# SON Function Performance Prediction in a Cognitive SON Management System

Simon Lohmüller[1], Fabian Rabe[1], Andrea Fendt[1], Bernhard Bauer[1], Lars Christoph Schmelz[2]
[1] Universität Augsburg, Department of Computer Science, Augsburg, Germany
{simon.lohmueller, fabian.rabe, andrea.fendt, bernhard.bauer}@informatik.uni-augsburg.de
[2] Nokia, Network Management and Automation, Munich, Germany
christoph.schmelz@nokia.com

***Abstract*—As a reply to the increasing demand for fast mobile network connections the concept of Self-Organising Networks (SONs) has been developed, reducing the need for humans to execute Operation, Administration and Maintenance (OAM) tasks for mobile networks. However, a SON contains functions which are provided by different vendors as black boxes, making it hard to predict the performance of the network, especially under untested configurations. Since Mobile Network Operators (MNOs) have to fulfil rising mobile network performance demands while reducing costs at the same time, it is crucial to gain a better understanding of the network behaviour to allow a cost-neutral performance improvement while simultaneously reducing the risk of network misconfiguration and service disturbance. In this paper an approach is introduced to enhance SON Management models with cognitive Machine Learning (ML) methods. Therefore, the simulated behaviour of three different SON Functions is analysed and described by a Linear Regression (LR) Model. In a second step, performance data of network cells are analysed for similarities using k-Means Clustering. The findings of these two steps are then combined by fitting the models onto smaller clusters of cells. Finally, the utility of these models for predicting the performance of the network is evaluated and the different stages of refinement are compared with each other.**

## I. Introduction

In addition to an increasing demand for mobile internet connections, the Internet of Things is creating a whole new category of devices demanding reliable and capable Radio Access Technologies (RATs) [1]. However, prices per megabyte are actually going down due to fierce competition among Mobile Network Operators (MNOs). This puts high pressure on MNOs to reduce expenses to remain profitable. To meet the expected demand, MNOs are forced to invest into their infrastructure and will inevitably be faced with high Capital Expenditures (CAPEX). Currently, the operation of a network is still largely based on a centralised MNO approach, executed by human operators. Therefore, the focus lies on limiting Operational Expenditures (OPEX) [2] which can be achieved by automating the time-consuming, expensive and error-prone tasks manually executed by human MNOs [1].

Modern cellular networks are large systems consisting of countless separate cells covering a wide variety of different environments. These environments differ in multiple domains such as their geographical layout or the motion profile of each user. To enable a satisfactory user experience it is necessary to configure cells in accordance to requirements posed by the environment. This can be achieved by changing the so called Network Configuration Parameters (NCPs), e.g., increasing the transmission power or changing the tilt of an antenna. However, the Operation, Administration and Maintenance (OAM) of such networks is very labour intensive, i.e., to adjust NCPs manually, not mentioning the experience an operator would need to do so. Furthermore, certain aspects of the environment can change over time, e.g., commuters leaving the down town area of a city in the evening and thereby producing a multiple of traffic compared to the rest of the day. This requires a constant readjustment of the parameters, immensely increasing the workload for an MNO. Based on these needs the concept of Self-Organising Network (SON) was developed [1].

In SON, a multitude of SON Functions, i.e., autonomously working closed control loops, monitor the performance of the network in terms of Key Performance Indicators (KPIs) and adapt the NCPs if required. Such SON systems are faced with several problems: First, SON Functions are implemented and sold by different vendors who provide their products as black boxes where the inner workings are unknown to the MNO. Second, these SON Functions usually only aim at one dedicated KPI and hence, several of these functions need to run in parallel to satisfy a variety of MNO's goals. Third, these SON Functions themselves have SON Function Configuration Parameters (SCPs), such as activation thresholds and step-sizes which need to be set. Determining suitable SON Function Configuration Parameter Values (SCVs) for the SCPs is a crucial step to guarantee a satisfactory Quality of Service (QoS) in domains like network coverage, voice quality, and data rates. Finally, different SON Functions may aim at improving different KPIs while adapting the same NCPs and hence may not be achievable together. While SON Coordination [3] tries to resolve such NCP conflicts after their appearance, SON Management tries to find optimal configurations beforehand that do not influence each other negatively. To overcome these problems, several projects, e.g. [4] and [5], have worked on automating the process of finding suitable SON Function configurations. Also the authors in [6], [7] and [8] have constantly amplified the initial SON Management. However, due to the SON Functions being black boxes, the MNO is forced to rely on documentation provided by the vendor of a SON Function which can be provided in the form of lookup tables mapping SCV sets to expected

KPIs - also known as SON Function Model (SFM) [2] [9]. Since each vendor of a SON Function derives these mappings from experiments in his/her own testing environment, there is little guarantee that they hold true in real world applications where the SON Function might face a different environment and work alongside other SON Functions [8]. Testing all SCV sets in the real network is simply not feasible. Furthermore, a situation may occur where none of the tested SCV sets do well in achieving given objectives. In this case it would be helpful to predict the behaviour of untested sets which possibly do better in fulfilling KPI targets. These facts motivate the following Core Questions (CQ):

**Q1** Is it through means of Machine Learning (ML) possible to generate models which accurately map SCVs to KPIs?

**Q2** Is it possible to reliably estimate the performance of unknown SCV sets?

**Q3** Can ML help finding similarly behaving cells, allowing for more tailored models?

There are several use cases in SON where ML is already applied. For instance, [10] proposes a Reinforcement Learning framework for the coordination of two distributed SON Functions and [11] proposes a solution to learn the best configuration for a Mobility Load Balancing (MLB) function. However, all these approaches do not deal with the problem of several SON Functions concurrently aiming at different KPI targets. Therefore, an approach is presented to enhance SON Management with ML capabilities to overcome this problem.

## II. Background

### A. Supervised Learning - Linear Regression (LR)

**Q1** has been whether it is possible to generate models which accurately predict KPIs based on the SCVs of a SON Function. For this paper SON Functions from the area of Self-Optimisation, namely MLB, Mobility Robustness Optimization (MRO) and Coverage and Capacity Optimization (CCO), were simulated with different SCV sets and the resulting KPIs were measured. Since this needs labelled data, meaning it needs to know the true output for given inputs, this is a problem for *Supervised Learning algorithms*. Supervised models are trained on datasets of the form $(X_1, Y_1), (X_2, Y_2), ...$, with $X_i$ representing the input and $Y_i$ the associated output or label. The quality of a model built through Supervised Learning can be checked by comparing the predicted $\hat{Y}$ to the actual $Y$. Once a satisfying model is built, it can then be used to predict the behaviour of the actual system under unseen inputs. Since each KPI is a continuous value and an algorithm is asked to predict continuous results, an effective approach is *LR*.

*1) The Model:* LR takes an input vector $X$ with $n$ features $\{x_1, x_2, ..., x_n\}$ resulting in the following model:

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n = \hat{f}(X) \quad (1)$$

Fitting an LR Model equals estimating the coefficients $\{\beta_0, ..., \beta_n\}$. In order to do so there needs to be a labelled dataset with $k$ observations of the form $\{(X_1, Y_1), (X_2, Y_2), ...\}$, with $k \geq n$, otherwise Eq. 1 can not be solved. If $k = n$, then there is precisely one solution for each $\beta$. If $k > n$, then it is in most cases not possible to find $\beta$s such that each sample can be described with zero residual.

A residual is defined as the difference between a known output $y_i$ and the estimated output $\hat{y_i}$:

$$\epsilon_i = y_i - \hat{y_i} \quad (2)$$

To fit $\hat{f}()$ to the dataset, the least squares method is employed in order to reduce the residuals. Since the model for LR $\hat{f}()$ was defined in Eq. 1, it is now possible to merge the two equations into the following:

$$\begin{aligned} \epsilon_i &= y_i - \hat{f}(X_i) \\ \epsilon_i &= y_i - (\beta_0 - \beta_1 * x_{1,i} - \beta_2 * x_{2,i} - ... - \beta_n * x_{n,i}) \end{aligned} \quad (3)$$

The goal is to determine the set of $\beta_0, \beta_1, ...$ which have the minimal Residual Sum of Squares (RSS) which is defined as the sum over all squares of each available residual:

$$RSS = \epsilon_1^2 + \epsilon_2^2 + ... + \epsilon_k^2 \quad (4)$$

Inserting Eq. 3 into Eq. 4 leads to:

$$RSS = \sum_{i=1}^{k} (y_i - (\beta_0 - \beta_1 * x_{1,i} - ... - \beta_n * x_{n,i}))^2 \quad (5)$$

Since there are more data points $k$ than unknown coefficients $\beta$, the equation has more than one solution. Through derivation and some calculus the optimal set of $\beta$s can be calculated.

*2) Increasing Flexibility:* So far only linear combinations of the different features $\{x_1, x_2, ...\}$ of $X$ have been considered. However, polynomials with higher degrees than 1 as model are thinkable. According to [12] this is still considered to be LR, since the polynomials can just be considered as additional features where $X_{poly}$ now consists of $\{x_1, x_1^2, x_1^3, ..., x_2, x_2^2, ...\}$. This transformation of $X$ enables more complicated models while still retaining the same procedure as with classical LR. In the same vein as adding polynomials of features it is quite reasonable to expect that certain inputs have synergies with each other and influence the output to a different extent than each of them does individually. These so called *interaction terms* (and their polynomials) can then be used to extend the input vector $X$ and as a result lead to an even more flexible model.

*3) Bias vs. Variance:* Bias refers to the error that is introduced by approximating a real-life problem which may be extremely complicated, by a much simpler model. In contrast, variance describes how much a model changes when fit to a different dataset. Both a too biased model and a too flexible model have shown to have flaws when trying to build a reliable model [12], hence bias and variance need to be balanced.

*4) Quality of Fit:* For a comparison of models it is necessary to evaluate the fit of a model to the available data. Metrics to measure the overall error of a model are, e.g., the Mean Absolute Error (MAE) [13], Mean Squared Error (MSE) [14] and Root Mean Squared Error (RMSE) [13], each with their own advantages and drawbacks. In this paper the error will be measured using the *RMSE* (Eq. 6) since it is interpretable in the actual units of the output, thus being easier to understand while still retaining the high penalty for larger residuals [13].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2} \tag{6}$$

After selecting the error metric, different models can be compared and tuned to reduce the error and lead to a model which better captures the behaviour of the underlying system. Nonetheless, searching for a model with zero error is not the way to generate successful predictions. This is due to the fact that all measurements $Y$ are tainted by the error $\epsilon$ when measured: $Y = f(X) + \epsilon$. Since $\epsilon$ is random, it can not be included in the model $\hat{f}()$. Hence, it is logical to split the residuals into

$$|Y - \hat{Y}| = \epsilon_{reducible} + \epsilon_{irreducible} \tag{7}$$

with $\epsilon_{reducible}$ referring to the reducible error induced through wrong models and $\epsilon_{irreducible}$ referring to errors from the random noise in the system.

*5) Training and Testing:* Since it is impossible to know how much of an error is due to $\epsilon_{reducible}$ and how much due to $\epsilon_{irreducible}$ without explicitly knowing $f()$, an extra step is necessary to evaluate the true performance of any given model regarding its original goal: Predicting outputs of unseen inputs and comparing these to the measured outputs. A drawback of the polynomial models of high degrees is that they are able to capture the training data very well, but show an increased amount of jitter in areas without data points. These high variations in $\hat{y}$ lead to big residuals when tested on new data, an effect called *over-fitting*: The model describes the data presented during fitting so well, that it incorporates the $\epsilon_{irreducible}$ in its predictions and fails to capture the underlying $f()$. The solution for this problem is to not fit the model on all available data from the start. Instead, the dataset is split into training data and testing data. The model is then trained by fitting it to the training data, before it is tested against the remaining testing data. All models are then compared on the basis of their performance on the test dataset. Over-fitted models will have small residuals on the training data, but are too sensitive to the noise and fail on the testing data.

### B. Unsupervised Learning - k-Means Clustering

Although there are many approaches to unsupervised learning, for this paper only *Clustering* is relevant, predominantly for **Q3** mentioned in Sec. I: Finding similarly behaving cells. The goal of Clustering algorithms is to examine the similarity of samples based on their features, and then find groups of samples which share a high similarity within the group compared to samples from groups laying outside. These clusters are usually defined through centroids which are virtual data points representing the proto-element of each group. One of the simplest and fastest algorithms for creating clusters is the *k-Means* algorithm. It is important to note that the user needs to decide manually on the number of clusters $k$ when running the algorithm, so even more domain knowledge is necessary for a successful application of this algorithm.

## III. Data Generation

### A. Tooling

In order to evaluate the SON Function performance prediction proposed in this paper, a Long Term Evolution (LTE) network was simulated in System Experience of Advanced SON (SEASON), a simulation engine initially developed by Nokia Siemens Networks [15]. On top of SEASON sits the SON Function Engine (SFE) which is the run-time environment for the SON Functions and acts as an interface to SEASON. For simulation a scenario based on the topology of the Helsinki city centre is used, instantiating buildings as well as the road network. The scenario spans an area of around 60 km$^2$, equipped with 35 cells and populated with up to 2000 mobile users, half of them travelling at the speed limit of the roads and the other half walking by foot. To put the network under increased stress a highway with 1000 very fast moving mobiles has been added to the scenario. Three SON Functions have been evaluated, namely MLB, MRO and CCO where each of them aims at optimising one single KPI, namely the Physical Cell Load (Load), the Handover Ping Pong Rate (PiPo) and the Channel Quality Indicator (CQI). For each simulation run, only one single SON Function is active to prevent both coordination issues and interferences of two SON Functions. After each granularity period, i.e., 90 minutes of simulated time, SEASON reports measured KPIs per cell to the SFE. Each function is thereby configured with the same SCVs, but since each cell is situated in a different environment with varying KPIs, they become active independently and change NCPs of cells individually. For data evaluation Python, in particular the Pandas and Scikit-learn libraries have been used which host all the implementations of the ML algorithms and methods described in Sec. II.

### B. Data Description

Before beginning with the creation of models, it is worth examining the samples gathered through simulation.

*1) Raw Data:* To make the analysis easier, the data is parsed into one data frame per SON Function, each data frame following the structure of Tab. I. Each row represents the measurements from a single cell in one round. Columns CQI, PiPo and Load store the measured KPIs (normalised to a value range of $[0, 1.0]$) for that instance whereas *threshold* and *step-size* hold the respective SCV under which these measurements were collected. Each SCV set is a tuple of threshold and step-size out of 32 possible combinations. Looking at the raw data distribution some interesting properties are noticeable: First, the data shows similar patterns across SON Functions.

TABLE I: Excerpt of the CCO Raw Data

| Round | Cell ID | CQI | PiPo | Load | threshold | step-size |
|---|---|---|---|---|---|---|
| 1 | Cell 1 | 0.641 | 0.024615 | 0.35 | 0.01 | 1.0 |
| | Cell 10 | 0.672 | 0.000027 | 0.94 | 0.01 | 1.0 |
| | Cell 11 | 0.279 | 0.000029 | 0.94 | 0.01 | 1.0 |
| ... | ... | ... | ... | ... | ... | ... |

Although each SON Function optimises only for a single

KPI, the resulting KPIs show similar distributions across all SON Functions. One might expect that the respective KPI of the active SON Function shows a different distribution compared to the cases when the SON was not optimised for this specific KPI. In addition, even within a SON Function the measurements cluster around the same value. One might expect that due to the large differences in tested SCVs, the values show a wider spread across the complete domain, e.g., some configurations having a huge negative impact on the KPI. Also, the values within single configurations often spread over a wide interval. This is a crucial observation since the fit of a model is measured based on the predicted output of $\hat{y}$ compared to the actual output of a sample $y$ (Eq. 2). However, any model will only predict a single value $\hat{y}$ based on the input which is the SCV set. Due to the wide interval of measured $y$s for a single SCV set the irreducible error will always be very large, therefore making accurate predictions for a single cell hard. To sum up the findings so far: Changing SCV sets of a single SON Function appears to have only a small effect on the performance of the network. Additionally, the measured values show a large variance for each configuration which makes it hard to postulate definitive statements such as ”under configuration $X$ the cell will have the KPI $\hat{Y}$”.

*2) Prepared Data:* It was shown above that the irreducible error would be fairly large when predicting the performance of a single cell in a single round, leading to bad models due to the large amount of noise. To combat that noise, the high level of detail (KPI per cell, per round, per SCV set) of the data is forgone and solely the average KPI per SCV set is aggregated, their structure is shown in Tab. II.

TABLE II: Excerpt of the CCO Prepared Data

| threshold | step-size | sample count | CQI variance | CQI average |
|---|---|---|---|---|
| 1.00 | 5.0 | 385.0 | 0.039244 | 0.526753 |
| 0.90 | 5.0 | 385.0 | 0.039113 | 0.522019 |
| ... | ... | ... | ... | ... |
| 0.10 | 2.0 | 1155.0 | 0.035454 | 0.540144 |
| 0.01 | 2.0 | 1155.0 | 0.036258 | 0.547881 |

## IV. SON Function Performance Prediction

### A. Generating a Learned SFM

The goal is to model and describe the causal relationship between the SCV sets and the respective KPI of each SON Function. Referring back to Subsec. II-A, the input consists of threshold and step-size. Depending on the SON Function the output variable is the respective KPI: CQI, PiPo or Load.

*1) Model Construction:* Before building the actual model it is worth investigating the premises of the experiment, in order to avoid common pitfalls and misinterpretations of the results. A fundamental premise is the assumption of *ceteris paribus*. As the simulation is a controlled environment with little noise and the only factors changed are the independent variables of the SCV sets, an actual causal relationship between SCV sets and observed average KPIs can be assumed. To avoid over-fitting, the raw data samples are split into a train and a test set with a relation of $2:1$ before the averaging happens, thereby ensuring that the model selection happens on a different set of data than the model evaluation. In order to answer **Q2** how well the performance of untested SCV sets can be predicted, it is necessary to split the group of configurations into training and testing as well. However, due to the relatively few distinct configurations simulated, *Leave One Out (LOO) Cross Validation* is employed. Let $n$ be the amount of simulated SCV sets. Then the model is trained on $n-1$ configurations and asked to predict the KPI of the $n$-th configuration. This step is repeated for each configuration, therefore leading to $n$ slightly different models with the same degree of flexibility, and $n$ residuals. On these residuals the error metrics can be calculated describing how well one type of model is able to predict the KPIs of unseen SCV sets.

*2) Model Selection:* As explained in Subsec. II-A, the LR Model can be made increasingly more flexible by extending the input vector with polynomials and combinations of the existing features. The potential drawbacks of an overly flexible model are illustrated in Par. II-A3. Each SON Function was fitted with different models mapping threshold $t$ and step-size $s$ to the KPI. The flexibility was increased stepwise from the linear model in Eq. 8 up to the cubic model with cubic interaction terms in Eq. 9, so that for each SON Function 9 different models were created.

$$\widehat{KPI} = \beta_0 + \beta_1 * t + \beta_2 * s \tag{8}$$

$$\begin{aligned}\widehat{KPI} = \beta_0 + \beta_1 * t + \beta_2 * s + \beta_3 * t * s +\\ \beta_4 * t^2 + \beta_5 * s^2 + \beta_6 * (t * s)^2 +\\ \beta_7 * t^3 + \beta_8 * s^3 + \beta_9 * (t * s)^3\end{aligned} \tag{9}$$

Each model is trained on the training data averages and its performance measured against the training labels, following the LOO Cross Validation protocol. Fig. 1 shows how the different models behave with increasing levels of flexibility. Based on these findings the best performing model for each SON Function has been selected (see Tab. III).

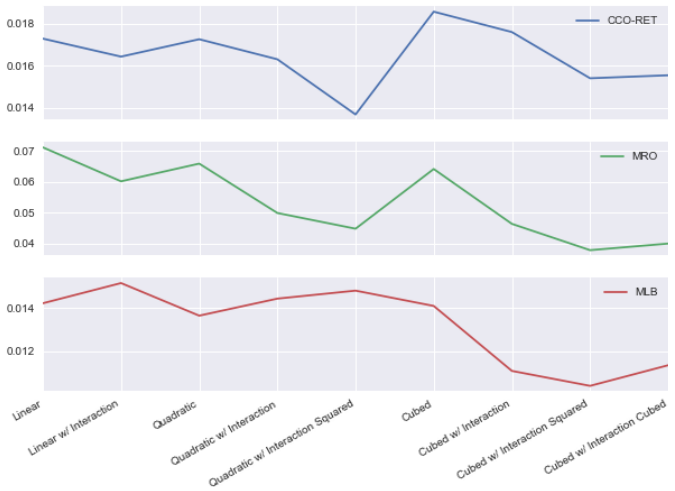

Fig. 1: LOO Validation RMSE for increasingly flexible models

TABLE III: Best Performing Models for each SON Function

| SON Function | Best Model | Training RMSE |
|---|---|---|
| CCO | Quadratic with Interaction Squared | 0.0137 |
| MRO | Cubed with Interaction Squared | 0.0378 |
| MLB | Cubed with Interaction Squared | 0.0104 |

*3) Model Evaluation:* Now that a model for each SON Function has been selected and trained, the actual performance is evaluated on the test data set. Since the focus is again on predicting the performance of unknown SCV sets, the LOO method is applied and the RMSE error calculated.

The **CCO** model has an RMSE of $0.013$. Since the average CQIs only have a maximum deviation of ~$0.07$ this is a high error. For a qualitative evaluation: the model manages to roughly follow the shape represented by the samples and gives a valuable indication for untested configurations to the MNO, but it can not predict each configuration reliably. The **MRO** model has an RMSE of $0.040$. In relation to a maximum difference of ~$0.34$ this model appears to be a better fit than the CCO model. Fig. 2 plots the average PiPo depending on step-size and threshold. The black dots represent the average PiPo of all testing samples. The blue dots show the predictions of the models for this configuration which were trained through the LOO method on all other configurations. For an easier visual understanding of the model behaviour, the blue dots are connected through the semi-transparent surface. The **MLB** model has an RMSE of $0.010$ which is again quite high in relation to the maximum difference of average Load of ~$0.05$. As before, the model reflects the distribution of the average Load pretty well, but does not manage to predict the actual averages reliably in all cases. To sum up, it can be said that each of the models manages to describe the behaviour of the SON Function for different SCV sets. However, the error rates of the models need some improvement, before they can be considered as trustworthy enough to be used in praxis.

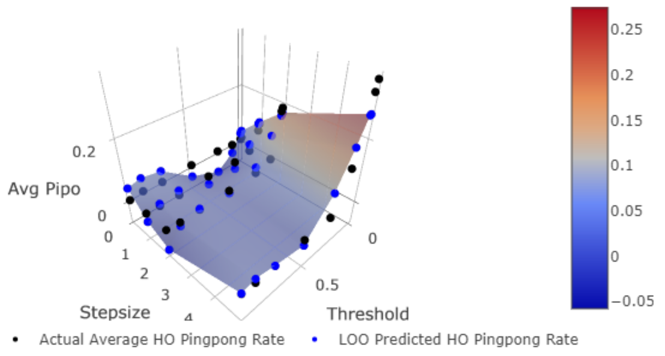

Fig. 2: Results of the MRO model

## B. k-Means Clustering on Network Cells

In [16] an approach is presented where cells are classified based on mobile network context information. However, in this approach the MNO has to do the classification manually by inspection. **Q3** now refers to whether it is possible to automatically divide the group of 35 cells into reasonable groups of similarly behaving cells. The assumption is that those smaller groups might show a more homogeneous behaviour in regard to the relationship between threshold, step-size and the respective KPI. Additionally, clusters might support an MNO to find descriptive features of cells and hence, configure newly deployed cells in a suitable way. On a technical level, Clustering should characterise cells based on their average KPI under various SCV sets and group cells with similar characteristics into the same group, allowing for more tailored LR Models per group.

The cells are split into two groups per SON Function: A cluster of *default* cells and one for *outlier* cells. Therefore, one k-Means Classifier with $k = 2$ is fitted to the dataset of each SON Function. As explained in Subsec. IV-B there is no correct way to determine the fit or quality of a Clustering algorithm due to the unlabelled data. Instead it is up to the user to decide on the utility of the Clustering result. While classifying the cells according to their CQI yields a very clear result (all the turned off cells belong to one class), this offers very little new information. Additionally, the other two SON Functions do not deliver any helpful results either. For both MRO and MLB, one cluster contains cells close to the highway, with the other cluster containing the rest of the cells. But marking each cell based on its classes in the MRO- as well as the MLB-Clustering shows a pattern which offers itself for an easy interpretation: One cluster refers to areas with a high amount of fast moving User Equipments (UEs) (*white* in Tab. IV), a second one to areas with high amount of foot traffic (*blue*), a third one is characterised through a low Load and a low PiPo which refers to turned off cells as well as one more cell (*green*). Finally, the rest of the cells (*unmarked*) belong to the cluster with a lower Load and higher PiPo, the default classification in this scenario. Instead of creating four classes through the merging of two separate classifications, it is also possible to fit a single k-Means Classifier to the complete dataset of MRO, MLB and CCO simulation results: Each cell is now characterised by all SCV to KPI mappings at once. To get a comparable result to the previous merging, $k$ is set to $4$ for this 1-Step Classifier, the results of which can be seen in Fig. 3. The results differ in a few notable places: The 1-Step Clustering process singles out the passive cells perfectly, compared to the merged classifier, which puts cell 8 into the same cluster. Cells close to the highway are put into the same cluster by both classifiers, except for cell 28. Instead, cell 9 is added in case of the 2-Step Classifier. Additionally, the remaining cells are split into two groups, although they appear to be non-distinguishable in their main characteristics, compared to the obvious default cluster created by the 1-Step Classifier. Overall, it appears to be easier to interpret an explanation into the four clusters created by the 2-Step Classifier. Whether it is actually the better classifier depends on whether the MNO can use these classifications for network configuration.

## C. Regression Learning on Clusters

Whilst Clustering provided useful information regarding the similarity of KPI response on different SCV sets, it might also

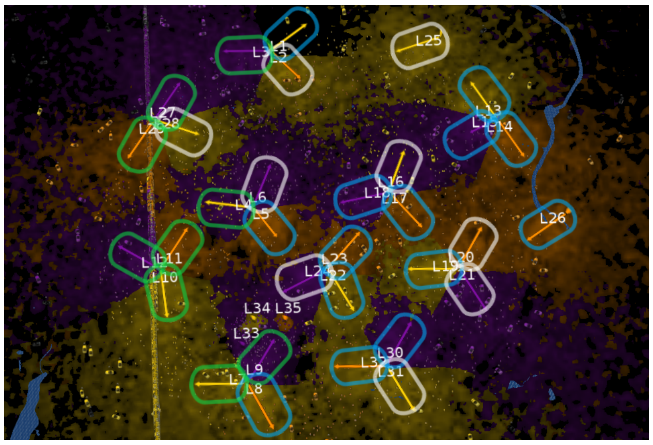

Fig. 3: 1-Step Clustering

TABLE IV: Comparing Un-clustered to Clustered RMSEs

| SON function | Un-clustered | 2-Step Clusters | | | |
|---|---|---|---|---|---|
| | | unmarked | white | blue | green |
| CCO | 0.01321 | 0.0071 | 0.0154 | 0.0139 | 0.0050 |
| MRO | 0.04048 | 0.0463 | 0.0228 | 0.0551 | 0.0115 |
| MLB | 0.01038 | 0.0064 | 0.0228 | 0.0310 | 0.0030 |
| | | **1-Step Clusters** | | | |
| | | blue | white | green | unmarked |
| CCO | 0.01321 | 0.0063 | 0.0093 | 0.0150 | 0.0 |
| MRO | 0.04048 | 0.0473 | 0.0487 | 0.0263 | 0.0 |
| MLB | 0.01038 | 0.0038 | 0.0093 | 0.0129 | 0.0 |

help to improve the Regression Models built in Subsec. IV-A. In this section the best performing models from Par. IV-A2 are fitted onto data from clusters created with the 1-Step and 2-Step Classifiers. For each SON Functions and each cluster a model is trained and evaluated via the LOO method. The resulting RMSEs are presented in Tab. IV. When comparing the un-clustered RMSEs of the SON Functions to the RMSEs of each of the four clusters, it can be observed that, while some clusters have a significantly lower RMSE, there is no case in either Clustering method where all clusters have a lower RMSE compared to the un-clustered scenario. However, looking at the RMSEs of the 1-Step Clusters compared to the un-clustered RMSEs reveals that the RMSEs only get slightly worse for some of the clusters and models but at the same time significantly improves some other RMSEs.

## V. Conclusions and Outlook

At the beginning of this paper **Q1**-**Q3** about the relationship between SCV sets and KPIs have been posed. To answer **Q1** and **Q2**, the best fitting LR Model for each SON Function was determined by evaluation with a separate testing dataset. The results show that the behaviour can be predicted for each of the SON Functions and most of the SCV sets. In order to achieve even more trustworthy predictions on untested SCV sets, other Supervised Learning methods should be evaluated, since it is possible that the underlying relationship between the SCVs and the KPI values is not a linear combination but follows a more chaotic pattern. *Non-Parametric Regression* or *Decision Trees* might be able to capture those characteristics even more accurately and provide a better performance. **Q3** motivated the search for subgroups of cells showing a similar behaviour when it comes to KPI effects. The 35 cells have been classified into 4 similarly behaving clusters. The best performing models for each SON Function were trained on data from each of the clusters. Thereby, the 1-Step Classifier taking all KPIs into account, provided the best results. A bigger scenario with a more heterogeneous network including, e.g., rural areas, might lead to stronger contrasts among the different cells and hence, absolute improvements when fitting models onto the subgroups. Also choosing a higher number of clusters and using more clustering parameters, e.g., the average number of users or the throughput per user, is a step worth evaluating. Beyond that, throughout this paper predictions only have been made on the behaviour of single SON Functions. The next promising step is to analyse the behaviour of combined SFMs. Finally the learned SFM as well as the learned clusters must be integrated into the SON Management.

## References

[1] S. Hämäläinen et al., "LTE Self-Organising Networks (SON) - Network Management Automation for Operational Efficiency", John Wiley & Sons, 2012
[2] C. Frenzel, "Objective-driven Operations of Self-Organizing Networks", Dissertation, 2016
[3] K. Tsagkaris et al., "SON Coordination in a Unified Management Framework", in IEEE 77th Vehicular Technology Conference (VTC), 2013
[4] Kostas Tsagkaris et al., "Unified Management Framework (UMF) Specifications Release 3," UniverSelf Project, Deliverable D2.4, 2013
[5] EU FP7 Project SEMAFOUR, "Deliverable 6.6 'Final report on a unified self-management system for heterogeneous radio access networks'", 2015, [Online]
[6] C. Frenzel et al., "Dynamic, Context-Specific SON Management Driven by Operator Objectives", in 2014 IEEE/IFIP Network Operations and Management Symposium (NOMS), 2014
[7] C. Frenzel et al., "SON Management based on Weighted Objectives and Combined SON Function Models", in ISWCS 2014 - Fourth International Workshop on Self-Organising Networks (IWSON), 2014
[8] S. Lohmüller et al., "Adaptive SON Management Using KPI Measurements", in 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS), 2016
[9] S. Hahn et al., "Managing and altering mobile radio networks by using SON function performance models", in 11th International Symposium on Wireless Communications Systems (ISWCS), 2014
[10] O. Iacoboaiea et al., "SON Coordination in Heterogeneous Networks: A Reinforcement Learning Framework", in IEEE Transactions on Wireless Communications 15.9, 2016
[11] Stephen S. Mwanje et al., "A Q-Learning Strategy for LTE Mobility Load Balancing", in 2013 IEEE 24th International Symposium an Personal, Indoor and Mobile Radio Communications (PIMRC), 2013
[12] G. James et al., "An Introduction to Statistical Learning", Springer, 2013
[13] C. J. Willmott et al., "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance", in Climate Research, 2005
[14] Z. Wang et al., "Mean Squared error: Love it or leave it? A new look at Signal Fidelity Measures", in IEEE Signal Processing Magazine, 2009
[15] Nokia Siemens Networks, "Self-Organizing Network (SON) Introducing the Nokia Siemens Networks SON Suite - an efficient, future-proof platform for SON", in Nokia Siemens Networks White Paper, 2009
[16] S. Hahn et al., "Classification of Cells Based on Mobile Network Context Information for the Management of SON Systems", in IEEE 81st Vehicular Technology Conference (VTC), 2015