The good, the bad, and the KPIs: how to combine performance metrics to better capture underperforming sectors in mobile networks

Ilias Leontiadis[*], Joan Serrà[*], Alessandro Finamore[*], Giorgos Dimopoulos[†], Konstantina Papagiannaki[*]

[*]Telefónica Research, Barcelona {firstname.lastname}@telefonica.com
[†]UPC BarcelonaTech, Barcelona gd@ac.upc.edu

***Abstract*—Mobile network operators collect a humongous amount of network measurements. Among those, sector Key Performance Indicators (KPIs) are used to monitor the radio access, i.e., the "last mile" of mobile networks. Thresholding mechanisms and synthetic combinations of KPIs are used to assess the network health, and rank sectors to identify the underperforming ones. It follows that the available monitoring methodologies heavily rely on the fine grained tuning of thresholds and weights, currently established through domain knowledge of both vendors and operators.**

In this paper, we study how to bridge sector KPIs to reflect Quality of Experience (QoE) groundtruth measurements, namely throughput, latency and video streaming stall events. We leverage one month of data collected in the operational network of mobile network operator serving more than 10 million subscribers. We extensively investigate up to which extent adopted methodologies efficiently capture QoE. Moreover, we challenge the current state of the art by presenting data-driven approaches based on Particle Swarm Optimization (PSO) metaheuristics and random forest regression algorithms, to better assess sector performance. Results show that the proposed methodologies outperforms state of the art solution improving the correlation with respect to the baseline by a factor of 3, and improving visibility on underperforming sectors. Our work opens new areas for research in monitoring solutions for enriching the quality and accuracy of the network performance indicators collected at the network edge.

I. Introduction

It is well known that mobile carriers are facing an explosion of data service demand. According to recent estimates [9], mobile data traffic will increase 8 times by 2020 while video services will represent more than 75% of the overall traffic. At the same time, the number of connected devices such as smartphones, tablets, vehicles, and Internet-of-Things objects, is expected to double, accounting to 1.5 mobile devices per capita.

This growth challenges operators, so that they have to radically rethink their network structure in order to keep offering good Quality of Experience (QoE) to their customers [17] in a ever-evolving mobile applications ecosystem. It follows that efficient network monitoring and engineering is paramount.

Currently, the health of the network is mostly monitored at the edge via the so-called *Key Performance Indicators (KPIs)*, i.e., passive measurements periodically collected from network radio access elements (i.e., sectors, towers, and controllers) to monitor both wireless channels and backhaul performance. Operators complement passive measurements with drive tests and controlled field experiments for fine-grained benchmarking and root cause analysis [2], [21], [24], [25], [28].

Assessing KPIs and per customer QoE for the whole network at scale is a daunting task: cellular networks are composed of hundreds of thousands of elements (e.g., sectors, cell towers, radio controllers, etc.), daily used by millions of customers using an heterogeneous set of applications. In practice, there is the need to consolidate multiple metrics into *single performance values* that reflect network elements' health. Considering sectors, such value is commonly referred to as *hotspot score*: the higher the score, the more "hot" or problematic a sector is [2], [30], [35]. The hotspot score combines multiple KPIs related to failures, signaling, coverage, voice, data availability, congestion, etc., into a single performance index, normally used to objectively *rank sectors*. In this way network teams can prioritize troubleshooting and interventions to address performance bottlenecks, understand and forecast current demand, define planning stategies, etc.

The methodology to combine KPIs into a single metric has been established by equipment vendors and operators based on logical decisions, Service Level Agreements (SLAs), and controlled experiments such as drive tests and field tests [3], [12], [22], [28], [34]. However, it remains an empirical approach, founded on deep domain knowledge fine-tuned over the years.

While the current methodology addresses some of the operators' current needs, it suffers from two main limitations. Firstly, it is unknown whether the current scoring function truly reflects user experience. Secondly, the aforementioned methodology lacks flexibility, as it obeys to a static picture of the users' and network's needs. For instance, in the 90s, the quality of the voice network used to be the most significant factor to improve but, with the roll out of Voice over LTE (VoLTE), focus will further steer towards data services. Such perturbations in the network equilibrium require (potentially significant manual) effort from vendors and operators to identify a novel configuration of KPIs capabile to reflect the new network scenario. In a nutshell, there is the need for a methodology that can easily cope with such service and demand changes, and technology evolution.

In this paper, we tackle the aforementioned problems by proposing a data-driven framework that improves monitoring flexibility with respect to state of the art solutions. We leverage data provided by a large mobile operator serving more than 10

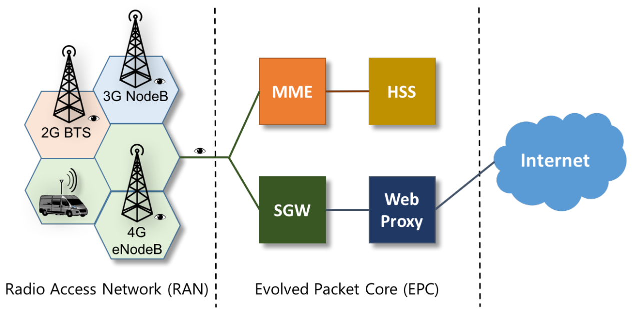


Fig. 1. Sketch of a mobile network. Notice that each tower might house multiple sectors.

KPI	Thres.	Category
Failed RRC/RAB requests ratio	$> 5\%$	Signaling
Signaling failure	$> 1\%$	
Call setup failure rate	$> 2\%$	Voice
Call drop rate	$> 5\%$	
HSDPA/HSUPA session setup success %	$< 90\%$	Data
Average Users Queuing	> 2	
Time Slots Transmitting	$> 80\%$	
Noise Rise	$> 10db$	Radio

TABLE I. Example of some KPIs and thresholds that indicate poor QoE. Notice that these are vendor-recommended values and, thus, they might differ from the ones used by the operators.

million subscribers, and we extensively study how to combine KPIs to create rankings that better capture underperforming sectors. Furthermore, we target to capture individual QoE components that affect users' experience such as web throughput, latency and video stalls. Our results indicate that the resulting ranking correlates three times more than the currently used method, hence offering a better vision on underperforming sectors and their relation to user experience.

II. Monitoring network KPIs

Fig. 1 sketches the architecture of a 4G mobile network. The *radio access* consists of hundreds of thousands of components (e.g., sectors, towers, and controllers). Those compose the mobile network "last mile", and bridge the users' devices with the *core network*, which enables access to voice calls and the Internet. To support troubleshooting and optimization, the vendors provide monitoring platforms to (passively) collect KPIs from radio access network elements and backbone links. To simplify the monitoring complexity, KPIs normally are aggregated over time (with periodicity varying from minutes to hours) and users.

There is no definitive list of KPIs. However, through the joint effort of vendors and operators, it is easy to identify a set of KPIs whose importance is acknowledged by multiple parties [2], [12], [21], [24], [34]. Table I lists some examples. We can categorize KPIs into five different groups:

- Signaling: These KPIs are mostly related to faults such as failure to establish a Radio Access Bearer (RAB), or Radio Resource Control (RRC), or the fact that the sector cannot efficiently reach the radio controllers.
- Voice: These KPIs capture failure to establish or maintain a voice call.
- Data availability: These metrics reflect the availability of high-speed data channels (such as HSDPA/HSUPA) at any given time or the number of data-active connected devices.
- Data Congestion: These metrics capture the fact that the capacity was reached. For instance, the average number of users queuing to get an HSDPA/HSUPA channel, or the number of times a data connection had to be dropped to make room for a voice call. Furthermore, they indicate the percentage of time that the radio was active transmitting data (radio utilization).
- Radio: Radio KPIs have to do with interference, power statistics, measured wireless noise, signal conditions, etc.

A. *Thresholding individual KPIs*

The primary use of KPIs is to enable operators to monitor the health of the network and to quickly identify bottlenecks. Therefore, it is natural to use KPIs as a way to flag network conditions that deteriorate below an established performance limit [2], [3], [35]. As a result, for each KPI, a *threshold* has been set. Such thresholding mechanisms are widely used in the industry, and allow network planners and radio resource operators to focus their attention where it is required. Table I shows examples of such thresholds.

Default threshold values are proposed by the equipment vendors, while operators further fine-tune them based on their experience, objectives, and domain knowledge. Therefore, significant investment is made through drive tests, controlled experiments, and A-B testing in order to identify performance bottlenecks, and how these relate to the KPIs and thresholds [3], [12], [22], [28], [34].

B. *Combining different KPIs*

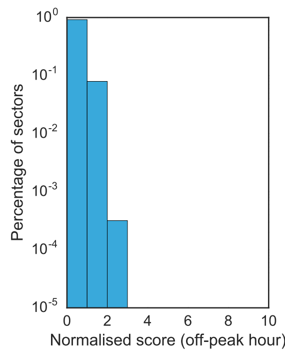
While establishing thresholds for each KPI enables a fine-grained vision of specific problems, each network element is associated with hundreds of metrics. It is then desirable to consolidate measurements into a single performance index so to i) quantify the "health" of each network element, ii) easily assess the whole network status and trends and iii) narrow down on sites that need attention.

An example of such indices is the *hotspot score* [2], [30], [35], which represents how "hot" or problematic a given sector is. It is a weighted combination of thresholded KPIs related to signaling, voice, data availability and data congestion:

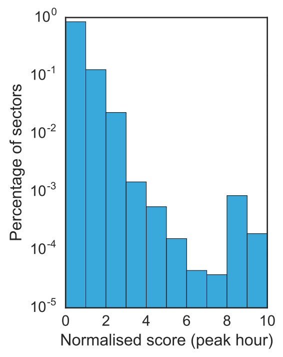
$$s_b = S\left(\mathbf{k}_b, \mathbf{w}, \mathbf{t}\right) = \sum_{i=1}^{n} w_i \cdot H\left(k_{b,i} - t_i\right), \tag{1}$$

where b is the sector under study generating n KPIs collected in the vector $\mathbf{k}_b$. The KPIs are associated to weight $\mathbf{w}$ and threshold $\mathbf{t}$ vectors, while $H(\cdot)$ is the Heaviside step function which outputs 1 when the KPI value $k_{b,i}$ reaches the corresponding threshold t_i and 0 otherwise. The higher the score s_b, the more "hot" the sector b is. In a nutshell, a hotspot score is a linear combination of the weights associated to KPIs that trigger.

Notice also that, since KPIs are gathered periodically, s_b is time dependent, but we avoid to express it in Eq. 1 for



(a) Hourly score, off-peak



(b) Hourly score, peak

Fig. 2. Ratio of sectors that achieve a certain hotspot score given the default formula. The majority of sectors have a score of zero.

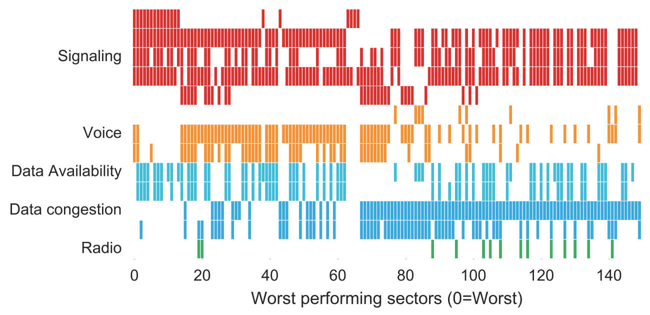


Fig. 3. The 150 sectors with the worse hotspot score and the type of KPIs that triggered it.

readability. This means that individual s_b values can be further combined (e.g., simply accumulated) to reflect performance over a longer time span. Sectors that exhibit high scores for a long period of time (e.g., a few days or even weeks) can then be flagged for intervention (such as changing the configuration, fixing possible faults, adding capacity, or even upgrading the whole site to a newer technology like 4G). However, when referring to a hotspot score in the following, we will consider the values computed using KPIs gathered at the baseline periodicity of the monitoring system, i.e., one hour for the network under study.

C. Challenging the state of the art

To better understand the hotspot score dynamics, let us consider a few examples related to the whole set of 3G sectors of a real mobile network operator serving a country with over 10 million subscribers (see Sec. IV for more details). Fig. 2(a) reports the fraction of sectors having a certain hotspot score for an off peak hour. We consider only the KPI values within this hour to create per-sector scores. As expected, when the network is lightly used, only a very small fraction of KPIs trigger with a maximum score of 2 (i.e., not critical for performance). Instead, the scenario significantly changes when considering the peak hour as reported in Fig. 2(b). The distribution is bimodal, with a cluster of sectors having a score higher than 8.

Fig 3 shows 13 of the important KPIs (grouped in sub-classes) for the worse performing 150 sectors out of tens of thousands in the network.[1] Columns are associated to sectors, sorted by decreasing score values (worse performing sectors are on the left), while colors are used to associate KPIs into categories (colored cells highlight which KPI triggers for which sector).

As previously introduced, the hotspot score is a mixture of signaling, voice, and data KPIs. Among the three, typically signaling issues have the highest weights since they can prevent utilization of voice and data. We can observe how those KPIs (Fig. 3 top) trigger more than the remaining ones. From left to right, we find sectors with signalling issues, followed by the ones with poor voice quality, and finally the ones that are congested. Voice problems are also highly valued due to the traditional service model of mobile operators, and due to the fact that users clearly perceive voice failures when they occur. Finally, data issues and congestion are given the lowest priority. Interestingly, the worse 60 sectors extensively trigger both voice and data KPIs, while for the rest there is a predominance of data KPIs.

The previous figures are clear examples that state of the art KPI analysis is useful to spot critical sites. However, this comes at the cost of a very rigid, static and empirical methodology for which we see a number of issues. First of all, it is unknown up to which extent the adopted approach reflects QoE as, to the best of our knowledge, no quantitative analysis has been performed. Secondly, even assuming the hotspot score function is properly configured, current techniques require a significant (domain knowledge and manual) effort to keep tools at pace with the constant evolution of Internet services and new technologies. Finally, while the hotspot score provides rankings of overall performance, it does not provide any indication about application specific-QoE.

One could argue that application-specific QoE is best captured through its own KPI. However, this has a clear overhead: monitoring application performance in a large-scale cellular network requires either the deployment of middleboxes that are capable of interpreting application performance by analyzing the (mostly encrypted) traffic of millions of users, or it requires collaboration with individual application developers in order to share their KPIs. Therefore the question we try to answer is: *could we use network KPIs that are already collected to monitor the health of the network to get insights into app performance?*

We believe there is the need to focus on network performance monitoring methodologies that i) shed light on the quality that people experience when interacting with specific data applications (e.g., web or video) and ii) can adapt to the mobile network dynamics. Hence, in this work, we address the following two challenges:

- We *study hotspot scores* using data collected from a real mobile network serving over 10 million subscribers. Specifically, we investigate up to which extent the used threshold and weighting mechanisms are sufficient to identify underperforming sectors.

[1]We cannot reveal the exact list of KPIs nor the exact number of sectors due to their sensitive nature.

- We *provide new methodologies* to enable us to gain some visibility on underperforming sectors. Specifically, we want to leverage the vast amount of data collected to create an automated data-driven framework capable to go beyond current domain knowledge and empirically-driven approaches.

III. A DATA DRIVEN APPROACH TO KPIS ANALYSIS

We assume to have a set of KPI values per sector that we want to match with specific QoE metrics that are affected by the performance of the edge-network, such as web throughput and latency or video performance. In this section, we examine how we can use the KPIs to extract a score that better matches the groundtruth. Based on Eq. 1, we identify two possible directions of exploration. On the one hand, we can keep the already defined hotspot formula, but create an optimization engine to better tune its parameters: the weights and the thresholds. On the other hand, we can create a new formula or an implicit model to combine the same input KPIs.

A. Optimizing the current score function

As mentioned, the most straightforward mechanism to better relate a given groundtruth $\mathbf{g}$ and the scores $\mathbf{s}$ is by optimizing the thresholds $\mathbf{t}$ and weights $\mathbf{w}$ involved in the calculation. To relate $\mathbf{g}$ and $\mathbf{s}$, we can use standard correlation. However, in our application, we are not interested in the raw quantities in $\mathbf{g}$ and $\mathbf{s}$, but rather in the sector rankings they define. In fact, the two quantities could lie in different ranges or have a monotonic non-linear relation and still produce the same sector ranking. Therefore, a natural choice to measure the correlation between $\mathbf{g}$ and $\mathbf{s}$ is the Spearman's rank correlation coefficient [33], often called Spearman's ρ. The Spearman's ρ is only affected by changes in the rankings of the input variables, and is thus invariant to scale, location, and monotonous non-linear relations. Its output ranges from -1 (perfect reverse ranking) to 1 (exact same ranking).

Having defined the function relating $\mathbf{g}$ and $\mathbf{s}$, we can now formulate our problem in terms of a classical optimization problem [6]: we want to find the weights $\mathbf{w}$ and thresholds $\mathbf{t}$ that maximize

$$\rho(\mathbf{s}, \mathbf{g}), \tag{2}$$

where $\mathbf{s} = [s_1, \ldots s_m]$ and $\mathbf{g} = [g_1, \ldots g_m]$ are vectors collecting all m sector scores and ground truth measurements, respectively. Notice that, as the values in $\mathbf{w}$ and $\mathbf{t}$ are real numbers, we have a potentially infinite number of such combinations.

Given that the calculation of ρ is not directly differentiable (due to the ranking operation), and that it is computationally cheap (essentially involving two vector sorting, one vector subtraction, and one vector multiplication), we decide to solve the previous optimization problem with a *Metaheuristic algorithm* [26]. Metaheuristics conform a general algorithmic framework for addressing optimization problems. Unlike classical optimization algorithms and iterative methods, metaheuristics make few assumptions about the optimization problem being solved, and can thus be applied in a variety of problems, including problems with non-differentiable functions. Metaheuristics allow us to define our own optimization function, while they provide a way of exploring the parameter space (thresholds $\mathbf{t}$ and weights $\mathbf{w}$) in a structured and efficient way. Despite being approximate and non-deterministic, metaheuristic algorithms can efficiently explore the search space and provide near-optimal solutions within a reasonable amount of time [6], specially if the optimization function is not computationally expensive (as it is in this case). These methods are often inspired by processes occurring in nature, such as Darwinian natural selection, annealing, and collective behaviour of ants [5].

1) Particle swarm optimization (PSO): Particle swarm optimization (PSO) [32] is a population-based metaheuristic for solving continuous and discrete optimization problems [6]. PSO has recently gained increasing popularity among researchers and practitioners as a robust and efficient technique for solving difficult optimization problems. It makes few or no assumptions about the problem being optimized, can search very large spaces of candidate solutions, and can be applied to problems that are irregular, incomplete, noisy, dynamic, not necessarily differentiable, etc. (see [6], [10], [32] and references therein).

PSO operates by having *a population of candidate solutions*, which are metaphorically represented as *particles*. At each iteration, these particles explore the search-space according to their current positions and a velocity towards a goal. They iteratively calculate the fitness ρ corresponding to their current position, and update the latter according to the available knowledge of the search space. The movement of a particle is affected by the best position it has found but, in addition, it is also affected by the best known positions discovered by other particles [10]. Moreover, particle movements are not deterministic, but partly stochastic, thus facilitating the exploration of the search space.

Before delving into the details of our PSO algorithm, we first need to define what corresponds to a particle's position, which we will denote by the vector $\mathbf{x}_i$. Specifically, in our case, the position will correspond to the $2n$ weights and thresholds: $\mathbf{x}_i = [\mathbf{w}, \mathbf{t}]$. With that, we see that we can easily constrain the search space with some domain/intuitive knowledge, imposing upper and lower bounds for the particles' positions. For the weights, we define them to be $-1 \leq w_i \leq 1$ (a resulting weight of zero means that the triggered KPI does not influence the ground truth whereas values above or below zero contribute positively or negatively towards the score). For the thresholds, for each KPI, the minimum and maximum over all sectors is used such that $\min(k_i) \leq t_i \leq \max(k_i)$. When a particle exits the search space, its position is reset to a random location inside the established bounds.

Algorithm 1 details the PSO functioning. First of all, we initialize τ particles $\mathcal{P}_i$ with random positions $\mathbf{x}$, random velocities $\mathbf{v}$, and lowest possible personal fitness $\rho = -1$. Next, we start iterating, with a maximum number of iterations λ. When iterations are finished, we return the best found correlation $\rho^\star$ and position $\mathbf{x}^\star$ which, as mentioned, comprises the best found weights $\mathbf{w}^\star$ and thresholds $\mathbf{t}^\star$. The first inner loop of Algorithm 1 checks every particle's fitness by computing the Spearman's ρ. A particle i updates its best known location $\mathbf{x}_i^\star$ and found correlation $\rho_i^\star$ if the latter has improved over the previous value the particle had. The second inner loop performs the actual search towards a better solution.

Algorithm 1: Particle swarm optimization algorithm basics.

```
Input: The set of KPIs $\mathbf{k}$, a scoring function $S$, the ground truth $\mathbf{g}$, a fitness
    function $\rho$, and position bounds $\Gamma$.
Output: Best found correlation $\rho^\star$, weights $\mathbf{w}^\star$ and thresholds $\mathbf{t}^\star$.
Parameters: Maximum number of iterations $\lambda$, number of particles in the
swarm $\tau$, and mutation probability $\alpha$.
// Initialize particles at random with correlations $\rho^\star = -1$
$\mathcal{P} \leftarrow Init()$;
// Iterate
for $l \in [1, \lambda]$ do
    // For each particle $\mathcal{P}_i = \{\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_i^\star, \rho_i^\star\}$
    for $i \in [1, \tau]$ do
        // Calculate $\rho$ in the current position
        $\mathbf{w}, \mathbf{t} \leftarrow \mathbf{x}_i$;
        for $b \in [1, m]$ do
            $s_b \leftarrow S(\mathbf{k}_b, \mathbf{w}, \mathbf{t})$;
        end
        $\rho_i = \rho(\mathbf{s}, \mathbf{g})$;
        // Update if better
        if $\rho_i \geq \rho_i^\star$ then
            $\rho_i^\star, \mathbf{x}_i^\star \leftarrow \rho_i, \mathbf{x}_i$;
        end
    end
    // For each particle $\mathcal{P}_i$, identify the $\mathcal{P}_j'$ particle in its
    // neighbourhood having the highest $\rho_j^\star$
    $\mathcal{P}' \leftarrow GetNeighbors(\mathcal{P})$;
    // For each particle $\mathcal{P}_i = \{\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_i^\star, \rho_i^\star\}$ and
    // best neighbor $\mathcal{P}_j' = \{\mathbf{x}_j, \mathbf{v}_j, \mathbf{x}_j^\star, \rho_j^\star\}$
    for $i \in [1, \tau]$ do
        // Compute new velocity using random vectors $\mathbf{u}$
        $\mathbf{v}_i = \chi \cdot \left(\mathbf{v}_i + \phi_1 \mathbf{u}_1 \otimes (\mathbf{x}_i^\star - \mathbf{x}_i) + \phi_2 \mathbf{u}_2 \otimes (\mathbf{x}_j^\star - \mathbf{x}_i)\right)$ ;
        // Mutate velocity components with probability $\alpha$
        $\mathbf{v}_i \leftarrow Mutate(\alpha, \mathbf{v}_i)$;
        // Compute new position
        $\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i$ ;
        // Constrain the particle within the desired bounds
        $\mathbf{x}_i \leftarrow Constrain(\Gamma, \mathbf{x}_i)$;
    end
end
// Identify the particle $j$ with the best correlation $\rho_j^{best}$
$\rho^\star, \mathbf{w}^\star, \mathbf{t}^\star \leftarrow \rho_j, \mathbf{x}_j$;
return $\rho^\star, \mathbf{w}^\star, \mathbf{t}^\star$
```

The first step in the second inner loop is looking for promising positions found by the particle's neighbours. To facilitate exploration, it is common practice to consider different interactions between the particles, grouping them in so-called neighbourhood topologies [6]. In our implementation, particles are grouped following a *ring* topology (Fig. 4). Therefore, the neighborhood of a particle only consists of two other particles. Particles that are neighbors in the ring collaborate together and influence each other. This implicitly creates small groups of particles that explore different parts of the search space, while the ring itself is ultimately pulled towards the best known solution [10]. In the example of Fig. 4, particle A will be influenced to move towards particle G (as it has found a solution with better correlation), whereas particle B will move towards particle C. Particle G will keep moving towards the same direction.

The next step in the second inner loop 1 updates the particles' velocity[2]. The new velocity of a particle $\mathcal{P}_i$ ($\mathbf{v}_i$) is influenced by the current position $\mathbf{x}_i$, the distance from the position with its best known fit $\mathbf{x}_i^\star$, and the distance from the best known fit of the closest particle $\mathbf{x}_j^\star$. To further enhance the exploration capabilities of particles, the velocity vectors defined by $\mathbf{x}_i^\star - \mathbf{x}_i$ and $\mathbf{x}_j^\star - \mathbf{x}_i$ are component-wise multiplied

[2]Notice that a velocity vector in more than one dimension has a modulus and an angle. Therefore, it also carries information of direction.

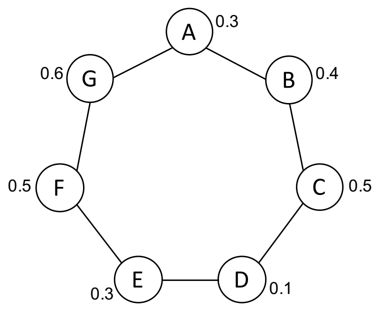


Fig. 4. Swarm of 7 particles and the Spearman's ρ of their best discovered position. Each particle movement is influenced by its own best found position and, at the same time, is also guided toward the best known position of its two neighbours.

($\otimes$) with random vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, formed by uniformly-distributed random values between 0 and 1.

Constants χ, ϕ_1, and ϕ_2 are pre-set following the so-called Clerc's constriction method [10], which is common practice in the PSO literature. These constants control the behaviour of the particles, and allow an elegant and well-explained method for preventing explosion and ensuring convergence [32]. In our implementation, to prevent the stagnation of the swarm and to facilitate escaping from local maxima, a further random mutation of the velocity components is employed [16]. For that, we use a small probability α.

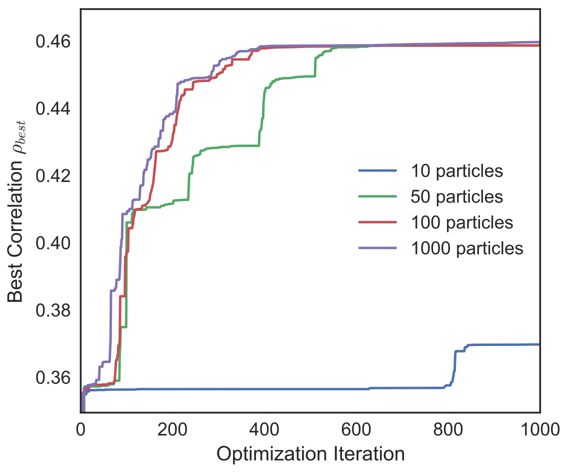
According to Algorithm 1, we need to define three parameters: the number of iterations λ, the number of particles τ, and the mutation probability α. From a theoretical stand point, the larger the value of λ and τ, the higher the chances to find globally optimal solutions [10]. However, an unreasonably high value of those parameters could harm the efficiency of the algorithm (in terms of computation time). In practice, we found that, with a ring of $\tau = 100$ particles, $\alpha = 0.001$ and $\lambda = 500$ iterations, we converge towards a solution that does not significantly improve if we wait longer or perform additional trials. Fig. 5 shows the sensitivity of the correlation ρ with respect to the number of iterations. The figure also shows that, for 500 interations, only unreasonable settings of τ and α produce non-optimal results. We empirically find these settings to correspond to $\tau < 50$ and $\alpha > 10^{-3}$. Therefore, we set $\alpha = 10^{-3}$.

B. Improving over the current function

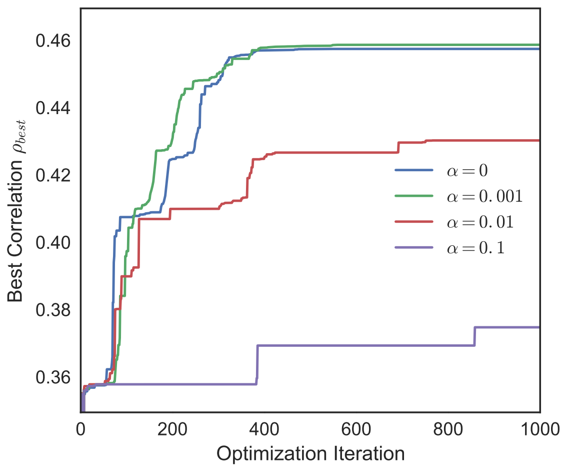
Until now, we have only discussed how to optimize the default weights and thresholds used in the scoring function of Eq. 1. However, such function could be limited by the application of the thresholds, which hide the *raw "analog"* KPIs value. Thus, we could potentially significantly improve the correlation between **g** and **s** by using the richer information in the raw KPI values. We envision two alternative hotspot score functions.

Linear combination of raw KPIs: this simply consists in removing the thresholding from Eq. 1, such that

$$\hat{S}_b(\mathbf{k}_b, \mathbf{w}) = \sum_{i=1}^{n} w_i \cdot k_{b,i}. \tag{3}$$



(a) Swarm size



(b) Mutation probability α

Fig. 5. Sensitivity of PSO with respect to number of iterations considering different swarm sizes τ (a) and mutation probabilities α (b).

To avoid having heavily asymmetric weights, we however need to normalize the KPIs to be in the range between 0 and 1. This is the most simple model we can think of in order to combine KPIs into a single hotspot score.

Non linear models: in this case, we rely on machine learning algorithms to explore non-linear relationships between KPIs and the ground truth. More specifically, we will not create anymore a scoring function, but we will associate directly KPIs and ground truth $\mathbf{g}$. A generic way to learn such association is through a machine learning regression algorithm [19].

A machine learning-based approach indeed offers a number of advantages. Firstly, the resulting model does not make any assumptions about the underlying function. Secondly, the model can (potentially) produce an actual value of the estimated performance rather than just a ranking. For instance, a regression tree [7] builds a model that exploits the KPIs to estimate the actual conditions for each sector at that time: $f(\mathbf{k}) \rightarrow \hat{\mathbf{g}}$. Depending on the availability of ground truth for the training set, the predicted value $\hat{\mathbf{g}}$ can be any condition that correlates with the KPIs.

At the same time, a machine learning-based approach presents some disadvantages. Firstly, most machine learning models optimize against the target variable (e.g., mean squared error), whereas in the optimization process we define our target: optimize the ranking. Thus, this does not exactly match current practice, as operators would prefer a prioritized ranking of sectors to address. Secondly, the resulting machine learning model is often not easily human-interpretable, and network engineers cannot modify it to include external restriction such as SLAs or other priorities. Furthermore, certain KPIs might have significant importance in terms of detecting a *failure* rather than a performance degradation. Thirdly, it breaks the current paradigm and the tools that are already used in operation.

In our experiments, we use the scikit-learn toolkit [31], a well-known machine learning framework offering various techniques (e.g., decision trees, random forests, SVMs, neural network regression, etc). In preliminary analysis, the best results were achieved with *random forests* [20]. We use the *mean square error* for our tree split criterion and 100 tree estimators. To avoid any over-fitting, we restrict the minimum number of samples that end up in leaf-nodes to 0.5% of the dataset, as this will not allow the tree to grow indefinitely. These parameters were studied with separate training set and they were later applied to our validation data set.

IV. Data sets

As introduced in Section III, we aim to study how to combine sector KPIs to obtain a synthetic score that better reflects users' QoE. For this purpose, we combine two data sources i) a set of KPIs collected per sector and per hour, ii) QoE metrics collected using weblogs at per HTTP request granularity and then associated to specific sectors and for the same hour window. We can think of KPIs as the *features* that we aim to associate with the QoE *groundtruth* for the same sector during the same time period.

We leverage one month of such measurements, collected between January and February 2016 in an operational mobile network serving over 10 million subscribers. To ease the analysis and reduce the impact of night/day fluctuations, we study the peak hour (same hour across multiple days). It is important to underline that the data sets capture the network at scale, i.e., all sectors and all customers observed during the investigation period.

A. Sector KPIs - Features

As described in Section II, a number of KPIs related to i) signaling, ii) voice, iii) data availability, iv) data congestion and v) radio performance are collected. In total, we consider 21 KPIs [3] for n sectors, where n is in the order of hundreds of thousands. These KPIs are measured *hourly*. Notice that this granularity has been chosen by the operator as a higher granularity imposes high network overhead. Therefore, we will also aggregate our groundtruth into the same one-hour bins per sector.

[3]KPIs are selected based on both internal knowledge and vendors recommendation

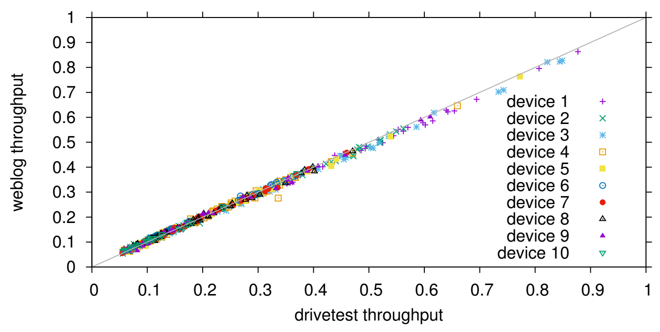


Fig. 6. Weblog throughput validation using country-wide drive tests and various devices. The performance measured by the operator's middlebox matches the one measured experimentally.

B. *Weblogs - Groundtruth*

Web QoE metrics are provided by a web accelerator middlebox (*webproxy* in the following) that is deployed in the operator's network and is responsible to cache and compress HTTP objects (see Fig. 1). At the same time, for each HTTP request, the webproxy logs information such as timestamp, download duration, subscriber's id, bytes transferred, and URL[4]. More importantly, each transaction contains *TCP metrics* (e.g., min/max/avg round trip time, dropped or duplicated packets etc.) capturing the delivery performance between the middlebox and users' device.

Video QoE: To extract video QoE we exploit the fact that the YouTube player reports to Google servers summary statistics at the end of each video playback. These include if the video has successfully loaded, if the playback has started, paused or stopped, if there were stalls and how long these lasted [13]. Despite most of the content is now served via HTTPs [15], we still see a residual amount of YouTube videos served over HTTP and we use this to generate statistics per video transaction (e.g., number and duration of stalls, average resolution, etc).

Mapping requests to a sector: The webproxy logs do not contain any information related to the *sector* where the data were consumed. Therefore, we use *radio events* recorded by the Mobility Management Entity (MME) (Fig. 1) in order to enhance each web transaction with the set of sectors that were used. Indeed, MME servers logs the "control plane" messages (related to paging, radio channel requests, and handovers) each containing the sector from which users devices sent the message. For each subscriber we can then create a timeline describing the sectors connected. With such information we can tag weblogs entries based on the sector where the requests have been generated. To avoid any ambiguities, if more than one sector were used to serve a object (i.e., there was a handover), we discard the transaction (this occurs for less than 5% of the downloads).

Notice that this process is not trivial: radio events and weblogs corresponds to more than 3TB per day. In our case, we exploited the BigData cluster of the operator to implement this enrichment. However the described job requires processing that can take hours, even in large data clusters.

[4]Users' privacy is protected by proper anonymization techniques.

Validation: We validated the fact that the weblog transactions can capture the resulting web QoE by performing large-scale drive tests. We discovered that the metrics correlate well with one exception: the weblog-based throughput estimation is only meaningful when considering larger downloaded objects ($> 700kB$). Therefore, in our analysis we filtered out smaller objects when measuring throughput. Fig. 6 demonstrates this correlation between hundreds of drive tests and the associated weblogs entries for different mobile devices. These results indicate that web transaction performance counters, as seen by the webproxy, are a good proxy for the conditions that are experienced by the end-users.

Per-sector metrics: The final step is to aggregate per requests metrics at a per sector and per-hour granularity (to match the KPI granularity). Therefore, we extract distribution related to the (min/max/avg and percentiles). In this paper we provide correlations with respect to the median performance but we have internally considered other metrics too.

C. *Combining the KPIs with the groundtruth*

At the end of this process we have hourly per sector information related to the groundtruth $\mathtt{g}$ of i) *latency*, ii) *download throughput* and iii)*YouTube video streaming stalls.*

To make any statistically significant correlations we remove all samples where less than 20 web downloads were made by users in a given sector-hour. Overall, the data set contains more than *2 million of throughput and delay* samples and *7,000 video streaming* samples (there are fewer sectors that had more than 20 non-encrypted video downloads within one hour).

V. Evaluation

We start our evaluation with a high level comparison of the achieved correlation by all methodologies. Then, we drill down into the results of PSO and machine learning methodologies to better investigate their strengths and weaknesses. Finally, we investigate the intersection among the under performing sectors found by different methodologies, i.e., up to which extent they offer a different view on performance issues.

To avoid any possible bias due to over-fitting, we randomly split each samples population in two halves: the first is used to identify the best solution (training), while with the second we assess the correlation with respect to ground truth (testing).

A. *Normalization*

Notice that, for all results, weblogs QoE metrics are normalized with three distinct factors, one for throughput, one for latency, and one for video stalls. *The normalization factor values are consistent throughout this paper.* For example, if the delay normalization factor is 1000ms then all delays in all graphs have been divided by 1000ms. This allows to still have meaningful comparisons without revealing actual values which unfortunately cannot be publicly disclosed.

Although the exact values are not given it is important to understand the significance of the normalized values [8].

- **Throughput:** Normalized values bellow 0.1 typically mean extremely slow performance, slow web loading and the inability to play videos. Values between 0.1 and 0.25

Approach	Configuration	Delay	Thput	Stalls
① Baseline	All KPIs	0.15	-0.14	0.07
② Baseline	Only data KPIs	0.19	-0.17	0.09
③ PSO on S	Only weights	0.29	-0.26	0.17
④ PSO on S	Only thresholds	0.30	-0.29	0.19
⑤ PSO on S	Weights and thresholds	0.41	-0.36	0.22
⑥ PSO on $\hat{S}$	weights	0.46	-0.42	0.23
⑦ Non-linear	Random forest regression	0.40	-0.32	0.23

TABLE II. SPEARMAN'S CORRELATION FOR DIFFERENT OPTIMIZATION STRATEGIES BETWEEN THE RESULTING SCORE AND THE GROUNDTRUTH (WEB DELAY, THROUGHPUT, AND VIDEO STALLS). POSITIVE CORRELATION FOR VALUES THAT INCREASE AS RESULTING HOTSPOT METRIC INCREASES (E.G., DELAY AND VIDEO STALLS) AND NEGATIVE WHEN VALUES DECREASE (E.G., THROUGHPUT).

indicate slow performance that might be acceptable for mobile networks. Values between 0.25 an 0.5 can be considered normal performance whereas values above 0.5 are represent excellent performance (i.e., suitable to watch HD videos).

- **Delay:** Similarly, normalized delays above 0.75 typically mean extremely slow performance where webpages take several seconds to open. Values between 0.25 and 0.75 indicate latencies that can result in noticeable lag in VoIP applications that results in poor QoE . Values between 0.1 and 0.25 indicate normal performance whereas values below 0.1 indicate fast, low-latency network that can facilitate real-time, latency-sensitive gaming.

B. Overall results

Table II details the Spearman correlation coefficient ρ for all considered techniques. Results are averaged across 10 runs.

Baseline ①: This corresponds to the achieved correlation with currently used weights and thresholds. Table II shows a small correlation. This means that the state of the art solution indeed captures the performance of the sectors. In fact, recall the bimodal distribution between peak and non-peak hours seen in Fig. 2, and Fig. 3 detailing sectors triggering multiple KPIs. Both pictures were suggesting the presence of correlation which is now quantified in Table II.

Interestingly, correlation is different for the considered QoE metrics, with video stalls presenting the smallest values. This is expected since video stalls depend on a number of other factors including users' device type, CDN providers, and is the metric for which we have the least amount of samples, i.e., the overall performance of sectors is expected to be associated to a varied set of services besides video streaming.

Baseline (only data KPIs) ②: As shown in Fig. 3, the default parameters of the baseline approach put emphasis on hotspots suffering from signal and voice issues. Given that we focus on understanding data-related QoE it is important to examine the correlation when *only data KPIs are considered.* By only considering data KPIs in the baseline formula ②, we do notice a slight improvement with respect to ①, although the magnitude of the correlation is still small. This is an indication that a holistic approach which combines signaling, voice, and data might not be ideal to capture the performance of individual applications.

PSO on current hotspot formula (③, ④, ⑤) Applying PSO on the baseline formula improves correlation, but the entity of such improvement differs based on which parameters are optimized. Specifically, optimizing only weights presents the smallest benefit ③, while the best solution is achieved optimizing both weights and thresholds ⑤ with a striking improvement factor of $\times 2.7$, $\times 2.5$, and $\times 3$ for delay, throughput, and video stalls, respectively. This is because PSO allows to identify an optimal set of parameters specific for each target metric, rather than enforcing a single configuration as for the baseline.

PSO on modified hotspot formula ⑥: As discussed in Sec. III-B, the Heaviside function transform the raw KPI values into binary values (KPI triggered or not). Our intuition is that those values provide a wealth of information on performance, hence the raw KPI values should be directly exploited in the hotspot formula. We now see that when applying PSO on the modified hotspot formula $\hat{S}$ (Eq. 3), correlation further improves with respect to baseline by a factor of $\times 3.06$, $\times 3$, and $\times 3.2$ for delay, throughput, and video stalls, respectively.

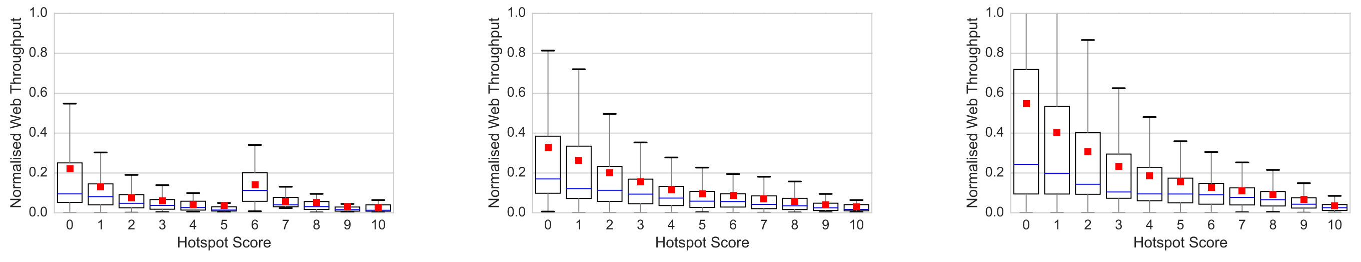
Random forest regression ⑦: Finally, having assessed correlation for all discussed linear combinations, we explore implicit *non-linear* functions via random forest regression (Sec. III-B). Results show that the obtained Spearman's correlation is not as good as the best PSO solution ⑥. The main reason is the nature of the objective function that is used in each algorithm: while in PSO we can set the optimization target to match our needs (in our case to provide a better ranking or Spearman's correlation), off-the-shelf regression trees depend on metrics such as mean squared errors that better correlate with groundtruth values rather than the ranking. Still, it is interesting to notice that such algorithms already provide a significantly better solution than the baseline approach.

***Takeaways**: Metaheuristic and machine learning algorithms provide significant improvement over baseline approaches. It is also recommended to apply a simple linear combination of raw KPI values and to avoid the currently used thresholding mechanisms.*

C. Focusing on PSO

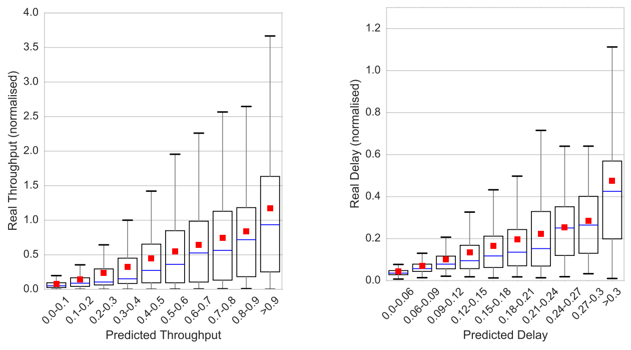
Fig. 7 shows the relationship between the hotspot score and the throughput for the baseline approach (Fig. 7a), the hotspot formula S optimized for both weights and thresholds (Fig. 7b), and the hotspot formula $\hat{S}$ without the Heaviside function (Fig. 7c). Specifically, we bin the performance score in bins from 0 to 10, and for each bin we plot the throughput distribution using box plots capturing 5^{th}, 25^{th}, 50^{th}, 75^{th}, 95^{th} percentiles. We further report the average of each bin with a dot.

Notice how the boxplots are relatively "flat" for the baseline approach, with a high concentration of sectors having a score of zero or six (Fig. 7a). This bimodal distribution, also noticed in Fig. 3, is the result of the (manual) tuning of weights and thresholds of the current monitoring system.



(a) Baseline (all KPIs) (b) PSO on S: weights and thresholds (c) PSO on $\hat{S}$: weights

Fig. 7. Detail PSO correlation results for throughput measurements.



(a) Throughput (b) Delay

Fig. 8. Estimating web delay and throughput using the KPIs. We observe that the average delay matches the predicted value but there is a wide distribution within each bucket.

Real\Est.	poor	medium	good	poor	medium	good
poor	85%	14%	1%	71%	26%	3%
medium	32%	59%	9%	32%	42%	26%
good	5%	21%	74%	2%	16%	82%

TABLE III. NORMALIZED CONFUSION MATRIX FOR REAL V.S. PREDICTED THROUGHPUT (LEFT) AND DELAY (RIGHT). IDEALLY, ALL SAMPLES SHOULD BE ON THE DIAGONAL.

Conversely, PSO spreads the scores across all bins, better separating well-performing from poor-performing sectors. Notice also how median values for the two PSO optimizations (Figs. 7b,c) are very similar, but average values are better separated using the modified formula $\hat{S}$. More in details, while 37% of sectors have a score larger than 0 for Fig. 7b, this drops to 19% for Fig. 7c. Considering results for PSO on $\hat{S}$, sectors with score 10 have $\times 11.1$ less throughput when compared with the sectors with score 0. In contrast, the baseline method (Figs. 7a) shows significantly smaller separation, namely $\times 5.7$. Finally, sectors with score 0 have $\times 2.6$ the throughput when compared to the baseline scoring, demonstrating that this method can better isolate such cases. Results for delay and video stalls are similar but we do not report them due to lack of space.

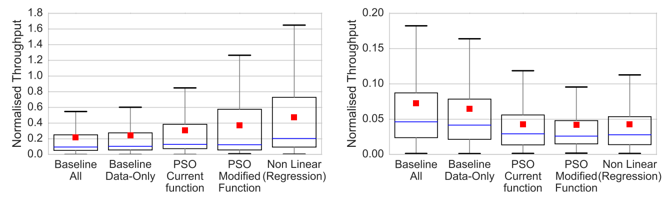
D. Focusing on Random Forests

Table II shows how PSO overperforms random forest regressions in terms of correlation due to the difference in the optimization function. However, the advantage of regression is that it provides an estimation of the actual conditions (e.g., estimates the expected delay, throughput an video stall *values*).

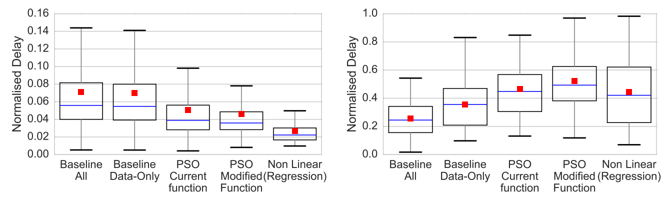
Fig. V-C shows these estimations of throughput (a) and delay (b) obtained with the regression. Specifically, we split regressed values in 10 bins, and for each we plot the distribution of groundtruth values using box plots. In theory, a very good model would have very narrow distribution centered around each decile. While averages are indeed very close to the real value, each bin presents a wide distribution of values, i.e., the model is still affected by some noise. Nevertheless, we can clearly separate performing from underperforming sectors.

Such observation motivated us to perform an additional experiment: create a *classifier* based on ground truth buckets. In fact, despite the fact that regression allows to have an estimate of QoE, as a first approximation it can be sufficient to divide sectors in classes based on performance 'labels' (e.g., poor, medium, good). To achieve this, we need to partition regression values in classes (we obtained the raw thresholds for the split through conversations with the operator network teams). The selected thresholds however create highly imbalanced classes: we can rarely find samples that belong to the high delay or the low bandwidth case (most of the sectors exhibit medium or high performance even during peak-hours). To address this issue, we use a balanced tree to assign a higher weight to classes having smaller number of samples (low throughput or high latency). The reasoning is that providers mostly care about discovering underperforming sectors and some miss-classification between high and medium performance classes is tolerable. The confusion matrix in Table V-C reports the obtained classification accuracy for both throughput and latency. The best performance is obtained for good-vs-good and poor-vs-poor, i.e., the classes we aim to separate, while the major confusion comes from medium performance values being confused with poor performance values.

Takeaways: *Off-the-shelf machine learning regression techniques offer a slightly lower correlation with respect to PSO. However, they enable an estimation of the actual values of performance metrics (instead of a score) which is, as a first order approximation and without very fine-grained tuning, sufficiently accurate to separate correctly performing from under performing sectors.*

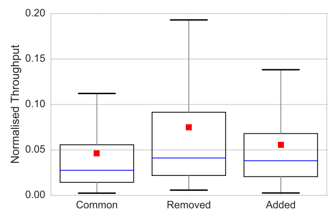


(a) throughput for top (left) and bottom (right) sectors

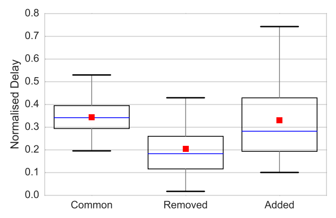


(b) latency for top (left) and bottom (right) sectors

Fig. 9. The actual throughput and latency of the sectors that ended up at the top or at the bottom 500 of each ranking. Results for video are similar.



(a) Throughput



(b) Delay

Fig. 10. Comparing the PSO optimized vs. baseline rankings for the top 500 samples. *Common* sectors appear in the top 500 of both rankings, *removed* sectors do not appear in the optimized ranking and *added* sectors did not appear in the default ranking.

E. Comparing top/bottom performing sectors

The final objective of the discussed methodologies is to rank sectors based on performance. In our case, bottom performing sectors are the ones with the highest delay, lowest throughput and highest percentage of videos that experienced stalls. However, it is important to identify healthy sectors as well, since they can provide further information to understand the causes behind poor performance (e.g., comparing sites configurations, locations, etc.)

Comparing different approaches: We focus on the top and bottom 500 sectors, and for each group of sectors, Fig. 9 shows the distribution of the ground truth throughput (a) and latency (b) using box plots. As previously observed, the application performance of sectors that end up at the top and bottom of the rankings varies significantly with each methodology. We observe that PSO applied on the modified hotspot formula $\hat{S}$ ⑥ and random forest regression ⑦ achieve significantly better results in identifying the well-performing sectors compared to the baseline (left plots). The main reason is the nature of the original function: to identify faults, the baseline methodology would just assign a score equal to zero to most of the average and good performing sectors since no KPI thresholds are defined conservatively. Instead, PSO ⑥ is better at identifying

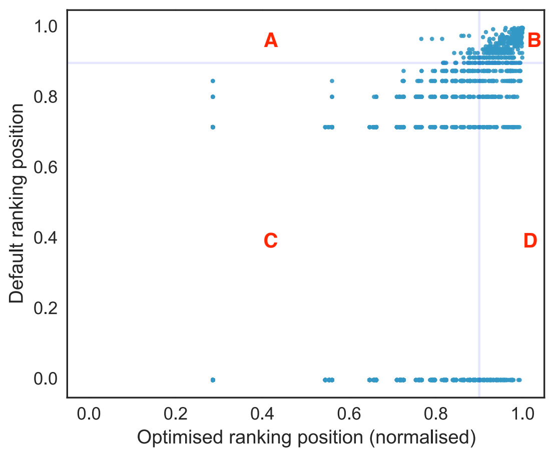


Fig. 11. Comparing baseline vs. optimised rankings of the 10 percentile worse-performing sectors in terms of throughput groundtruth. Ideally, both rankings should place them in their top 10 percentile too.

poor-perfoming sectors in each of the applications (including for video stalls which results are not reported due to space constraints). These results further corroborate the overall comparison discussed in Sec. V-B. Considering instead the bottom performing sectors (right plots), PSO ⑥ presents the most consistent performance across metrics, while random forest regression suffers of lower accuracy in assessing latency performance.

Comparing baseline and best solution found: In Fig. 10 we further quantify the intersection between the baseline and the best PSO approach ⑥ rankings. In particular, we focus on the worst 500 performing sectors and compare the QoE metrics distribution for sectors found in both, only PSO (added), and only baseline (removed) rankings. First of all, only 56% (49%) are found by both techniques for throughput (latency). On closer inspection, these are sectors that exhibit poor performance in all possible KPI classes (voice, data, signaling) and show the lowest throughput and delay. Notice how the remaining sectors captured only by the baseline approach (*removed*) present on average a higher throughput (lower latency) with respect to the common sectors, whereas the ones discovered (*added*) show performance that is as poor as the ones in the common set.

To conclude, Fig. 11 shows a scatter plot of the sectors' rank position between these two methods for the worse-performing 10% sectors in terms of throughput. An ideal ranking methodology would place these sectors in the top 90% (e.g., areas A and B for ① and areas B and D for ⑥). As expected, the majority of the sectors cluster at area B, indicating that both rankings are able to identify a good part of the underperforming sectors. More interestingly, areas C and D present sectors having very small scores according to the baseline approach. The reason is that the baseline methodology assigns a large number of sectors with score zero as none of the KPIs "triggered". This is compatible with the nature of the original function: to identify important faults in the network rather than sectors with poor throughput. We further notice that PSO puts most of these sectors at the top 60% of the ranking with quite a lot being correctly in the top 10% (lower and middle parts of area D). Results for delay and video stalls

Category	Throughput	Latency	Stalls
Signaling	2%	37%	3%
Voice	0%	1 %	2%
Availability	8%	30%	30%
Congestion	88%	26%	36%
Radio	2%	6%	29%

TABLE IV. DECOMPOSING THE MOST IMPORTANT FEATURES FOR EACH QOE COMPONENT INTO KPI CATEGORIES (INFORMATION GAIN).

are similar (not shown due to space limitations), demonstrating that this methodology can better capture individual application performance. Finally, notice that points in area B do not perfectly lay on the bisect line, i.e., under performing sectors are spot as an aggregate, but severity is differently captured by the two methodologies.

Takeaways: *By applying a data-driven methodology like PSO, operators are empowered with a flexible tool that use the already collected KPIs to improve their view on underperforming sectors.*

VI. DISCUSSION

Selecting QoE metrics: While we used KPIs to estimate web experience (delay, throughput, video), this methodology can be used to build a ranking for other QoE metrics that can potentially correlate with the KPIs. These can include dropped calls, VoIP performance, specific application performance (e.g., facebook, gaming, etc) or even customer satisfaction. The main requirement is to have enough samples of groundtruth to build the model.

State of the art solutions try to provide a single performance scoring function to capture different types of problems. To quantify this effect, we focus on the worst 500 sectors ranked by the best PSO solution ⑥ and compute the fraction of the score associated to each KPI class. Table IV compares the importance of the KPI classes for the three QoE metrics considered. As we observe, *KPIs' importance significantly varies depending on the QoE metric*. Throughput, depends almost exclusively on KPIs that have to do with congestion and availability of high-speed channels. Latency is a more complex phenomenon as it depends on signaling failures (e.g., failures to establish a dedicated channel), availability of resources and congestion. Finally, interestingly enough, even if video streaming is a throughput driven service, the importance of KPIs when capturing stalls significantly differs from generalized throughput: Video quality also depends on the radio conditions (interference, noise, etc) as these can create jitter and download stalls.

These results manifest the need of building an adaptive data-driven approach to exploit the, already collected, KPIs in order to better understand both historical and real-time performance of each application at each sector of the network.

Building a generic ranking: We envision a system where multiple objective metrics (individually studied to optimize correlation with groundtruth) are optimally combined into a single *Sector Priority Index (SPI)*. In this work, we focused on network measurements since, currently, sector rankings are based on a KPI scoring function. However, when defining the importance of a sector, one should consider also other information sources such as network sites profitability, CAPEX/OPEX investments, country regulations, etc. How to synthesize such SPI is an open question. Nevertheless, we believe that the optimization methodology discussed in this work can support the creation of such index.

Constant evolution: Network priorities and user QoE expectations are constantly changing as new usage paradigms emerge. Furthermore, the network is constantly evolving too, with 5G deployments being just a few years away. One of the implicit benefit of data-driven approaches is that they empower analysis automation. In our case, we envision a system that periodically adjust the hotspot score function parameters to capture long and short term trends. For instance, it can automatically incorporate seasonality effects. Similarly, automation can provide fine grained configuration across time (e.g., extract different optimization parameters to capture separately morning, afternoon, and night hotspots) or space (e.g., investigate separately residential areas from downtown districts). Notice that there is is no currently available solutions capable to achieve such a flexibility.

From KPIs to groundtruth: In this paper, we used weblogs to build an estimation about individual QoE components such as web throughput, delay, and video stalls. Building such a dataset requires some instrumentation (e.g., middleboxes), and exhibits significant computation complexity as we have to bridge the gap of per user (or per flow) performance with per-sector metrics. Therefore, it is prohibitive to run such processes in a continuous manner. Our work allows to use relatively sparse samples of groundtruth to associate them with KPIs that are already collected. Furthermore, other data sources can be used for groundtruth: active tests (e.g., drive tests), crowd-sourced data such as OOKLA [1] and user surveys, or even quality metrics coming through collaboration with application developers and CDNs.

VII. RELATED WORK

Metaheuristics have been used in the past for network optimization and planning [27]. For instance, simulated annealing and tabu search were used to allocate radio channels or to discover the minimum connected dominating set for wireless networks [29]. Randomized greedy algorithms have been used to find the optimal location of base-stations in order to maximize the traffic covered and minimize installation costs [4]. In [18] cloud services are ranked based on diverse KPIs such as cost, performance, stability, usability, elasticity, etc., using multi-criteria decision-making [14]. With respect to these works, we attempt to build a methodology that is able to estimate individual QoE components using a set of already collected KPIs. To the best of our knowledge, this is the first time that someone addresses this challenging problem.

Over the years there have been many attempts to understand how KPIs can be used to spot cellular performance bottlenecks [30] and network planning [24]. In [3], an iterative process of network deployment and monitoring through KPIs and drive tests was used to identify the optimal network configuration. Nokia engineers demonstrated how controlled experiments such as drive tests and on-site inspections, together with A-B testing, can be used to establish the relation between groundtruth and KPIs in order to optimize the network [21]. In [11], the authors show how drive tests can be used to identify the main KPIs that relate to QoS in a tetra network. Finally, field tests are used to build an empirical correlation between KPIs and throughput [23]. In addition to these works, there have been vendor recommendations on how to set performance thresholds and identify the worse performing sectors in the network. For instance, Huawei [35] also describes an iterative A-B process and provide recommendations for default values while Nokia [2] has extensively analyzed the meaning of each KPI. Our work breaks these long-term assumptions that QoE has to be build only with iterative field tests that are costly and inflexible. We propose a data-driven methodology that is able to automatically establish the relation between KPIs and the groundtruth in order to provide insights about underperforming sectors.

VIII. Conclusions

With the explosion of mobile internet traffic and the ever-evolving user expectations, it is of paramount importance for mobile operators to be able to quantify the performance of their network in terms of the delivered application services quality. We applied a novel data-driven methodology that builds upon the already collected sector KPIs and bridges them with different QoE metrics. By doing so, the system empowers operators with an automated methodology that provides better visibility on underperforming sectors. Moreover, our results indicate that the currently used solution that is based on thresholding is sub-optimal to identify critical sectors. This opens new areas or research for monitoring solutions enriching the quality and accuracy of the network performance indicators collected at the network edge.

References

[1] Speedtest.net by ookla - the global broadband speed test. http://www.speedtest.net/.

[2] N. Agarwal. Wcdma : Kpi analysis & optimization. *Nokia Technologies Co., Ltd.*, 2008.

[3] M. A. Alam. Mobile network planning and kpi improvement. In *Thesis, Linnaeus University, Department of Physics and Electrical Engineering, Sweden*, 2013.

[4] E. Amaldi, A. Capone, and F. Malucelli. Planning umts base station location: optimization models with power control and algorithms. *IEEE Trans. Wireless Communications*, 2(5):939–952, 2003.

[5] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.

[6] C. Blum and D. Merkle. *Swarm intelligence*. Springer, Berlin, Germany, 2008.

[7] L. Breiman and J. Friedman. *Classification and regression trees*. Chapman and Hall/CRC, Monterey, USA, 1984.

[8] P. Casas, R. Schatz, F. Wamser, M. Seufert, and R. Irmer. Exploring qoe in cellular networks: How much bandwidth do you need for popular smartphone apps? In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, AllThingsCellular '15, pages 13–18, New York, NY, USA, 2015. ACM.

[9] Cisco. Cisco visual networking index: Global mobile data traffic forecast update. *White Paper*, February 2016.

[10] M. Clerc. *Particle swarm optimization*. ISTE, London, UK, 2006.

[11] J. D. L. Delgado and J. M. R. Santiago. Key performance indicators for QOS assessment in TETRA networks. *CoRR*, abs/1401.1918, 2014.

[12] Ericsson. Measuring and improving network performance. *White paper*, 2014.

[13] G. D. et al. "Analysis of YouTube user experience from passive measurements". In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 260–267. IEEE, 2013.

[14] J. Figueira, S. Greco, and S. O. service). *Multiple Criteria Decision Analysis: State of the Art Surveys*. International Series in Operations Research & Management Science,. Springer New York,, New York, NY :, 2005.

[15] A. Finamore, M. Mellia, Z. Gilani, K. Papagiannaki, V. Erramilli, and Y. Grunenberger. Is there a case for mobile phone content pre-staging? In *CoNEXT '13*.

[16] P. C. Fourie and A. A. Groenwold. Particle swarms in size and shape optimization. In *Proc. of the Int. Workshop on Multidisciplinary Design Optimization*, pages 97–106, 2000.

[17] B. C. Gabriel. Managing the new mobile data network. *White Paper, Rethink Technology Research Ltd*, 2015.

[18] S. K. Garg, S. Versteeg, and R. Buyya. A framework for ranking of cloud computing services. *Future Generation Computer Systems*, 29(4):1012 – 1023, 2013. Special Section: Utility and Cloud Computing.

[19] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, Berlin, Germany, 2nd edition, 2009.

[20] T. K. Ho. Random decision forests. In *Proc. of the Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 278–282, 1995.

[21] H. Holma and A. Toskala. *WCDMA for UMTS: HSPA Evolution and LTE*. John Wiley & Sons, Inc., New York, NY, USA, 2007.

[22] L. Huawei Technologies Co. HUAWEI RAN KPI for performance management. *Instruction manual*, 2006.

[23] S.-V. Hn. Case studies of network planning for wireless broadband services hsdpa and wimax. *Master Thesis, OmniTele*, 2007.

[24] R. Kreher and K. Gaenger. *Key Performance Indicators and Measurements for LTE Radio Network Optimization*. John Wiley & Sons, Ltd, 2010.

[25] J. Laiho. *Radio Network Planning and Optimisation for Umts*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[26] S. Luke. *Essentials of metaheuristics*. Lulu, Raleigh, USA, 2nd edition, 2013.

[27] S. L. Martins and C. C. Ribeiro. *Handbook of Optimization in Telecommunications*, chapter Metaheuristics and Applications to Optimization Problems in Telecommunications, pages 103–128. Springer US, Boston, MA, 2006.

[28] A. R. Mishra. *Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G*. John Wiley & Sons, 2004.

[29] M. Morgan and V. Grout. Metaheuristics for wireless network optimisation. In *Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on*, pages 15–15, May 2007.

[30] A. Nika, A. Ismail, B. Y. Zhao, S. Gaito, G. P. Rossi, and H. Zheng. Understanding data hotspots in cellular networks. In *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QShine 2014, Rhodes, Greece, August 18-20, 2014*, pages 70–76, 2014.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] R. Poli, J. Kennedy, and T. M. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[33] C. E. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 3-4:441–471, 1904.

[34] S. Y. K. Ting and T. T. Chai. Wcdma network planning and optimisation. In *Telecommunication Technologies 2008 and 2008 2nd Malaysia Conference on Photonics. NCTT-MCP 2008. 6th National Conference on*, pages 317–322, Aug 2008.

[35] G. H. X Kaiping. Gsm kpi monitoring and improvement guide. *Huawei Technologies Co., Ltd.*, 2008.