and compares it with the learned profiles. In case a significant difference is detected, the diagnosis module is triggered. Based on a knowledge database populated with fault cases, it tries to identify the possible cause. Furthermore, a performance report containing the suggested corrective action is provided to the operator who can improve the underlying models by providing feedback to the system. Nonetheless, the presented framework focuses extensively on the anomaly detection part, rather than on collisions that occur when rolling back CM changes.

IX. Conclusion

The verification of Configuration Management (CM) changes in a mobile Self-Organizing Network (SON) is a three step process during which we divide the network into sets of cells, trigger an anomaly detection algorithm, and generate CM undo requests in case we spot a degradation in performance. Those sets of cells are referred to as verification areas and are comprised of the reconfigured cell and a certain number of neighbors surrounding it. In the case of an anomalous area, an undo request is generated for the reconfigured cell.

However, the successful completion of this process can be quite challenging since there are factors that may negatively impact its outcome. In this paper, we contributed to the area of SON verification by addressing the problems it still has. First, a verification approach has to be resistant against fluctuating Performance Management (PM) data, i.e., it has to be able to filter out temporal performance drops that appear in the PM data. We achieve that by using exponential smoothing of the deviation from the expected performance. Second, it has to be able to resolve a verification collision, i.e., an uncertainty which undo request to deploy due to an overlap of verification areas. To solve this problem, we make use of constraint optimization techniques. Third, a verification approach requires capabilities for detecting and eliminating false positive collisions. In other words, it has to prevent the unnecessary delay of undo requests from being deployed. We achieve that by using a Minimum Spanning Tree (MST)-based clustering technique that groups similarly behaving cells together. Fourth, it has to be able to re-adapt its deployment strategy if the available execution time is insufficient. We solve this problem by using rules that can be violated under certain circumstances, also known as soft constraints.

The evaluation of our verification approach consists of several parts. During the experiments we observed its capabilities to handle those challenges. The results show that our concept is able to provide a robust solution that is able to circumvent the problems SON verification is currently facing.

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] T. Yamamoto, T. Komine, and S. Konishi, "Mobility Load Balancing Scheme based on Cell Reselection," in *International Conference on Wireless and Mobile Communications*, Venice, Italy, Jun. 2012.

[3] 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*, San Diego, CA, USA, Dec. 2015.

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

[5] 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*, Ottawa, Canada, May 2015.

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

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

[8] "The Oxford Dictionary of English," Revised Edition, Oxford University Press, 2005.

[9] S. Nováczki, "An Improved Anomaly Detection and Diagnosis Framework for Mobile Network Operators," in *International Conference on Design of Reliable Communication Networks*, Mar. 2013.
[10] MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, U. of California Press, Ed., vol. 1, Berkeley, California, 1967, pp. 281–297.
[11] T. Kohonen, "Self-organized formation of topologically correct feature maps," in *Proceedings of Biological Cybernetics*, vol. 69, 1982.
[12] B. Fritzke, "A growing neural gas network learns topologies," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 6, no. 3. MIT Press, 1995, pp. 625–632.
[13] T. Tsvetkov, S. Nováczki, H. Sanneck, and G. Carle, "A Configuration Management Assessment Method for SON Verification," in *International Workshop on Self-Organizing Networks*, Barcelona, Spain, Aug. 2014.
[14] T. Tsvetkov, J. Ali-Tolppa, H. Sanneck, and G. Carle, "A Minimum Spanning Tree-Based Approach for Reducing Verification Collisions in Self-Organizing Networks," in *IEEE/IFIP Network Operations and Management Symposium*, Istanbul, Turkey, Apr. 2016.
[15] R. Romeikat, H. Sanneck, and T. Bandh, "Efficient, Dynamic Coordination of Request Batches in C-SON Systems," in *IEEE Vehicular Technology Conference*, Dresden, Germany, Jun. 2013.
[16] S. Nováczki, T. Tsvetkov, H. Sanneck, and S. Mwanje, "A Scoring Method for the Verification of Configuration Changes in Self-Organizing Networks," in *International Conference on Mobile Networks and Management*, Santander, Spain, Sep. 2015.
[17] Next Generation Mobile Networks Alliance, "A Deliverable by the NGMN Alliance: NGMN 5G White Paper (ver. 1.0)," Feb. 2015.
[18] S. Chen and J. Zhao, "The Requirements, Challenges, and Technologies for 5G of Terrestrial Mobile Telecommunication," *Communications Magazine*, Jan. 2015.
[19] T. Tsvetkov and J. Ali-Tolppa, "An Adaptive Observation Window for Verifying Configuration Changes in Self-Organizing Networks," in *Innovations in Clouds, Internet and Networks*, Paris, France, Mar. 2016.
[20] D. Freedman, R. Pisani, and R. Purves, *Statistics*, ser. International student edition. W.W. Norton & Company, 2007.
[21] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer-Verlag Berlin Heidelberg, 2009.
[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction To Algorithms*, 3rd ed. MIT Press, 2009, ISBN 0262258102.
[23] 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.
[24] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.
[25] NSN, "Self-Organizing Network (SON): Introducing the Nokia Siemens Networks SON Suite - an efficient, future-proof platform for SON," White Paper, Oct. 2009.
[26] 3GPP, "Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.2.0)," 3rd Generation Partnership Project (3GPP), Technical report TR 101 112 V3.2.0, Apr. 1998.
[27] ——, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," 3rd Generation Partnership Project (3GPP), Technical Specification 36.213 V12.1.0, Mar. 2014.
[28] 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*, Ottawa, Canada, May 2015.
[29] E. W. Weisstein, "Minimum Vertex Coloring From MathWorld - A Wolfram Web Resource," Feb. 2015.
[30] T. Bandh, R. Romeikat, and H. Sanneck, "Policy-based coordination and management of SON functions," in *IFIP/IEEE International Symposium on Integrated Network Management*, Dublin, Ireland, May 2011.
[31] T. Kürner, M. Amirijoo, I. Balan, H. van den Berg *et al.*, "Final Report on Self-Organisation and its Implications in Wireless Access Networks," Self-Optimisation and self-ConfiguRATion in wirelEss networkS (SOCRATES), Deliverable D5.9, Jan. 2010.
[32] T. Bandh, R. Romeikat, H. Sanneck, L. C. Schmelz, B. Bauer, and G. Carle., "Optimized Network Configuration Parameter Assignment Based on Graph Coloring," in *IEEE/IFIP Network Operations and Management Symposium*, Osaka, Japan, Apr. 2010.
[33] S. Nováczki and B. Gajic, "Fixed-Resolution Growing Neural Gas for Clustering the Mobile Networks Data," in *International Conference of Engineering Applications of Neural Networks*, Sep. 2015.

Tsvetko Tsvetkov graduated in November 2012 with a M.Sc. in computer science at the Technical University of Munich, Germany. In January 2013 he joined the chair on Network Architectures and Services at the department of computer science, Technical University of Munich. At the same time, he started as an external expert on OAM automation at T&I Research, Nokia Siemens Networks (now Nokia Bell Labs), Munich, Germany. His research interests include the automation and management of cellular networks, SON coordination, verification, and anomaly detection. Further, he is interested in machine learning techniques as well as energy saving management.

Janne Ali-Tolppa graduated in 2003 with a M.Sc. in computer science at Tampere University of Technology, Tampere, Finland. He joined Nokia Networks the same year. Since then he has worked developing network management solutions in different roles in Finland, China and Germany in projects involving improved methods for multi-vendor integration and network security management. Currently, he is working as a research engineer at Nokia Bell Labs, Munich, Germany, with a focus on Cognitive Network Management and Self-Organizing Networks. Special areas of interest include analytics, anomaly detection, machine learning, self-healing, small cells and energy saving management.

Henning Sanneck studied electrical engineering at the University of Erlangen-Nuremberg, Germany. After receiving his Diploma in 1995, he joined GMD FOKUS in Berlin. At FOKUS, he worked as a researcher in the area of QoS support for Real-Time Services in IP-based networks. He received his Dr.-Ing. degree in Electrical Engineering from the Technical University of Berlin. In 2001 he joined Siemens - Mobile Networks in Munich, as a Senior Research Engineer. In 2007, at the formation of Nokia Siemens Networks, Henning started to lead the "Network Management Automation" team as a Research Manager with a focus on LTE SON. Since 2014, as Head of Cognitive Network Management, his responsibility also includes the internal coordination of Research and Standardization work in Network Management Automation.

Georg Carle is full professor in computer science at the Technical University of Munich, where he holds the chair on Network Architectures and Services. He conducts research on Network Security, Peer-to-Peer and Overlay Networks, Quality of Service support and measurements for IP networks, Active Networking, voice and video services over IP, charging and accounting. Georg Carle received a M.Sc. degree from Brunel University London in 1989, a diploma degree in electrical engineering from the University Stuttgart in 1992 and a doctoral degree from the faculty of computer science of University Karlsruhe in 1996. From 1992 to 1996 he worked at the Institute of Telematics at the University Karlsruhe, being supported by a "Graduiertenkolleg" scholarship. In 1997, he worked as postdoctoral researcher at Institut Eurécom, Sophia Antipolis, France. In October 1997 he joined GMD FOKUS in Berlin. From January 2003 to March 2008, he has been full professor at University of Tübingen.