A Steiner Tree-Based Verification Approach for Handling Topology Changes in Self-Organizing Networks

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

***Abstract*—In today's Self-Organizing Networks (SONs) we differentiate between closed-loop functions, which have a predefined absolute goal, and such that form an action plan that achieves a high expected utility. Both function types perform changes to Configuration Management (CM) parameters, but only the second type may re-adapt the action plan in order to maximize the utility. A SON verification approach is one member of this particular function class. It is seen as a special type of anomaly detection that divides the network into sets of cells, triggers an anomaly detection algorithm for those sets, and finally generates CM undo actions for the abnormally performing cells.**

Unfortunately, one of the challenges verification strategies are facing are network topology changes. Typically, cells are switched on or off when energy saving features are enabled. However, enabling or disabling cells can negatively influence a verification mechanism which may create a suboptimal action plan or even blame certain CM changes that actually did not harm performance. In order to overcome this issue, we present an approach that is based on Steiner trees. In graph theory, a Steiner tree is a Minimum Spanning Tree (MST) whose costs can be reduced by adding additional vertexes to the graph. We use this tree to filter out anomalies caused by topology adjustments and such induced by other CM changes.

In this paper, we also evaluate the proposed solution in several scenarios. First, in a simulation study we evaluate the functions that are used to build the Steiner tree. Second, we show how it positively affects the network performance when having concurrent CM and topology changes.

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. It has the purpose to optimize the operation of the network, supervise the configuration and auto-connectivity of deployed Network Elements (NEs), and enable automatic fault detection and resolution [1]. In order to perform those tasks, though, such a network has to be managed by a set of autonomous SON functions that are designed to perform specific network management tasks. Usually, they are implemented as closed control loops which monitor Performance Management (PM) and Fault Management (FM) data, and depending on their objectives, change Configuration Management (CM) parameters. For example, the Mobility Robustness Optimization (MRO) function tries to minimize the call drops, radio link failures as well as unnecessary handovers by adjusting the Cell Individual Offset (CIO) [2].

However, there is another class of SON functions, namely such forming a course of actions that have high expected utility rather than having a predefined absolute goal. Such functions plan under uncertainty, either because they have incomplete information about the world, or because their actions have uncertain effects on the environment. In literature, this type of planning is referred to as Decision-Theoretic Planning (DTP) [3]. A common approach for overcoming uncertainties is to make the actions of the plan depend in some way on the information gathered during execution. This particular strategy is referred to as feedback planning.

One representative of this class is SON verification [4]–[6]. It is seen as a special type of anomaly detection which monitors CM changes within the network and rolls back those causing abnormal behavior. Based on those changes, a verification approach partitions the network into sets of cells, also called verification areas, assesses the performance of each area by triggering an anomaly detection algorithm, and generates a CM undo action for those that are abnormally performing. Furthermore, there are uncertainties when it comes to executing undo actions that impact a shared set of cells. Hence, a verification mechanism may serialize the deployment process and successively observe the impact of each action.

One problem that verification approaches still have, is how they react to topology changes. Usually, such changes occur when cell energy saving features are activated. However, disabling or enabling cells creates uncertainties during the process of verification which may lead to a suboptimal action plan or, even worse, the deployment of suboptimal actions. For instance, turning on a cell may cause an anomaly at its neighbors since they did not expect a topology change like that to happen. In a similar manner, we are facing this problem when we do the opposite, namely turning off a cell. The neighbors may expect that the cell is always switched on during its operation. Furthermore, the fact that we need to enable or disable cells typically means that the network and service demands have drastically changed, e.g., because

978-3-901882-85-2

numerous User Equipments (UEs) have either entered or left the network. An event like that can also induce anomalies at already enabled cells for which a verification mechanism cannot provide an appropriate corrective undo action.

In order to become more resistant against topology changes, we present a verification approach based on Steiner trees. The Steiner tree problem is a combinatorial optimization problem that tries to reduce the length of a Minimum Spanning Tree (MST) by adding extra vertexes and edges to the initial edge weighted graph. Those additional vertexes are referred to as Steiner points whereas the initial nodes are called terminals. In general, Steiner points represent cells that can be turned on or off during their operation whereas terminals describe cells that remain always switched on. Based on whether a cell is used as a Steiner point to form the tree, we decide if and how to consider it while generating the undo action plan.

The remainder of this paper is structured as follows. In Section II we describe the generic structure of feedback planning functions and the components of a verification mechanism. Section III gives an overview of the challenges experienced by SON verification. In Section IV we present our concept. Section V is devoted to the evaluation of the introduced approach. The paper concludes with the related work in Section VI and a summary in Section VII.

II. Background

SON functions that utilize feedback planning operate in three phases: an observation, an assessment, and an action phase. In the following sections, those phases are described and examples are given. It should be noted that the examples focus on SON verification, as introduced at the very beginning of this paper.

A. Observation phase

Let us start by denoting the set of all cells in the network as Σ. This gives us the opportunity to define the scope of a feedback planning function which is a set of cells Σ^M, where $\Sigma^M \subseteq \Sigma$. Those cells are actively monitored by the function. Moreover, Σ^M is further split into nonempty subsets $\{\Sigma_1^{M'}, \ldots, \Sigma_i^{M'}\}$, which are called *observation areas*. If we denote this split as $\mathcal{P}^\Sigma$, the following conditions must hold:

- $\emptyset \notin \mathcal{P}^\Sigma$
- $\bigcup_{\Sigma^{M'} \in \mathcal{P}^\Sigma} \Sigma^{M'} = \Sigma^M$
- for any two sets $\Sigma_1^{M'}, \Sigma_2^{M'} \in \mathcal{P}^\Sigma \colon \Sigma_1^{M'} \neq \Sigma_2^{M'}$

In SON verification, those areas are referred to as *verification areas* and are formed by taking the reconfigured cell, also known as the *target cell*, and a set of cells surrounding it, called the *target extension set*. In particular, the extension set consists of neighboring cells that are possibly impacted by the reconfiguration of the target, e.g., by taking all first degree neighbors. Note that two cells are called neighbors only if there is a neighbor relation that allows them to handover UEs between each other.

In addition, the size of an observation area can be subject to the location of the cells. For instance, in SON verification a cell that is part of dense traffic or known trouble spots may get forced to join a verification area, even if it was not initially supposed to do so.

B. Assessment phase

During this phase, the performance of each observation area $\{\Sigma_1^{M'}, \ldots, \Sigma_i^{M'}\}$ is assessed. In SON verification, anomaly detection algorithms are used for that purpose. By definition, an anomaly is understood as "something that deviates from what is standard, normal, or expected" [7]. Usually, the target of verifying CM changes is to detect *abnormal* cell behavior. As a result, the network has to be *profiled*, i.e., analyzing the network performance indicators and specifying how the network should usually behave [8]. Once the cell performance considerably differs from all learned states of the normal network operation, it is marked as anomalous and is later considered during the plan generation and action execution phase. In literature, several approaches have been introduced for learning faultless states, e.g., by making use of clustering algorithms like K-means [9] or Self-Organizing Maps [10].

C. Action phase

During the action phase, a feedback planning function creates a deployment plan that specifies the action execution order. Let us denote the set of all actions as U and the plan as $\mathcal{P}^U$. The plan itself is a partition [11] of U, i.e., a grouping of U into non-empty subsets $\{U_1', \ldots, U_i'\}$, also called blocks or steps. Thereby, the following conditions must hold:

- $\emptyset \notin \mathcal{P}^U$
- $\bigcup_{U' \in \mathcal{P}^U} U' = U$
- if $U_1', U_2' \in \mathcal{P}^U$ and $U_1' \neq U_2'$ then $U_1' \cap U_2' = \emptyset$

In SON verification, an element of U' is a CM undo action for the target cell of a verification area. The action itself includes the IDs of the target and the cells of the extension set as well as a previous stable configuration of the target.

In addition, while creating the plan a feedback planning function has to consider two aspects. At first, it has to make sure that $\mathcal{P}^U = \{U_1', \ldots, U_i'\}$ is *collision-free*, i.e., for any two actions $u_j, u_k \in U_i' \colon \theta(u_j, u_k) = \emptyset$. Here, function θ returns the cells due to which two actions are marked as being in a collision, i.e., $\theta \colon U \times U \to \mathbb{P}(\Sigma)$.

In SON verification, the outcome of θ are the anomalous cells that are shared between the verification areas of the target cells for which the undo actions are generated. This is also known as a *verification collision* [12], i.e., an uncertainty which undo action to execute. This procedure delays the verification process since in the worst case scenario the deployment plan $\mathcal{P}^U = \{U_1', \ldots, U_i'\}$ can be completely serialized, i.e., $\forall i \colon |U_i'| = 1$.

Second, a deployment plan $\mathcal{P}^U$ has to be *gain-aware*, i.e., $\mathcal{G}(U_1') \geq \cdots \geq \mathcal{G}(U_i')$, where $\mathcal{G} \colon \mathbb{P}(U) \to \mathbb{R}$ is a function that returns the gain of deploying a set of actions. In SON verification, the deployment order maximizes the probability of returning the network performance to the expected state. Consequently, CM changes that are most likely causing a degradation must be undone at first.

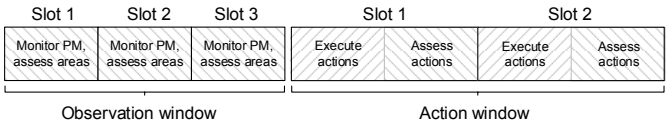


Figure 1. Example of a 3-slot observation and 2-slot action window

D. Time windows

A feedback planning function requires two time windows, as shown in Figure 1. The first one is known as the *observation window*, during which each observation area is monitored. 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 algorithm used during the assessment phase.

The second window is referred to as the *action window* and is used for the deployment of the actions as well as for assessing their impact on the network performance. The length of an action window slot depends on the delay until the impact of those actions becomes visible in the PM data. Furthermore, the total number of available slots depends on the function type. In SON verification, the number is subject to the environment, e.g., in a highly populated area we may have a short action window since the restoration of the network performance has to be carried out as fast as possible.

III. Problem analysis

A. Problems

In general, topology changes induce uncertainties while a CM verification mechanism is running. They may lead to a suboptimal deployment plan and even to the rollback of CM changes that are not harming performance. Let us explain the problem by giving an example. Figure 2 depicts an exemplary network that consists of 9 cells and 10 cell adjacencies. Note that a verification area consist of the first degree neighbors of the target cell. Let us assume that cells 2, 4, and 7 have been reconfigured, cell 6 has been disabled, and cells 3 and 8 have degraded. Hence, for this network snapshot we have three undo actions u_2, u_4, u_7 and a verification collision because of the undo actions for cell 2 and cell 4. The collision occurs due to cell 3 which happens to be anomalous and shred between the two verification areas. Thus, a permissible deployment plan $\mathcal{P}^U$ is $\{\{u_2, u_7\}, \{u_4\}\}$ or $\{\{u_4, u_7\}, \{u_2\}\}$.

However, the situation becomes different as soon as topology changes are allowed. For instance, switching a cell on or off creates a potential for anomalies. One reason for that are *incomplete profiles*. As stated in Section II, a profile defines how cells should usually behave, i.e., it also specifies how a cell should behave with respect to its neighbors, as well as defines the expected performance from its neighbors. Typically, the assessment of the neighbor performance is based on handover Key Performance Indicators (KPIs) like the ping-pong rate or the Handover Success Rate (HOSR). However, if we add or remove a neighbor of a cell, and do not have an updated profile that specifies how the cell should behave

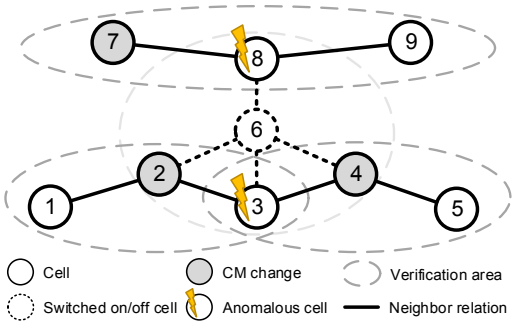


Figure 2. Exemplary network of 9 cells and 10 cell adjacencies

afterwards, we may induce an anomaly and even mark for an undo changes that did not do any harm and were actually required. In the aforementioned example, the enabling or disabling of cell 6 may be the actual anomaly cause, and not the CM changes made at cells 2, 4, and 7.

Furthermore, besides CM changes there are events that may cause anomalies and which cannot be corrected by executing an undo action. Usually, they are triggered by *UE movements*, or more precisely, by users unexpectedly entering or leaving the network. If a rather large group of UEs joins or leaves, it may induce an unusually high or low load and throughput. For example, the anomalies at cell 3 and 8 in Figure 2 can be caused by the recent movements of such UE groups. As a result, a verification mechanism would need to dismiss the deployment plan in order to prevent the rollback of regular changes, and rely on an Energy Saving Management (ESM) function [13] to turn on the appropriate cells.

B. Causes

One of the causes is the delay until we get statistically relevant PM data. Significant changes in KPIs like the cell load, throughput, or HOSR can be observed only when enough users are actively using the network, e.g., during the peak hours of a weekday. As a result, the verification of certain CM changes is possible only when such data is available. Moreover, this also induces delays while updating an incomplete profile.

Another cause is the presence of a high number of buffered undo actions. In such a situation the probability of rolling back regular changes increases if, in addition to that, topology changes are made. Table I visualizes the deployment plan generated for a real LTE network. As shown, we have a high number of undo actions allocated already to the first action window slot. In order to lower the probability of undoing good changes we can distribute the undo actions to more plan

Table I
Undo action plan for an LTE network [12]

Action window slot	Undo actions 2013-11-28	Undo actions 2013-12-04
1	166	52
2	2	0

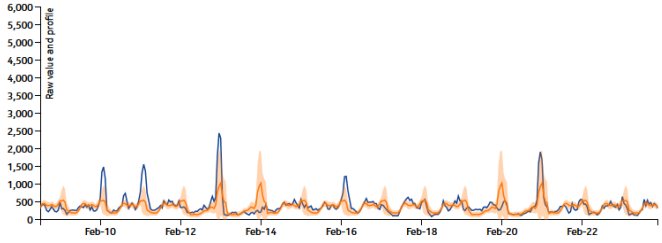


Figure 3. Traffic analysis of an LTE cell

blocks and assess in this way their impact on the network performance more frequently. However, this will also delay the process of deploying necessary undo actions. In [12], we have shown that the unnecessary increase of the plan blocks has a negative impact on the network.

The reason why topology changes are required is given in Figure 3. It shows the performance statistics of an LTE cell located near a stadium in a large city. In particular, it represents the KPI value as time series. The orange line is the mean, the orange shadowing the standard deviation in the profile and the blue line is the actual value. The considered KPI, used also for computing the profile, is the number of Radio Resource Control (RRC) connection setup attempts. February 13^{th} 14^{th} as well as 20^{th} and 21^{th} are weekends, which is clearly visible in the profile. However, the observed traffic line also shows that the actual traffic does not always follow the profile which was mainly caused by events that were taking place in the stadium. The solution would be to enable hot-spot cells that would take over traffic from the macro cell.

IV. Concept

The question that we aim at answering is how to minimize the negative impact on SON verification while facing topology changes. Generally speaking, this can be seen as a Steiner tree problem [14] which is very similar to the MST problem [15]. For a given undirected edge weighted graph, we have to find a tree that interconnects all vertexes and, at the same time, is of the shortest length. The difference between the MST and the Steiner tree problem is that to solve the latter one we may include extra vertexes to the graph that reduce the length of the spanning tree. Those new vertexes are called Steiner points, whereas the initial nodes of the graph are referred to as terminals.

However, the Steiner algorithm does not give us directly a solution to the problem. First of all, we need a metric that defines the state of a cell as well as a function that computes the weights of the edges. Second, we need to define which entities in the network are marked as terminals and which as Steiner points. In the following sections, we are going to describe each of those steps as well as give an example.

A. Cell state and weighting function

The Steiner algorithm takes an undirected edge weighted $G = (V, E, d)$ as input, where V is the set of vertexes, E the set of edges $V \times V$, and d a function computing the edge weights. In our case, the vertexes represent cells, whereas the edges neighbor relations. To define d, though, we need to consider several steps.

At first, for each cell in the network we take n *utilization KPIs* k_i, where $i \in [1, n]$ and all k_i are element of $\mathbb{R}$. Thereby, every k_i represents the value of the given utilization KPI. Examples of such KPIs are the current cell load or throughput. Those KPIs are put together in a KPI vector $\mathbf{k} = (k_1, \ldots, k_n)$, i.e., $\mathbf{k} \in \mathbb{R}^n$. Furthermore, for the set of all cells Σ, $K = \{\mathbf{k}_1, \ldots, \mathbf{k}_m\}$ is called the KPI utilization vector space, where $|\Sigma| = |K| = m$.

Next, for each utilization KPI k_i a *KPI anomaly level* value $a_i \in \mathbb{R}$ is computed. It represents the deviation from the expected KPI value. Thereby, for each cell an anomaly level vector $\mathbf{a} = (a_1, \ldots, a_n)$ is formed, where $\mathbf{a} \in \mathbb{R}^n$. In addition, $A = \{\mathbf{a}_1, \ldots \mathbf{a}_m\}$ is called the anomaly level vector space, where $|A| = |K|$.

However, the computation of vector $\mathbf{a} = (a_1, \ldots, a_n)$ depends on the cell type. First, we have cells that are never turned off, which in this paper are referred to as *static cells*. The most important fact about such cells is that they have a profile vector $\mathbf{p} = (p_1, \ldots, p_n)$. Each element p_i can be seen as a value or even as an interval of values that specifies the usual range of k_i in vector $\mathbf{k}$. The calculation of the anomaly level vector is done by function φ (Equation 1) which takes a profile vector $\mathbf{p}$ and the current KPI vector $\mathbf{k}$ and returns the corresponding anomaly level vector $\mathbf{a}$.

$$\varphi \colon P \times K \to A \tag{1}$$

Function φ, however, cannot be applied to cells that can be switched on or off, also called *on-demand cells*. The reason is the lack of a profile. Therefore, instead of using an own profile, an on-demand cell takes the anomaly level vectors $\mathbf{a}_j \in A$ and cell UE reports $\mathbf{r}_j \in R$ of its neighbors. Let us denote the number of those neighbors as ν, i.e., $j \in [1, \nu]$. Hence, we get a modified function φ', as shown in Equation 2.

$$\varphi' \colon \{\mathbf{a}_1 \ldots \mathbf{a}_\nu\} \times \{\mathbf{r}_1 \ldots \mathbf{r}_\nu\} \to A \tag{2}$$

It should be noted that a report $\mathbf{r}_j$ is a vector that includes information like the number of UEs served by a cell.

Finally, we define d as a function that takes the anomaly vectors of two cells, as presented in Equation 3.

$$d \colon A \times A \to \mathbb{R} \tag{3}$$

B. Steiner tree-based verification algorithm

1) Steiner point selection: To begin with, let us split the set of all cells Σ into the set of all static cells Σ^{SC} and all on-demand cells Σ^{OC}, where $\Sigma^{SC} \cup \Sigma^{OC} = \Sigma$. All elements of Σ^{SC} are considered as terminals whereas all disabled on-demand cells are marked as Steiner points. However, enabled on-demand cells are evaluated by a Steiner point assessment function ι, as follows:

$$\iota \colon \Sigma^{OC} \times K \to V^S \cup \emptyset \tag{4}$$

The decision to exclude an on-demand cell from the set of Steiner points V^S depends on the most recent KPI vector $\mathbf{k}$.

For instance, if it is continuously reporting a low load, it will be set as a Steiner point. On the contrary, an on-demand cell that is continuously experiencing a high load will be considered as a terminal. The reason is that the cell may be still required in the future.

2) Algorithm description: The Steiner tree problem is an NP-complete problem which is why in practice heuristics are most commonly used. A well known algorithm is the one introduced by Kou, Markowsky and Berman [16]. Algorithm 1 lists a slightly modified version of that approach.

The algorithm itself starts by selecting the set of Steiner points $V^S \subset V$ (line 1). The remaining vertexes, denoted as V^T (line 2), are set as terminals. In line 3, an undirected distance graph $D_G(V^T)$ is formed. The set of vertexes of $D_G(V^T)$ is denoted as V^T. The edge wights correspond to the costs given by the shortest paths between the terminals in G. To compute those distances we use Dijkstra's algorithm [15].

As a next step, $D_G(V^T)$ is transformed to an MST, which is denoted as T_{D_G} (line 4). In particular, we take Kruskal's algorithm [15]. Should the outcome yield multiple trees, an arbitrary MST is chosen (lines 5-7). The selected tree is then transformed to a graph G' by replacing each edge by the corresponding shortest path from G (line 8). The newly formed graph G' is transformed to an MST T' (line 9) by using Kruskal's algorithm.

Finally, the Steiner tree $T^S = (V^{T^S}, E^{T^S})$ is formed by continuously removing non-terminal leaves from T' (line 10). The set of unnecessary Steiner points V^R is the complement of the set of Steiner points V^S and the vertex set V^{T^S}, as shown in line 11.

3) Corrective action: After executing the algorithm, the on-demand cells represented by the Steiner points or terminals that form the Steiner tree T^S are allowed to be turned on. The cells represented by the set of unnecessary Steiner points V^R are disabled.

Algorithm 1: Steiner tree algorithm

Input: Undirected, edge weighted graph $G = (V, E, d)$
Result: Steiner tree T^S of G and a set of unnecessary Steiner points $V^R \subset V$

1 Select Steiner points $V^S \subset V$;
2 $V^T \longleftarrow V \setminus V^S$;
3 Construct a complete undirected distance graph $D_G(V^T)$;
4 Compute a minimum spanning tree T_{D_G} of $D_G(V^T)$;
5 **if** *Multiple* T_{D_G} *present* **then**
6 Select an arbitrary minimum spanning tree;
7 **end**
8 Form G' by replacing each edge in T_{D_G} by the corresponding shortest path from G;
9 Form a minimum spanning tree T' from G';
10 Compute $T^S = (V^{T^S}, E^{T^S})$ by continuously removing leaves from T' that are $\notin V^T$;
11 $V^R \longleftarrow V^S \setminus V^{T^S}$;

C. Example

Let us assume that we have a network that consists of four static and three on-demand cells. Let the static cells have an ID between 1 and 4 whereas the on-demand cells an ID within the range of 5 to 7. Figure 4(a) shows the existing neighbor relations as well as the state of the on-demand cells. As shown, cell 5 is the only enabled on-demand cell.

The input graph G fed into the Steiner algorithm is shown Figure 4(b). The edge weights are based on the cell load, i.e., the higher the worse. Furthermore, we have the following initial sets of Steiner points and terminals: $V^S = \{v_5^S, v_6^S, v_7^S\}$ and $V^T = \{v_1^T, v_2^T, v_3^T, v_4^T\}$. The complete undirected distance graph $D_G(V^T)$ is visualized in Figure 4(c). The subsequent MST transformation yields T_{D_G}, as outlined in Figure 4(d). Finally, the Steiner tree T^S is given in Figure 4(e). Due to their assignment in T^S, cells 6 and 7 are switched on whereas cell 5 is turned off.

V. Evaluation

A. Simulation environment

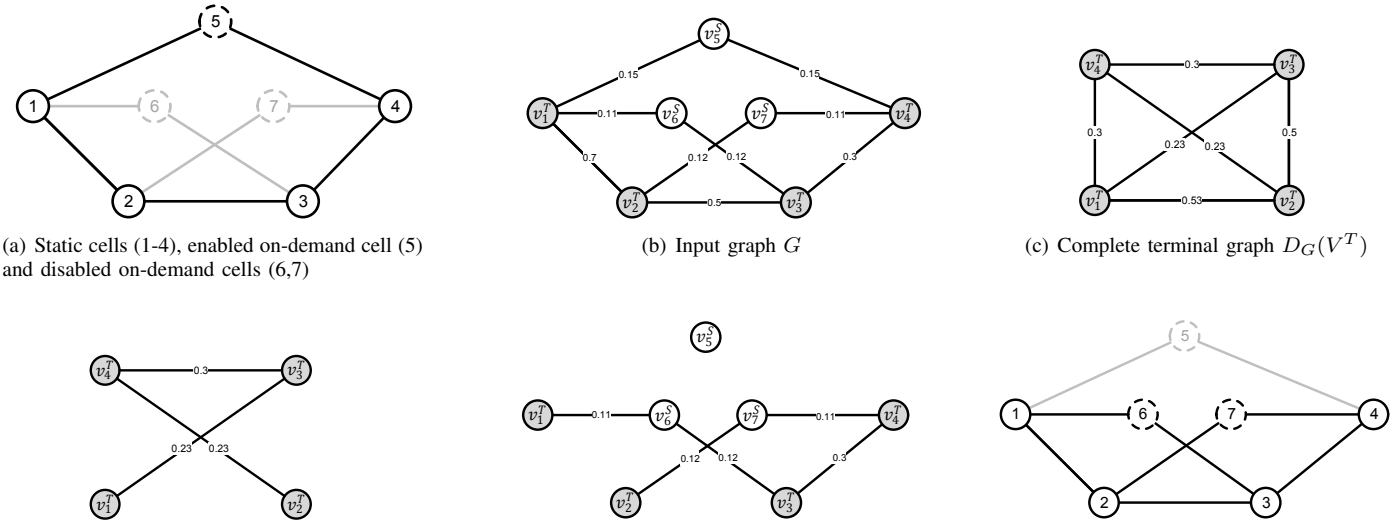
The simulation environment is called the SON Simulation System (S3) [12]. It consists of a set of closed-loop SON functions, a verification mechanism, as well as an LTE radio network simulator. The latter one is also part of the SON simulator/emulator suite [17], whose parameters are summarized in Table II. Note that the simulated scenario covers parts of Helsinki, Finland.

The LTE simulator performs continuous simulation by tracking the changes in the network over time. The time itself is divided into time slices, also called simulation rounds, which represent PM granularity periods. At the beginning of a round, the simulator configures the network as defined by the CM setup. During a round, a group of 1500 uniformly distributed users follow a random walk mobility model and actively use the mobile network. In addition, up to four additional UE groups may randomly join or leave the network. At the end of a round, PM data is exported for each cell and is used as input by every SON function. Those functions may then perform CM changes whose impact can be assessed during the next simulation round.

Furthermore, the following cell KPIs are considered by the verification mechanism:

- Channel Quality Indicator (CQI): computed as the weighted harmonic mean of the channel efficiency. The efficiency values are listed in [18].
- HOSR: counts the number of successful handovers divided by the total number of handovers.
- Handover ping-pong rate: the number of handover ping-pongs [1] divided by the total number of handovers.
- Cell load: calculation based on the number of utilized Physical Resource Blocks (PRBs) by 50% of the users.

Note that the first three are used for degradation detection, as defined in [12], whereas the last one by the Steiner approach.



(a) Static cells (1-4), enabled on-demand cell (5) and disabled on-demand cells (6,7)

(b) Input graph G

(c) Complete terminal graph $D_G(V^T)$

(d) MST T_{D_G} of $D_G(V^T)$

(e) Steiner tree T^S

(f) Allowed topology changes: disabling and enabling on-demand cells (5) and (6,7)

Figure 4. Concept example with 4 terminals (in gray) and 3 Steiner points (in white)

Table II
SON SIMULATION SYSTEM SETUP

Parameter	**Value**
Network Frequency	2000 MHz
Bandwidth	20 MHz
Number of PRBs	100
Shannon gap	-1.0 dBm
Thermal noise	-114.447 dBm
Pathloss coefficient La	128.1
Pathloss coefficient Lb	37.6
Path loss model	UMTS 30.03 [19]
Downlink scheduler mode	CBR mode
Shadowing correlation distance	50.0 m
Shadowing standard deviation	8.0 dB
RLF threshold	-6.0 dB.
RLF disconnection timer	0.3 sec
RLF reconnection timer	1.0 sec
Handover hysteresis threshold	2.0 dB
Handover ping-pong detection duration	4.0 sec
Handover states detection offset	1.0 sec
Handover timeout period	3.0 sec
Total macro cells	32 macro cells
Total small cells	9 small cells
Antenna height (macro cells)	17-20 m
Antenna height (small cells)	5-10 m
Simulated time during a round	5400 sec
Size of the 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

B. Parameter selection

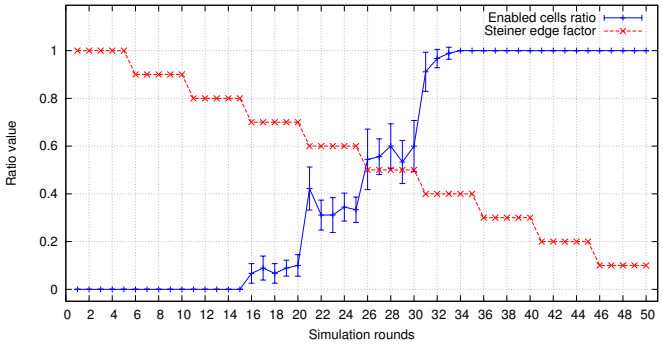
1) Function φ: An element p_i, where $i \in [1, n]$, of a profile vector $\mathbf{p} = (p_1, \ldots, p_n)$ is a sequence of t observations which were collected while the network was operating as expected. During this phase, all on-demand cells were disabled. The outcome of φ is calculated by taking the sequence of observations of p_i, as well as the current k_i, and computing the z-score [20] of all $t + 1$ values. Each z-score represents the distance between the current value and the sample mean in units of the standard deviation. Here, the z-score of k_i is the KPI anomaly level a_i.

To give an example, suppose that p_{Load} contains $\{70.1\%, 70.2\%\}$ and the current k_{Load} is 80.9%. Hence, after applying φ we get a load anomaly level a_{Load} of 1.15, i.e., the current k_{Load} is 1.15 standard deviations above the sample mean.

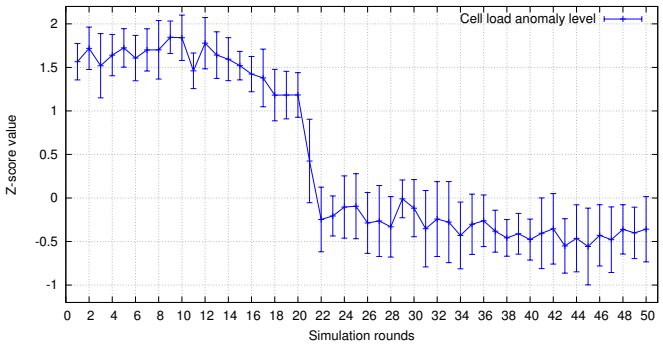
2) Function φ': It computes the weighted sum of the load anomaly levels of the direct static neighbors of an on-demand cell. The weight itself equals the served UE ratio, i.e., the number of UEs served by a direct neighbor divided by the total number of UEs served within the area.

3) Function d: In order to prevent edge weights $\in (-\infty, 0]$, all load anomaly level values are put within the interval of $[1; 2]$. The weight of an edge in G is computed by summing up the load anomaly level of the two cells. Furthermore, if one of the cells is an on-demand cell, only a portion of the edge weight is taken by multiplying it with a *Steiner edge factor*. In addition, if an on-demand cell gets enabled and shortly after that disabled, the Steiner edge factors of all edges, leading to the vertex repressing it, are increased.

4) Function ι: Function ι is based on exponential smoothing of the utilization KPIs, in particular, the cell load. In case the smoothed cell load value falls below a threshold of 15%, an on-demand cell is considered as a Steiner point, otherwise as a terminal.



(a) Correlation between the Steiner edge factor and number of enabled cells



(b) Impact on the cell load anomaly level

Figure 5. Evaluation of the edge weighting function d

C. Results

1) Edge weighting function: We start with an evaluation of how function d impacts the Steiner tree T^S. In particular, we are interested in how the factor used to multiply the weight between an on-demand and another cell affects the outcome. To do so, we deploy an additional user group that consists of 150 UE to one particular part in the network that is covered by 3 of the 9 on-demand cells. This should induce an anomaly in the surrounding macro cells.

The experiment starts with a Steiner edge factor of 1.0 which is decreased by 0.1 after every fifth simulation round. Note that all cell states are reset after changing the Steiner edge factor. Figure 5(a) shows the results, in particular, the percentage of on-demand cells that get enabled for the given Steiner edge factor. For a factor between 1.0 and 0.8, none of the on-demand cells are switched on, i.e., no Steiner points are added to the graph. After decreasing its value to 0.7, the first on-demand cells are allowed to be switched on. When selecting a factor of 0.6, approximately 33% of all on-demand are enabled. A factor of 0.5 allows roughly 60% of those cells to be added to the Steiner tree, whereas a factor of 0.4 and lower permits all on-demand cells to become active.

Figure 5(b) shows the average load anomaly level of all on-demand cells as well as all of their static neighbors. As presented, for a Steiner edge factor of 0.6 or less, the z-score ranges within approximately the same interval. Note that all figures represent 95% confidence intervals that are computed around the sample mean of a certain number of consecutive test runs. Here, the total number of test runs is 10.

2) Steiner point assessment function: Next, we are going to test function ι, which basically converts a Steiner point to a terminal and vice versa. Since it is based on exponential smoothing, we will evaluate the parameter used to update the smoothed value. We start with an update factor of 1.0 and decrease it gradually by using a step size of 0.1. In contrast to the previous setup, the evaluation window is set to 10 rounds. We start with the same setup as before, however, after the fifth round the UE group is removed. Afterwards, we count the simulation rounds during which the enabled but no longer necessary on-demand cells stay switched on.

Figure 6 gives us the results of this observation after 10 test runs. For a factor of 1.0, the on-demand cells are almost immediately disabled after removing the group. It is indicated by the low percentage of simulation rounds during which cells remain turned on. However, as we start decreasing the update factor, more on-demand cells stay switched on. For a factor of 0.2 none of the cells are disabled, even though the UE group has disappeared.

3) Compared strategies: On the one hand, we have the Steiner tree-based verification, as introduced in Section IV. For this setup, the cells that are verified by the Steiner algorithm are not processed by the CM verification process. On the other hand, we have only CM verification, as presented in [12]. This setup represents the verification strategy that is being used today. It monitors the activity of all SON functions, including those optimizing the energy consumption, and rolls back the changes harming performance.

4) Steiner tree-based verification: To begin with, we allow a Coverage and Capacity Optimization (CCO) function [1] to optimize the antenna tilt of 16 of the 32 macro (static) cells. Those cells have at least one small (on-demand) cell as a neighbor. Second, in order to trigger the wake up or sleep mechanism of the small cells, we insert four UE groups that have the following size: 150, 75, 85, and 120 users. Hence, we should see simultaneous CM and topology changes.

In total, we have 7 test runs each lasting 20 rounds. Moreover, after every fifth round we randomly change the UE group. The selection is made by adding all UE groups to a list, permuting the list, where all permutations occur with equal likelihood, and selecting the first item. Every time we change the UE groups, we measure the average anomaly level of each of the 16 macro cells. Hence, we have 28 samples for each of those cells. It should be noted that the Steiner edge update factor is set to 0.6 whereas the smoothing update factor to 0.5. The selection is motivated by the results from the experiments that are described in Sections V-C1 and V-C2.

Figure 7 visualizes the results. As outlined, the Steiner tree-based verification approach manages to improve the anomaly level of all 16 cells, i.e., putting it near zero which is the expected state. Especially the anomaly level of cells 6, 9, 10, 18, and 24 is significantly changed. On the contrary, using only CM verification leads to a worse anomaly level, which is caused also by the undo of CCO changes. Those changes were blamed although they did not do any harm.

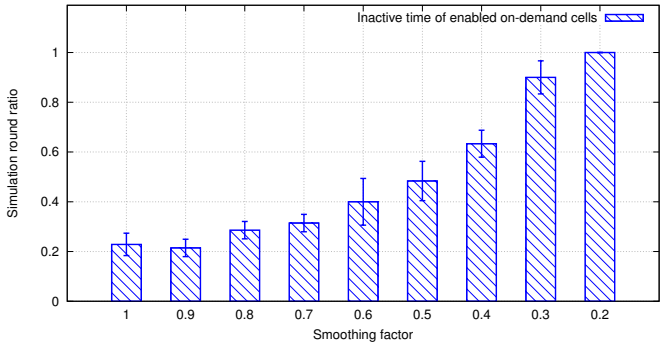


Figure 6. Evaluation of the Steiner point assessment function ι

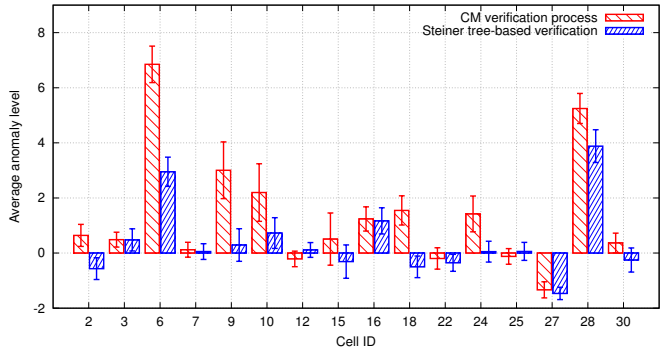


Figure 7. Evaluation of the Steiner tree-based verification approach

VI. Related work

The concept of SON coordination [21], [22] can be seen as a partial solution to the problems outlined in this paper. It is referred to as a pessimistic approach that specifies rules required for the detection and resolution of *known conflicts* between active SON functions. Hence, it prevents conflicting functions from getting active, rather than assessing the network performance after deploying the CM changes. For example, conflicts occur 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. Therefore, a coordination approach can be used for preventing rollback decisions while changing the topology of the network and vice versa. However, this will completely serialize the two operations which means that we may either prevent important undo actions from being deployed because cells have been turned on or off, or hinder cells from getting enabled since the verification process is still processing its deployment plan.

Within the SOCRATES project [23] the idea of having a function that detects undesired behavior in SONs is discussed. The authors 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. Despite the given ideas, neither a concept is given, nor solutions to SON verification-related problems are presented. In particular, the issue of having concurrent topology changes is neglected.

In [24], a framework is proposed for detecting anomalous network behavior, performing root cause analysis and providing a corrective action. It forces each NE to observe its own operation by measuring several types of performance indicators which are then uploaded to the operator's database. The anomaly detector determines whether the data is significantly deviating from the expectations. In the case of an anomaly, the diagnosis component may suggest a corrective action to the operator. Topology changes are not considered by the framework, i.e., it has the shortcomings of a CM verification approach, as described in Section III-A.

In [25], an anomaly detection technique for cellular networks has been introduced. It is based on the incremental clustering algorithm Growing Neural Gas (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 problem of representing the input data and determining when to stop sampling PM data. The solution does not take into account the issues caused by topology changes, i.e., the CM verification problems apply for it as well.

VII. Conclusion

In today's Self-Organizing Networks (SONs), we can distinguish between two generic types of functions. First of all, we have closed-loop SON functions which monitor Performance Management (PM) and Fault Management (FM) data from the network and change Configuration Management (CM) parameters based on a predefined absolute goal. Second, there is a class of functions that instead of following an absolute goal, form a sequence of actions, also called a deployment plan, which has a high expected utility. This plan can be re-adapted during execution in order to maximize the utility. One representative of this category are SON verification strategies. A verification approach is seen as a three step process during which we partition the network into sets of cells, trigger an anomaly detection algorithm, and generate undo actions for CM changes causing performance degradation.

However, changing the network topology while having the goal of saving energy may negatively impact the undo planning process. By switching on or off a cell, a verification mechanism may wrongly blame other CM changes, roll them back and even inappropriately re-adapt its deployment plan. To overcome this issue, we present a Steiner tree-based verification solution. A Steiner tree is a Minimum Spanning Tree (MST) whose total costs can be reduced by adding new vertexes to the graph. Those extra nodes are the cells that can be enabled or disabled. Based on the outcome, we are able to state whether a certain anomaly has been caused by such on-demand cells or by other CM changes.

The evaluation of our approach has been carried out in a simulation environment. The results show that the proposed method is able to handle topology changes and prevent regular CM changes from being undone.

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] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2," 3rd Generation Partnership Project (3GPP), Technical Specification 36.300 V13.2, Jan. 2016.

[3] C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[4] 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.

[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] S. Nováczki, T. Tsvetkov, H. Sanneck, and S. Mwanje, "A Scoring Method for the Verification of Configuration Changes in Self-Organizing Networks," in *7th International Conference on Mobile Networks and Management (MONAMI 2015)*, Santander, Spain, Sep. 2015.

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

[8] 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.

[9] 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.

[10] T. Kohonen, "Self-organized formation of topologically correct feature maps," in *Proceedings of Biological Cybernetics*, vol. 69, 1982, pp. 43–59.

[11] R. A. Brualdi, *Introductory Combinatorics (5th ed.)*. Pearson Prentice Hall, 2009, iSBN 978-0-13-602040-0.

[12] 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 (NOMS 2016)*, Istanbul, Turkey, Apr. 2016.

[13] 3GPP, "Telecommunication management; Study on Energy Savings Management (ESM)," 3rd Generation Partnership Project (3GPP), Technical Specification 32.826 V10.0.0, Apr. 2010.

[14] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel, "Lower Bounds for Approximation Algorithms for the Steiner Tree Problem," in *Lecture Notes in Computer Science*, 2001, pp. 217–228.

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

[16] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, Jun. 1981.

[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] ——, "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.

[20] D. Freedman, R. Pisani, and R. Purves, *Statistics*, ser. International student edition. W.W. Norton & Company, 2007.

[21] T. Bandh, R. Romeikat, and H. Sanneck, "Policy-based coordination and management of SON functions," in *12th IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM 2011) Work.* Dublin, Ireland: IEEE, May 2011, pp. 827–840.

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

[23] T. Kürner, M. Amirijoo, I. Balan, H. van den Berg, A. Eisenblätter *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.

[24] P. Szilágyi and S. Nováczki, "An Automatic Detection and Diagnosis Framework for Mobile Communication Systems," *IEEE Trans. Netw. Serv. Manag.*, vol. 9, no. 2, pp. 184–197, Jun. 2012.

[25] S. Nováczki and B. Gajic, *Engineering Applications of Neural Networks: 16th International Conference, EANN 2015*. Springer International Publishing, Sep. 2015, ch. Fixed-Resolution Growing Neural Gas for Clustering the Mobile Networks Data, pp. 181–191.