A Constraint Optimization-Based Resolution of Verification Collisions in Self-Organizing Networks

Tsvetko Tsvetkov and Georg Carle
Department of Computer Science
Technische Universität München
{tsvetko.tsvetkov, carle}@in.tum.de

Christoph Frenzel
Department of Computer Science
University of Augsburg
frenzel@informatik.uni-augsburg.de

Henning Sanneck
Nokia
Munich, Germany
henning.sanneck@nokia.com

***Abstract*—The verification of Configuration Management (CM) changes is an important step in the operation of a Self-Organizing Network (SON). In order to perform its tasks, a verification mechanism makes use of an observation and a correction time window. In the first window it assesses the impact of deployed CM changes by monitoring the network's Performance Management (PM) data. Furthermore, it partitions the network in one or more verification areas, detects anomalies within them, and generates CM undo requests, each having the purpose to set CM parameters to some previous state. In the second window it deploys those requests to the network.**

However, two or more verification areas might be overlapping and share anomalous cells. As a consequence, we have verification collisions preventing two or more generated CM undo requests to be deployed at same time. Thereby, the verification mechanism might not be able to deploy all generated CM undo actions for the given correction window. In this paper, we propose a method that makes use of constraint optimization techniques to identify which requests can be merged together in order to meet the time requirement. We achieve our goal by using constraint softening based on so-called performance rating values of the requests. We evaluate our method in two different scenarios. First, we highlight the need for handling verification collisions by observing CM and PM data of a real Long Term Evolution (LTE) network. Second, a simulation study shows the ability of our method to keep the network performance at a high level.

I. Introduction

Today, operators of mobile communication systems require a convenient way of configuring and optimizing their network. With the ever-increasing adoption of mobile communication services by users, the need to efficiently manage the deployed Network Elements (NEs) and guarantee a high degree of reliability increases as well. As a result, the Self-Organizing Network (SON) concept has been introduced to cope with the complex nature of standards like Long Term Evolution (LTE), LTE-Advanced [1], and even the next major 5G release [2]. It specifies features that automatically optimize the network, auto-configure newly deployed NEs, and deal with fault detection and resolution. Such features are implemented by SON functions which typically resemble control loops that observe Performance Management (PM) and Fault Management (FM) data generated by the network and, based on the type of tasks they are designed for, adjust Configuration Management (CM) parameters. An example for such a function is Coverage and Capacity Optimization (CCO) which physically changes the cell area by adjusting the antenna tilt or the transmission power.

However, the ability to perform auto-configuration and auto-optimization yields the need to verify the performance impact of deployed CM changes. As it is typically the case, every decision a function or the human operator makes has certain positive or negative consequences. For instance, if we inappropriately change the physical borders within a cell by adjusting the antenna tilt, we might negatively impact the handover performance to all neighboring cells as well. What is even worse is that if we do not detect this inappropriate change in time, we might negatively influence all upcoming decisions by any function that is monitoring the cell. Therefore, verification mechanisms have been developed to assess CM changes deployed to the network. Such a mechanism is typically seen as a separate SON function that implements a special type of anomaly detection, also known as the verification process. The outcome of this process is to either accept deployed CM changes or to revert some of them to some previous state, also known as a CM undo request or undo request for short.

Nevertheless, rolling back some actions is certainly not a straightforward procedure. For example, if we have an anomalously performing cell and spot a transmission power change at one neighbor and an antenna tilt adjustment at the other, we will not be able to simultaneously undo them since those requests would be in a so-called verification collision. Such a collision is an indicator for an uncertainty which change to revert at first and which to possibly omit. In this case, there are two possible approaches that we can follow. The first one is to completely neglect those collisions and undo both changes. The major drawback we have here is that we might undo changes that did not harm the performance of the network. The second one is to do a stepwise undo based on some criteria like the execution time of the changes or the overall degradation within the area around the cells that have been reconfigured. However, by following the latter approach we face the problem that we do not have an unlimited amount of time to test out those undo requests. One example is the arrival of peak hours. We simply do not want to rollback changes as we have a high demand on reliability and performance. Furthermore, every time we postpone the execution of the correct undo requests we face the risk other active functions to interfere and hinder the verification mechanism to achieve its goal.

In this paper we propose a verification approach that addresses those types of issues. It depicts the undo requests in a collision graph and searches for those that cannot be executed in time. Then, it makes use of constraint optimization techniques that allow defining soft constraints, i.e., rules that might get violated based on a certain priority. Based on the

selection which constraints to violate, it performs a grouping of undo requests so that the given time requirement can be met. Those grouped undo requests are considered as one entity as we deploy them to the network.

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

II. The Verification Process

A. Background

The verification process operates in three phases: (1) it partitions the network into verification (observation) areas according to the CM changes, (2) runs an anomaly detection algorithm for each area, and (3) marks the changes for an undo that are most likely responsible for causing an abnormal behavior. Finally, it schedules the undo requests for deployment.

Typically, a verification area is comprised of the reconfigured cell, also called the *target cell*, and a *target extension set* that includes the possibly impacted cells by that change. A typical approach to compute such areas is to consider the neighbor relations between the cells, e.g., by taking the reconfigured cell and all of its direct neighbors. Note that two cells are considered as neighbors when a handover of User Equipments (UEs) between them can be established. Another approach is to directly observe the impact areas [1] of the SON functions that have been recently active in the network. In addition, the selection of the cells that are being under observation may depend on their location, e.g., areas of dense traffic and known trouble spots [3].

The purpose of an anomaly detection algorithm is to observe the areas for unexpected performance behavior. In literature, several approaches have been presented. For instance, in [4] topic modeling is applied to the PM data from all cells within an area leading to the computation of topic model clusters that can be seen as indicators of the network state. Depending on the semantic interpretation, those clusters are further classified as either normal or abnormal. In [5] a technique is presented that is based on an extended version of the incremental clustering algorithm Growing Neural Gas (GNG), known as Merge Growing Neural Gas (MGNG). The proposed algorithm focuses on the capturing of input data behavior by taking the history data into account. The approach allows the learning of common sequences of input data and the prediction of their future values.

In case of an anomalously performing area, an undo request is generated. Such a message has three data fields: one that uniquely identifies the target cell in the network, another that consists of the identifiers of the impacted cells, and a third field that includes a list of CM parameter values for the target cell. The latter one can be a complete snapshot consisting of all CM settings of a cell made at some earlier point in time. It can also be a partial list that includes only a certain number of CM settings. For example, if we know that a function like Mobility Robustness Optimization (MRO) has been active, whose purpose is to optimize the handover between two neighboring cells, and spot at the same time a degradation of handover performance, we might rollback only the changes made by that function.

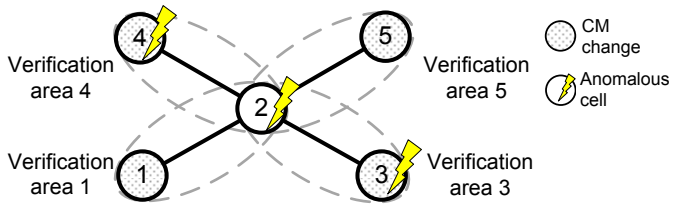


Figure 1. Verification collision example

The generated undo requests are then put on a queue which has to be processed over a certain time period. However, before deploying them to the network, the queue is scanned for verification collisions. A collision occurs when at least two verification areas share anomalous cells, i.e., the corresponding two undo requests impact two overlapping sets of cells. The conflict-free undo requests are grouped together, also called *conflict groups*, and scheduled based on a certain criteria.

B. Problem description

The verification process requires two time windows, each partitioned into one or more time slots, to achieve its goal. In the first window, also called the observation window, the performance impact of the deployed CM changes is assessed. In other words, during this time frame we collect PM data from the network and trigger the anomaly detection algorithm for the given verification areas. The second window, known as the correction window, is dedicated for deploying the undo requests for the abnormal performing areas. Unfortunately, the presence of undo requests that impact overlapping sets of cells delays the process of deployment as we are going to show in the following example.

Suppose that we have five cells, as shown in Figure 1, and three of them have degraded after deploying four CM changes. If a verification area is comprised of the reconfigured cell and its direct neighbors, we will have four areas, namely 1, 3, 4 and 5. For each of those areas an undo request is generated. Note that the areas and the undo requests are taking the index of the target cell. Furthermore, let us assume that the correction window consists of two time slots. The first question that arises here is how to group those requests so that we can deploy the undo requests in time. For example, we might group the undo requests based on the type of CM changes they are reverting. If in the presented example, the antenna tilt of cell 1 and 3 has been modified whereas cell 4 and 5 have been optimized for load balancing, we might group them in that way.

A second question is how to minimize the probability of undoing positive CM changes. For instance, in the example outlined above we might want to group the undo requests for area 3 and 4 and execute them at first since they include the highest number of degraded cells. In other words, they are more likely to be responsible for the abnormal network performance.

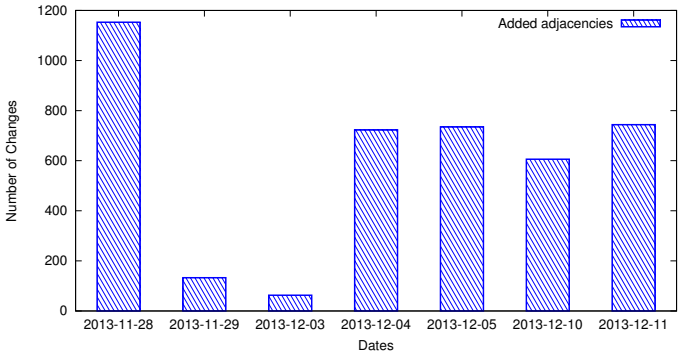


Figure 2. Number of added cell adjacency objects

C. Reasons for verification collisions

There are numerous reasons for having a high number of undo requests that are in collision and, therefore, not deployable for the given time. The first one is the location of the verification mechanism. Since it needs to have a wide view of the network and all ongoing CM changes, it 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 the CM changes and, therefore, it will have a coarse-granular view of the CM data. As a consequence, it will analyze a large set of CM changes which can make the verification process more complicated since it creates a potential for verification collisions. In Figure 2 we show what a verification mechanism would actually need to analyze if it is supplied with CM data once a day. The figure shows the daily number of changed adjacencies between network cells, which are needed for handover. The results originate from a real LTE network consisting of 3028 cells. As it can be seen, the verification mechanism would be required to assess seven CM change sets, five of which contain a high number of modifications.

The second reason is the cell size and the resulting cell adjacencies. As in every new technology, the size of a cell shrinks which, as a consequence, leads to more neighboring cells, including those from other technologies towards which handovers are possible. Moreover, if we consider 5G, base stations will start having a range similarly to commonly used Wi-Fi routers [6]. Consequently, verification areas will get larger and include more entities requiring verification, which creates a potential for more area overlaps and verification collisions.

The third reason comes from the way mobile communication networks are optimized today. What is often used is offline optimization, i.e., waiting until all the required PM data is collected, running an optimization algorithm, and simultaneously deploying the recommended changes, thus, leading to a CM change pattern similar to the one presented above. As stated in [7], offline SON solutions have the advantage that they are comprised of sophisticated optimization tools. Such tools can test a high number of CM settings and search the complete optimization space. However, the optimization steps are simulated which requires detailed knowledge like the user locations or the received signal strength for all possible tilt values. Such information is simply difficult to collect.

III. Concept

The concept is split in three phases. During the first phase, we identify the sets of undo requests that are not deployable for the given correction window. During the second phase we trigger the merging process by softening some of the constraints violating the time requirement. The last phase addresses the deployment of the requests.

A. Clique group selection

The set of all undo requests is represented as an undirected graph $G = (V, E)$ which comprises of a set V of vertexes and a set E of edges. The set V represents the requests whereas E the set of verification collisions $V \times V$. Two vertexes are connected with each other if and only if the corresponding undo requests are not allowed to be simultaneously deployed. In order to find undo requests that cannot be scheduled without violating a constraint, we start looking for cliques. A clique L is a subset of the vertexes of the graph G (i.e., $L \subseteq V$) such that every two distinct vertexes $v_1, v_2 \in L$ are adjacent in G, i.e., $(v_1, v_2) \in E$. A clique also induces a complete subgraph of G which means that all the undo requests of a clique are in a collision with each other. Furthermore, we are interested in those only that have more than a certain number of vertexes. If we denote the number of available time slots of the correction window as τ, we select the cliques that have more than τ vertexes upon all maximal cliques. Note that maximal cliques are ones that cannot be enlarged. The undo requests grouped in such cliques cannot be deployed in time since we have more than τ requests, each in collision with the other.

An example of how this process may look like is given in Figure 3. The presented network consists of 12 cells and 18 cell adjacencies. In addition, we have six reconfigured cells, 1, 2, 3, 6 , 10, and 11, as well as four degradations for cells 2, 3, 4, and 12. Furthermore, let us assume that the number of available time slots τ equals 2 and that the verification area is computed by taking the reconfigured cell and all of its direct neighbors. Note that the areas as well as the undo requests v_i take the index of the reconfigured cell. As a consequence, we have the following collision pairs: (v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_6), (v_3, v_6), and (v_{10}, v_{11}). Since we have a time slot limitation of 2, we start searching for maximal cliques consisting of at least 3 vertexes. In the presented example we have two cliques fulfilling this requirement: one comprised of v_1, v_2, v_3, and another including v_2, v_3, v_6.

The next step is to construct an undirected clique graph $G_c = (V_c, E_c)$, where V_c is the set of all cliques and E_c the set of all edges $V_c \times V_c$. Two cliques are connected with each other if and only if they have common vertexes. Such cliques we refer to as *clique groups*. In the presented example, such a group would be comprised of the two detected cliques. Each clique group is seen as one entity, used as input for the next phase.

B. Clique group size reduction

A clique group gives us a set of undo requests that cannot be scheduled for the given time. Thus, we start reducing the size of a group by merging some of the undo requests. To do so, we define at first a variable assignment function

(Equation 1) that associates a variable x_i to an undo request v_i. The range of a variable x_i is set to $[1, \tau]$.

$$\omega \colon V \to X \tag{1}$$

Then, we specify a set C of collisions $V \times V$ which we address as soft constraints. As known from constraint optimization, soft constraints are such that might under certain circumstances be violated [8]. Every two vertexes connected in the clique group are added to that set. Furthermore, soft constraints need to be prioritized in order to define how important they are for us to be satisfied. Thus, we make use of a rating function φ, as defined in Equation 2. This function computes a rating value based on the performance of the target cell as well as the impacted cells of the associated undo requests. Note that the higher the value is, the more important it is the constraint to be satisfied.

$$\varphi \colon C \to \mathbb{R} \tag{2}$$

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

$$\max \sum_{(v_1, v_2) \in C} \varphi(v_1, v_2) r(\omega(v_1), \omega(v_2)) \tag{3}$$

subject to

$$r(x_1, x_2) = \begin{cases} 0 & \text{iff } x_1 = x_2 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

As known from constraint optimization, Equation 4 gives us a reification function r that tells us whether the inequality of two variables x_1 and x_2 is satisfied. Equation 3 defines the objective, i.e., the maximization of the weighted sum of all soft constraints, based on which we reduce the size of a clique group.

Let us briefly describe what the optimization actually does. Every edge in a clique group is defined as a soft constraint that is rated with a certain priority. The better the performance of the cells of the given undo requests are, the more important the constraint becomes. By maximizing the weighted sum, the undo requests for the worst performing areas are merged together, i.e., the variables of those requests receive the same value. This is also why the variables need to be bound by the number of available slots τ. It simply gives us the number of maximum allowed vertexes in a clique group.

If we recall the example from Figure 3, each of requests within the clique group would be assigned to a variable x_i and the constraints $x_1 \neq x_2$, $x_1 \neq x_3$, $x_2 \neq x_3$, $x_2 \neq x_6$, and $x_3 \neq x_6$ would be used. One possible outcome of the maximization is that variables x_2, x_3, and x_6 receive the same value.

The undo requests whose variables receive the same value are merged together. This process is called edge contraction, i.e., for every two merged requests, an edge $e \in E$ is removed and its two incident vertexes $v_i, v_j \in V$, are merged into a new vertex v'_k, where the edges incident to v'_k each correspond to an edge incident to either v_i or v_j. This procedure gives us a new undirected graph $G' = (V', E')$ that does not have the edges between requests that have been merged. In the presented example, v_2, v_3, and v_6 are grouped together, leading to a new graph G' consisting of four vertexes v'_1, $v'_{2,3,6}$, v'_{10}, and v'_{11}, as well as the edges $(v'_1, v'_{2,3,6})$ and (v'_{10}, v'_{11}).

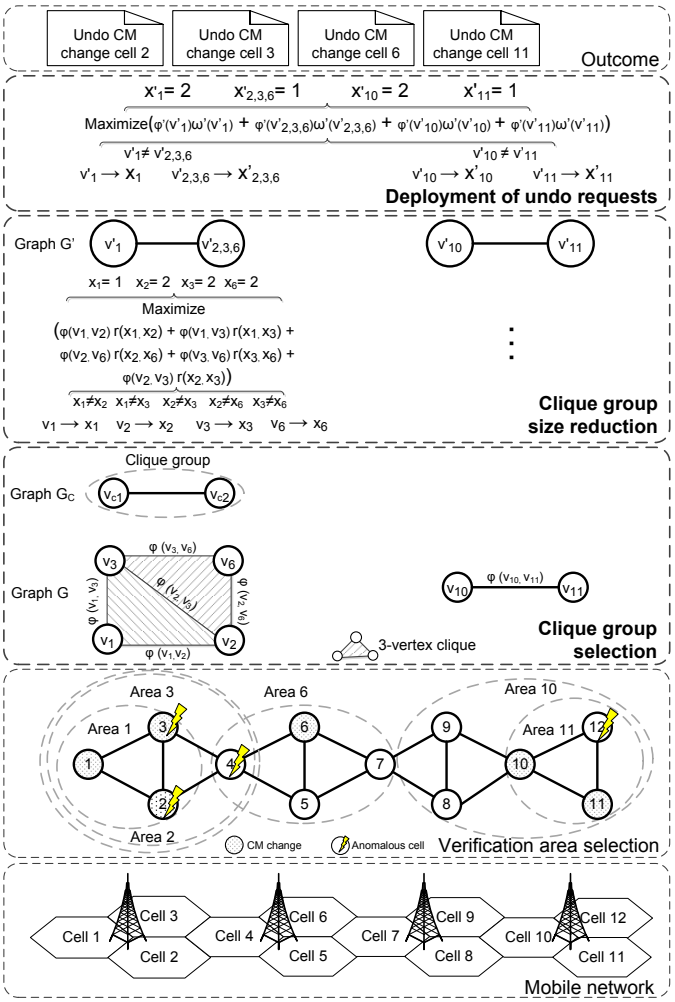


Figure 3. Example of the concept for a time slot limit of two

C. Deployment of undo requests

In the last step we reduced the size of every clique group by merging some of the undo requests. Now we attempt to find out which undo requests to deploy at first for the graph G'. Similarly to the second step, we define a variable assignment function, as follows:

$$\omega' \colon V' \to X' \tag{5}$$

The variables x'_k take values between 1 and τ. Similarly to φ, we make use of a performance rating function, however, for the nodes of G'. It is specified as follows:

$$\varphi' \colon V' \to \mathbb{R} \tag{6}$$

The deployment itself is defined as a constraint optimization problem, as follows:

$$\max \sum_{v' \in V'} \varphi'(v') \omega'(v') \tag{7}$$

subject to

$$\forall (v'_1, v'_2) \in E' : \omega'(v'_1) \neq \omega'(v'_2) \tag{8}$$

In Equation 8 we say that the variables of every two vertexes connected in G' must not receive the same value. Therefore,

the set of edges E' can be also seen as a set of hard constraints that must not be violated. The objective itself is defined in Equation 7, which similarly to the previous step is a maximization of the weighted sum. Thereby, the lower the rating of an undo request v'_k is, the lower the value of the corresponding variable x'_k would be. The undo requests associated with the variable receiving the lowest value are deployed at first.

Recalling the example from before, four variables, namely x'_1, $x'_{2,3,6}$, x'_{10}, and x'_{11}, are generated. The constraints are $x'_1 \neq x'_{2,3,6}$ and $x'_{10} \neq x'_{11}$. Each variable is multiplied with the outcome of φ' and the resulting weighted sum is maximized. One permissible outcome is undoing the change of cell 2, 3, 6, and 11.

Should the network still show an anomalous behavior after deploying the undo requests, we trigger the whole process again with a decreased τ since we have consumed one time slot for deploying the first set of undo requests.

IV. Evaluation

A. SON verification function

The SON verification function is based on our work outlined in [9], and has been extended with the method introduced in Section III. In order to determine whether a verification area is abnormally performing, it makes use of two metrics. One is the *Key Performance Indicator (KPI) anomaly level* which depicts the deviation of a KPI from its expectation. To compute it, we require a training phase during which we collect samples $k_1 \dots k_t$ for each KPI (t marks a training period). During this particular phase the network has to show an expected behavior. Furthermore, the duration of a training period depends on the granularity for gathering KPI data from the network. Then, we standardize the gathered data by computing the z-score of each data point $k_1 \dots k_t, k_{t+1}$. Here, k_{t+1} corresponds to the current sample that we are going to observe. The anomaly level of a KPI corresponds to the z-score of k_{t+1}.

The other metric is the *cell level* that depicts the performance state of a cell. We define it as the weighted sum of the KPI anomaly levels. The weights range in the interval $[0, 1]$ and the sum of all weights equals to 1. An area is considered as degraded if the average cell level falls in the range $(-\infty; -2.0]$ for success KPIs or $[2.0; \infty)$ for failure KPIs.

Furthermore, φ and φ' compute the performance rating by taking the average cell level of the considered cells in case of success KPIs or the negative average cell level in case of failure KPIs.

B. Real data study

1) Environment: The real network data set we are using consists of PM and CM data dumps generated by the LTE network mentioned in Section II. The observation days are the same as the ones listed in Figure 2. Moreover, we know that two SON functions have been actively performing changes during this time period: the Automatic Neighbor Relation (ANR) and the Physical Cell Identity (PCI) allocation function, as defined in [1]. The list of KPIs has been exported on hourly basis, as stated in [10]. The CM changes have been obtained from the history database once every day for the given time period. In addition, a verification area comprises of the cells of the reconfigured adjacency.

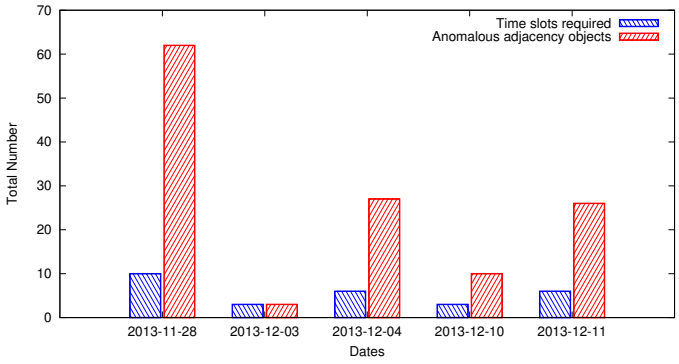
2) Results: In Figure 2 we have outlined the number of adjacency managed objects that have been added for the given time period. What we are interested in, are the cases where we have a CM change followed by an abnormal KPI value for two consecutive hours. Figure 4(a) shows the number of adjacency objects that have negatively impacted a KPI, in particular, the retransmission rate of Radio Link Control (RLC) Protocol Data Units (PDUs). In addition, the figure outlines the number of time slots, each with the length of one hour, that are required if we attempt to undo those changes. The question that arises here is what will happen when we halve the maximum number of required time slots and try then to deploy the undo requests. Note that the maximum number of 10 time slots comes from the day requiring the most ones, which in this case is the 28^{th} of November. As it can be seen, we cannot meet the requirement of 5 time slots for three days: the 28^{th} of November, as well as the 4^{th} and 11^{th} of December. However, by using our approach we are able to find a solution and assign all requests to one of the five time slots. The observation is presented in Figure 4(b). The undo requests associated with time slot 1 are the first marked for deployment.

C. Simulation study

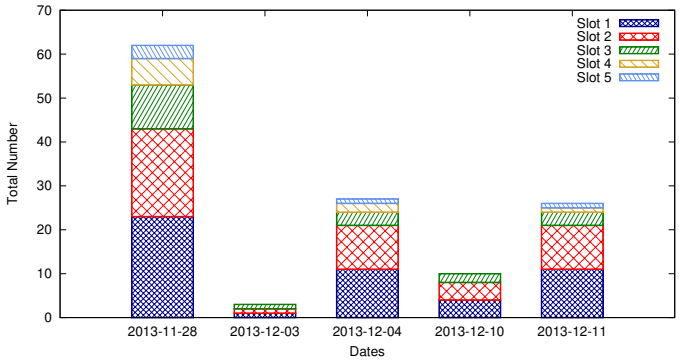
1) Environment: Our simulation environment, also called the SON Simulation System (S3), consists of an LTE radio network simulator, a set of SON functions, and a SON coordinator, as presented in [9].

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

For the SON verification function, t is set to a simulation round. The required training data is collected during a separate test run, lasting 70 rounds, where optimal network settings are used. In addition, the KPIs we consider for calculating the cell level are the Handover Success Rate (HOSR) and Channel Quality Indicator (CQI), each with a weighting factor of 0.5. Note that the CQI is computed as the weighted harmonic mean of the CQI channel efficiency. The required efficiency values

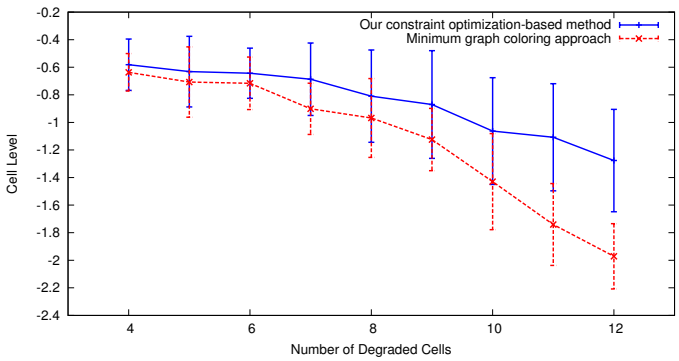


(a) Minimial number of slots required for deployment

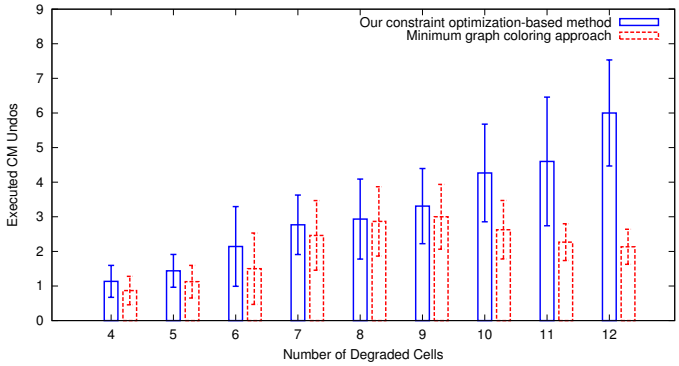


(b) Undo request distribution for a correction window of five slots

Figure 4. Results of the real data study



(a) Cell level result



(b) Number of excuted undo requests

Figure 5. Results of the simulation study

are listed in [12]. Moreover, a verification area is computed by selecting the reconfigured cell and all of its direct neighbors.

2) Results: In the simulation part of our study we observe how the network performance behaves when the suggested undo actions are deployed. We are interested in the average network cell level and the number of executed undo requests as the number of degraded cells increases. Furthermore, it is of particular interest for us to monitor how our method behaves compared to minimum graph coloring [13], a typical constraint satisfaction approach without any constraint softening. The latter approach deploys the undo requests based on the color they receive, starting from those getting the most frequently used one.

To perform the evaluation, we select for every test run a fixed number of cells for degradation based on an uniform distribution. The number of degraded cells ranges between 4 and 12. The degradation is done by deploying an untypical configuration for the used network scenario: an antenna tilt of 3 degrees and a transmission power of 42 dBm. The number of available time slots τ is set to 2 due to the small network size. 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 cell levels reported by all 32 cells from the last simulation round.

The results are shown in Figure 5(a). The shown 95 % confidence intervals are computed around the sample mean of 13 consecutive test runs. As the figure outlines, our method is able to provide a better cell level compared the minimum graph coloring approach. 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 approach. This trend can be seen in Figure 5(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 our method to start merging some of 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. An additional reason for this decrease is the activity of some SON functions in the regions where undo requests have been delayed. At the time they saw the degradation, RET and TXP tried to deploy a CM configuration which did not solve the problem. Thereby, more undo requests requiring deployment have been generated which effectively delayed the whole verification process.

V. Related Work

The concept of pre-action SON coordination [1], [14] can be considered as an alternative approach to the concept we have introduced. It can be seen as a pessimistic strategy that defines rules used to detect and resolve *known* conflicts between running SON functions. Instead of evaluating the performance after each round of deployed CM changes to the network, it prevents conflicting functions from getting active.

For instance, SON coordination rules prevent the simultaneous operation on shared CM parameters within the same area. In addition, it blocks the activity of one SON function if it affects the input measurements of another one. This information, however, is known beforehand.

The idea of rolling back changes based on the outcome of an anomaly detection process has also been introduced. Heckerman et al. [15] developed a procedure that tries to determine not only the most likely cause for a malfunctioning device, but to assemble an action plan for repair. This optimal troubleshooting plan is a sequence of observations and repairs that at the same time minimizes the expected costs. The authors make use of decision theory as well as Bayesian networks for generating the plan. Their work, however, presents the concept from a theoretical point of view.

The idea of using constraint optimization for deploying scheduling jobs has, e.g., been outlined in [16]. The authors propose generalizations of non-preemptive constraint propagation techniques to preemptive ones. Furthermore, they present the concept of generalizing those techniques to so-called mixed problems which allow certain activities to be interrupted whereas others remain non-preemptive. This work, though, gives only a theoretical overview of the concept.

Another example of constraint-based scheduling is given in [17]. In this work, the author gives an overview of generic constraint programming techniques for modeling and solving scheduling problems. He discusses the different scheduling variations as well as the faced challenges while realizing real-time scheduling for embedded devices. Furthermore, a concrete example is given how the printer paper path control, i.e., matching sheets with their respective images, can be improved.

VI. Conclusion

The process of verifying Configuration Management (CM) changes in a mobile Self-Organizing Network (SON) has been understood as a three step process. During the first step, the network is partitioned into verification areas, also referred to as observation areas. Those areas are comprised of a set of cells that are monitored due to a certain activity that has taken place, like a change of a CM parameter. During the second step an anomaly detection algorithm is used to determine whether a given verification area is performing according the expectations. Then, during the third step an attempt is made to assemble a new configuration that returns the badly performing cells to some previous state. Such a configuration is called a CM undo request.

However, CM undo requests can be in conflict with each other. Typically, this happens when two requests are in a verification collision which means that verification areas, for which the undo requests have been generated, share anomalous cells. As a consequence such requests must not be scheduled at the same time, i.e., they have to be deployed in different time slots. However, the time that is available for deploying them is limited which can prevent the verification mechanism from achieving its goal. What we have proposed in this paper is a concept that allows us to merge some undo requests in order to deploy them in time. Our method depicts the set of undo requests as a collision graph and divides it in cliques. Those cliques consist of undo requests that cannot be scheduled by the given time constraint. Then, edge contraction within the cliques is made which is based on constraint softening. As a result, some nodes in the graph are merged together which provides us a solution how to perform the deployment.

The evaluation of our method is twofold. On the one side, we show how conflicting CM undo requests and verification collisions can occur in a real Long Term Evolution (LTE) network. On the other side, we compare our method in a simulated environment with an approach that uses minimal graph coloring, i.e., a strategy that does not perform constraint transformation. The results show that our approach is able to let more CM undo requests through and return the network to the expected performance state.

References

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

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

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

[4] 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.

[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 (CogMan 2015)*, Ottawa, Canada, May 2015.

[6] S. Chen and J. Zhao, "The Requirements, Challenges, and Technologies for 5G of Terrestrial Mobile Telecommunication," *Communications Magazine*, Jan. 2015.

[7] S. Berger, A. Fehske, P. Zanier, I. Viering, and G. Fettweis, "Comparing Online and Offline SON Solutions for Concurrent Capacity and Coverage Optimization," in *Proceedings of the Vehicular Technology Conference (VTC Fall)*, Vancouver, Canada, Sep. 2014.

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

[9] T. Tsvetkov, S. Nováczki, H. Sanneck, and G. Carle, "A Post-Action Verification Approach for Automatic Configuration Parameter Changes in Self-Organizing Networks," in *6th International Conference on Mobile Networks and Management (MONAMI 2014)*, Würzburg, Germany, Sep. 2014.

[10] ——, "A Configuration Management Assessment Method for SON Verification," in *International Workshop on Self-Organizing Networks (IWSON 2014)*, Barcelona, Spain, Aug. 2014.

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

[12] 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.

[13] E. W. Weisstein, "Minimum Vertex Coloring From MathWorld - A Wolfram Web Resource," Feb. 2015, http://mathworld.wolfram.com/MinimumVertexColoring.html.

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

[15] D. Heckerman, J. S. Breese, and K. Rommelse, "Decision-Theoretic Troubleshooting," *Communications of the ACM*, vol. 38, pp. 49–57, Mar. 1995.

[16] C. L. Pape and P. Baptiste, "Resource Constraints for Preemptive Jobshop Scheduling," *Constraints*, vol. 3, no. 4, pp. 263 – 287, Oct. 1998.

[17] M. P. Fromherz, "Constraint-based Scheduling," in *IEEE American Control Conference*, Arlington, VA, Jun. 2001.