Chroma: Learning and Using Network Contexts to Reinforce Performance Improving Configurations

Changhan Ge[†], Zihui Ge[‡], Xuan Liu[‡], Ajay Mahimkar[‡]
Yusef Shaqalle[‡], Yu Xiang[‡], Shomik Pathak[‡]
[†] The University of Texas at Austin, Austin, TX, USA [‡] AT&T, USA
chge@cs.utexas.edu {gezihui, xuan.liu, mahimkar, yusef.shaqalle, yxiang, shomik.pathak}@att.com

Abstract

Managing network configuration and improving service experience effectively is essential for cellular service providers (CSPs). This is challenging because of cellular networks' large scale and complexity, the wide variety of configuration parameters, and the performance impact tradeoffs resulting across multiple metrics and geographical locations. This paper focuses on learning and using network contexts to recommend performance-improving configurations. While learning contexts, one must carefully account for the configuration parameter dependency, performance impact confusion that can arise due to co-occurring unrelated changes, and uneven change deployment distribution across locations. We present a new solution Chroma that addresses the above challenges. Using real-world data collected from a large operational LTE and 5G cellular service provider, we thoroughly evaluate and demonstrate the efficacy of Chroma. We successfully trial Chroma on an operational cellular network and highlight its benefits in practical settings.

CCS Concepts

• **Networks** → **Network management**; **Wireless access points, base stations and infrastructure**; **Network experimentation**; • **Computing methodologies** → **Classification and regression trees**.

Keywords

Network configuration recommendation, Performance impact analysis, Context learning, Configuration change clustering

ACM Reference Format:
Changhan Ge[†], Zihui Ge[‡], Xuan Liu[‡], Ajay Mahimkar[‡], Yusef Shaqalle[‡], Yu Xiang[‡], Shomik Pathak[‡]. 2023. Chroma: Learning and Using Network Contexts to Reinforce Performance Improving Configurations. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3570361.3613256

1 Introduction

Emerging applications such as autonomous vehicles, drones, augmented and virtual reality (AR/VR), Internet of Things (IoT), and high-definition live impose enormous challenges on cellular networks to provide seamless connectivity and excellent quality of service (QoS). Cellular service providers (CSPs) leverage artificial intelligence and machine learning (AI/ML) techniques in multiple aspects of network design, planning, optimization, and operations. Configuration tuning and management plays a pivotal role across all aspects in improving QoS and user experience. Across generations of cellular networks (from 2G to 5G and beyond), the set of configuration parameters is extremely huge, and the tuning process is deemed as art by the service providers leveraging their domain knowledge and immense experience.

User experience and QoS can be measured using multiple metrics such as coverage, data throughput (downlink/uplink), voice call quality, admission failures, or call drops. Different configuration settings have the potential to have a different service performance impact. For example, uptilting a base station antenna improves coverage, but has the risk of call drops for far-away users due to radio signal quality. This results in a performance impact tradeoff that must be balanced carefully during the tuning process. Further, the same configuration setting can have a contrasting service performance impact across diverse locations. For example, a particular handover configuration may better serve highly mobile users, such as on high-speed trains or on freeways, compared to stationary users at residential locations. Thus, the problem is determining which configuration setting can provide an optimal impact tradeoff across different locations. This paper focuses on the challenging problem of tuning network configuration to improve service performance.

3GPP self organizing networks (SON) [1–5, 8, 13, 22, 41, 46, 49] automates multiple network configuration phases, such as capacity optimization and outage restoration. It focuses on discovering the best configuration for each location by leveraging performance metrics at the same location. However, it currently does not learn and incorporate the impacts across different locations. Furthermore, each SON algorithm focuses on a specific optimization objective (*e.g.*, throughput improvement), but does not consider the impact tradeoff across multiple metrics. Operational deployments of SON solutions still require network engineers to configure the range of parameter values and default template values. Different locations may still require different default values to reach better impact tradeoffs.

Reinforcement learning (RL) and contextual RL have recently been explored for network planning and optimization [43, 61]. We assert that RL, deep RL, or contextual RL cannot be applied easily for cellular network configuration because of the large number of parameters and the wide range of values that they can take. This makes it hard to conduct experimentation of the exponentially large configuration space (also referred to as ***exploration*** in the RL setup) and the time window needed to assess the performance impacts (typically multiple days) for each configuration. In addition, the seasonal behavior of performance metrics (*e.g.*, time of day, weekend versus weekday traffic pattern) introduces new challenges to assess the performance impact tradeoffs. Contexts derivable using network attributes such as hardware version, location morphology (urban, suburban, or rural), base station types (macro, pico, or distributed antennas), carrier frequency, downlink channel bandwidth, traffic load, user mobility, radio signal quality, and user location can be on the order of hundreds, and this makes it even more challenging to scale up exploration across many contexts.

We scope our problem to the ***exploitation*** phase (of RL) with the novel contribution of learning the context under which the configuration settings yield better performance tradeoffs. The contexts can be diverse for different sets of configuration parameters - *e.g.*, morphology could be vital to set the handover parameters, versus carrier frequency and channel bandwidth to decide the power settings. We leverage the massive configuration exploration conducted by network engineers across the network. Operational practice today is to leverage domain knowledge and experience to guide the configuration tuning across different parts of the network. What lacks today are effective ways to learn and use the contexts or location characteristics for network configurations. We believe the contexts would help with propagating the beneficial configuration to similar locations. We leave the exploration of configuration settings to new contexts for the future.

Challenges: Learning contexts for configuration with performance impacts brings in interesting sets of challenges:

- **Configuration parameter dependency:** There are certain groups of parameters that have to be changed close in time to have the desirable performance impact, whereas other parameters can be altered independently or in isolation. Identifying such groups of configuration parameters that have high dependency is nontrivial.
- **Performance impact confusion:** Assessing the performance impact of configuration changes is difficult because of the confusion caused by the presence of external factors such as seasonal traffic changes, foliage, weather, software upgrades, capacity updates or even other configuration changes at the same location. It is important to account for the impact confusion when learning the contexts, without which one may falsely infer a configuration setting to be inducing an improvement while it is not in reality.
- **Uneven distribution during configuration exploration:** Given k attributes and each of which can take m values, the number of contexts is m^k. Ideally, one has to explore configuration settings in each of the m^k contexts to derive the best context under which the configuration yields the best performance impact. In practice, however, the locations that have been explored are picked selectively by the network engineers using their domain knowledge and experience with location characteristics such as high traffic or high user mobility. This can result in an uneven distribution of samples for the learning of the contexts.
- **Configuration parameter assertions:** Certain configuration explorations may only be available with specific software releases or feature activations. After learning the contexts, it is important to enforce the parameter assertions, without which the system may not permit implementing the configuration changes.

Our solution: We propose a new solution Chroma that learns the context after the exploitation phase and recommends configuration changes using the contexts for similar locations. We formulate context learning as a classification problem with predictors constructed using network attributes and predictee is the configuration change with the performance impact (either an improvement, or not an improvement). We create network attributes using configuration attributes (*e.g.*, morphology, channel bandwidth), and LMRD attributes (traffic **L**oad, user **M**obility, **R**adio signal quality, and user **D**istance using time-series data). We use a composite quality index (CQI) which is a weighted function of the key performance indicators (KPIs) to capture the impact tradeoffs. Specifically, Chroma workflow includes

- **Configuration change grouping.** We fetch configuration changes across locations and group changes occurring at the same location and close in time using posterior probability and Jaccard's similarity. This captures configuration changes always deployed together and have a high likelihood of jointly inducing the performance impact.

- **Performance impact de-confusion and labeling.** We conduct a statistical pre/post impact comparison of performance across locations with configuration change (study group) and without (control group). The study versus control relative comparison helps eliminate the effect of external factors. We further de-confuse the performance impacts by identifying locations with only a single group of similar configuration changes or isolated configuration changes that are not part of any group. In other words, we eliminate locations that have coincidentally co-occurring groups of configuration changes for which we do not know which group is actually causing the impact.
- **Performance impact classification and configuration recommendation.** Once we have the accurate performance impact labels after the de-confusion, we prepare for the classification task. For the unobserved attributes, we fill-in the impact labels as inconclusive resulting in two classes of labels – one for improvement and other no impact which captures any degradation, no impact or inconclusive. We then use RIPPER [16] and decision tree classifiers to learn the context or the set of attributes that best explain the performance improvements. Given the contexts or rules output from the classifiers, we recommend configuration changes to the locations that match the contexts but have different configuration settings.
- **Parameter value spatial clustering.** To enforce parameter assertions due to software versions or certain features, we use the configuration snapshots (instead of configuration changes) to derive similar parameter values across locations. We use the spatial clusters to then append our configuration recommendation with more configuration changes that would have otherwise violated the assertions.

Chroma can be viewed as a first step in improving configuration exploitation with the goal of optimizing performance experience using changes implemented in the past. Change clustering and impact de-confusion are key components to attain our goal. We believe Chroma will serve as a foundational component to conduct a better configuration exploration.

Our contributions:

- In Section 3, we use a massive data set collected from a large LTE and 5G cellular service provider to quantitatively highlight the existence of impact tradeoffs across multiple performance metrics and locations, the application of configuration changes close in time that causes impact confusion, the large search space of configuration parameters and range of their values, and finally the uneven distribution of configuration explorations. This strongly motivates the design of our solution Chroma.
- We present the design and implementation of our new solution Chroma (in Section 4) to automatically learn the context using configuration changes, their performance impacts and a wide variety of network attributes. Chroma recommends configuration changes to locations matching the contexts with the intention of improving the performance impact tradeoffs.
- We conduct an in-depth and thorough evaluation of Chroma using a large dataset collected from a very large live cellular network (Sec. 5) and demonstrate the importance of configuration change grouping and impact de-confusion.
- Finally, we successfully conduct trials of our recommendations (210K+ configuration changes) across 17 major markets (can also be viewed equivalent to US states). On 80%+ locations, we notice improvements to performance impact tradeoffs highlighting the practical benefits of Chroma. In some scenarios, we notice that the location characteristics (*e.g.*, co-existence or not of LTE and 5G) cause no improvements in the tradeoffs – this was considered acceptable by the network engineers and decision was made not to roll-back any changes since this prepares them for future changes in network behaviors.

Ethics Statement: No personally identifiable information (PII) is used. This work does not raise any ethical concern.

2 Related work

There has been a surge of significant works from industry and academia in the area of configuration synthesis [6, 10–12, 17, 20, 32, 36, 45, 48, 52–54] , verification [9, 12, 18, 19, 21, 24–26, 29, 30, 34, 35, 38, 39, 44, 47, 50, 51, 58, 59], and tuning [1–3, 8, 15, 22, 23, 27, 28, 33, 42, 43, 57, 60, 62]. Configuration synthesis aims to start with high-level intent specification and automatically generate the low-level node-specific configuration settings. Since networks keep evolving, it is important to verify and manage configuration over time. Verification works focus on comparing network properties such as reachability to expectation and raise alarms in case of violations. Configuration tuning involves the process of changing the configuration with the goal of optimizing a reward function such as service performance. Next, we describe works related to configuration tuning and highlight the differences and benefits of Chroma.

Configanator [43] and ConfigTron [42] proposes to use a contextual multi-armed bandit approach to dynamically learn the optimal configuration parameters to improve performance experience in CDN networks. CherryPick [7] and Metis [31] use Bayesian optimization to identify the best configuration in cloud environments for big data analytics workload. Dremel [60] proposes an interesting adaptive configuration tuning approach to dynamically adjust the arms within multi-armed bandits and discover the optimal configuration for improving performance of RocksDB KV-store. Auric [36] uses collaborative filtering across network configuration attributes and parameters to automatically generate configuration for new carriers added in cellular networks. Aurora [33] combines configuration and LMR attributes (load, mobility, radio conditions) to learn the setting configured across majority nodes in the network with the assumption that majority setting leads to an enhanced performance experience to the users. Self-tune [28] uses online reinforcement learning to dynamically tune multiple configuration parameters jointly in live operational environments to improve performance for large cluster networks. OtterTune [55] uses unsupervised learning approaches to tune database configurations. SmartConf [56] focuses on automatically adjusting performance-sensitive configurations to attain operating constraints in distributed systems such as Cassandra, HBase, HDFC and MapReduce.

We present a comparison of the works closest to Chroma in Table 1. We use performance (feedback, and tradeoff), configuration parameters (large space, and grouping), and attributes (contextual analysis, learning, and the number of contexts per parameter) to show that Chroma is the only one that can tackle all of them. Performance feedback is incorporated into the problem formulation by Configanator, Self-tune, Dremel to discover the optimal configuration setting. However, Auric and Aurora do not model performance and are primarily driven by the majority setting within the context. All the existing works focus on a single performance metric and do not capture the tradeoff across multiple metrics. Configanator and Self-tune deal with a small number of configuration parameters. On the other hand, Auric, Aurora and Dremel can handle the large number of parameters and their ranges. Parameter grouping is important to capture the dependencies and Dremel is the only existing works that incorporates the dependency by grouping or clustering the parameters. For contextual analysis, Auric, Aurora and Configanator determine the best configuration setting for each context, however, Self-tune and Dremel aim to discover the setting for the whole network instead of for each context. Context in Configanator is input based on the attributes (*e.g.*, traffic, or the number of connections) and is common across all parameters, but Auric and Aurora automatically learn the context for each parameter. Thus, Auric and Aurora can have different contexts for different configuration parameters. This is something we also need for our problem setup in Chroma where different contexts might yield different performance impacts across different configuration settings.

Following challenges limit the straightforward application of the approaches above and is thus the key distinction to Chroma. (i) ***Large number of configuration parameters:*** All of the existing approaches leveraging reinforcement learning or supervised learning operate on a few handful configuration parameters and thus the search space is tractable. In LTE and 5G cellular networks, thousands of parameters with huge ranges of their values make it very hard to apply RL. (ii) ***Context is hard to specify:*** Contexts in contextual RL is often input. However, in our case, it is hard to specify the context under which the configuration setting will result in performance improvements. Even with domain knowledge and years of experience, network engineers find it as a mission impossible task. (iii) ***Risk of performance degradation:*** Assessing performance impact of configuration changes requires on the order of days because of the variability and seasonality in performance time-series that could cause incorrect inferences. Any severe degradation could be detrimental to user experience and retention on the network. Thus, it is not easy to conduct arbitrary trials on the operational network.

Feature	Auric [36]	Aurora [33]	Configanator [43]	Self-tune [28]	Dremel [60]	Chroma
Performance feedback			✓	✓	✓	✓
Performance tradeoff						✓
Large parameter space	✓	✓			✓	✓
Parameter grouping					✓	✓
Contextual analysis	✓	✓	✓			✓
Context learning	✓	✓				✓
Contexts per parameter	✓	✓				✓

Table 1: Related work comparison.

3 Background and motivation

In this section, we first briefly introduce LTE and 5G cellular network, followed by analysis conducted using one-year worth of real-world data collected from live LTE and 5G cellular networks to highlight the motivation behind the design of Chroma and challenges faced in building our solution.

3.1 LTE and 5G cellular network

3.1.1 Topology: As shown in Fig.1, the nation-wide cellular network is divided into several regions, and these regions are subdivided into market clusters and then into markets. Each market consists of multiple geographical areas tagged by tracking area codes (TAC) respectively, and is operated by a team of network engineers responsible for managing the base stations (BS) - called eNodeB (eNB) in LTE and gNodeB (gNB) in 5G. A gNB is either non-standalone (NSA) or standalone (SA). An NSA gNB has a distinct data plane but is normally co-located with an eNB (together as one BS site) and share the access control to LTE core network, while SA gNB operates with independent data plane and access control to 5G core network. Each BS site is labeled with

Type	Attribute	Example Values	Attribute	Example Values
Configuration Attributes	Morphology	Rural, Suburban, Urban, etc.	Cell Type	Emergency, C-RAN, etc.
	BS Type	Pico, Micro, Macro, iDAS, oDAS	Bandwidth	5, 10, 15, 20 MHz, etc.
	Frequency Bands	**LTE**: LTE-A<LTE-B<LTE-C<LTE-D<LTE-E **5G**: 5G-A<5G-B<5G-C<5G-D<5G-E<5G-F	MIMO Mode	Dynamic Open Loop MIMO Closed Loop MIMO, etc.
	Hardware	**LTE**: HW_V1(.1-3),2(.1,2),3(.1-6),4(.1-5),5(.1-4),6(.1-9) **5G**: HW_V1(.1-6),2(.1-6),3(.1-3),4(.1,2)	Software	**LTE**: SW_V1(.1,2), 2(.1-5), 3(.1-9) **5G**: SW_V1(.1-6), 2(.1-12),3,4,5,6
Type		**Example Attribute**		**Example Values**
LMRD Attributes	Traffic <u>L</u>oad	Number of RRC Sessions, Uplink/Downlink PDCP Volume Uplink/Downlink PDCCH Utilization, PRB Utilization, etc.		Very Low, Low, Medium, High, Very High, etc.
	User <u>M</u>obility	Intra/Inter-frequency Handover, etc.		
	<u>R</u>adio Signal	RSRP, RSRQ, SNR, etc.		
	<u>D</u>istance	95% User Distance, Tower Height, etc.		

Table 2: Examples of Network Attributes of LTE and 5G Datasets

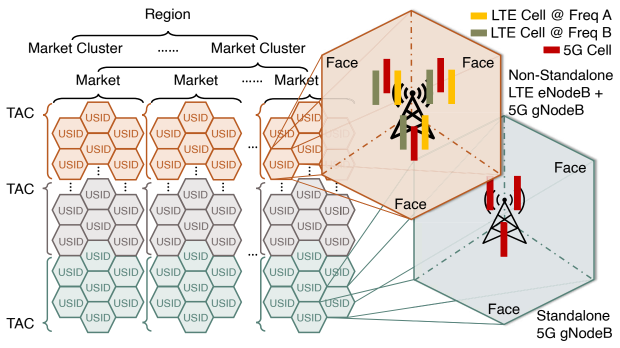


Figure 1: Nation-wide Cellular Network Topology

Service	Type	KPI (k_i)
Data	Access	Data Access Failure Rate
	Drops	Data Drop Sessions Rate
	Speed	Data Throughput
	Coverage	No Data Service Time
Voice	Access	Voice Access Failure Rate
	Drops	Voice Drop Sessions rate
	Coverage	No Voice Service Time

Table 3: CQI Decomposition

one USID (universal site identifier). Each BS typically has multiple faces dividing the space into different sectors, and each face can operate multiple cells on various frequency bands (*e.g.*, 5G mmWave operating between 24 to 40 GHz versus mid/low frequencies on under 6 GHz). In a cellular network, a cell is the fundamental functional unit where the configurations are implemented on.

3.1.2 Configuration parameters: The LTE and 5G networks offer a wide variety of configuration parameters ranging from admission control, handover management, carrier aggregation, scheduler, interference mitigation, power and MIMO control. Each parameter either takes an enumerated list of values (including binary to capture feature activations), or a range of values (for example, threshold for RSRP of serving cell can take values between -140 dBm and -43 dBm). We list description of a few example parameters below:

- **threshold3InterFreqQci1, threshold3aInterFreqQci1** are the thresholds for RSRP of serving cell and neighboring cell when the user equipment has voice over LTE bearer. If the RSRP of the serving value is less than threshold3InterFreqQci1 and the RSRP of the neighbor cell is greater than threshold3aInterFreqQci1, then handover is triggered (A5 event).
- **thresholdRsrpEndcFilt, thresholdRsrqEndcFilt** are the thresholds for filtering target cells out of the reported A5 event based on RSRP/RSRQ values due to EN-DC (dual connectivity between LTE and 5G) capability-based handover. This parameter is important for post-processing the A5 report.
- **qrxLevMin** is the minimum RSRP that a base station will establish or maintain a connection. If the signal strength falls below this level, the base station will not accept the connection, and the user will need to try to connect to another base station with a stronger signal.

3.1.3 Network attributes: Similar to Aurora [37], we extract the ***configuration*** and ***LMRD attributes*** from network inventory snapshots to characterize each cell. The configuration attributes mainly describe the node-related features, such as hardware/software version, morphology, frequency bands, channel bandwidth, MIMO mode, base station type, and cell type. Meanwhile, the LMRD attributes summarize the manner of serving area in which traffic **<u>l</u>**oad includes number of radio resource control (RRC) sessions, packet data convergence protocol (PDCP) volume, user **<u>m</u>**obility consists of inter/intra-frequency handovers, **<u>r</u>**adio signal quality comprises of received signal power/quality (RSRP/Q), signal-to-noise-ratio (SNR) and physical downlink control channel (PDCCH) utilization. Different from Aurora, we also introduce the **<u>d</u>**istance metrics, such as UE distance and tower height, to better capture the feature of the surrounding environment. Examples of network attributes and their potential values are listed in Table 2.

3.1.4 KPI and CQI: The cellular service provider monitors a list of KPIs for each cell every 15 minutes, such as voice/data accessibility (success rate of call/data establishment) and

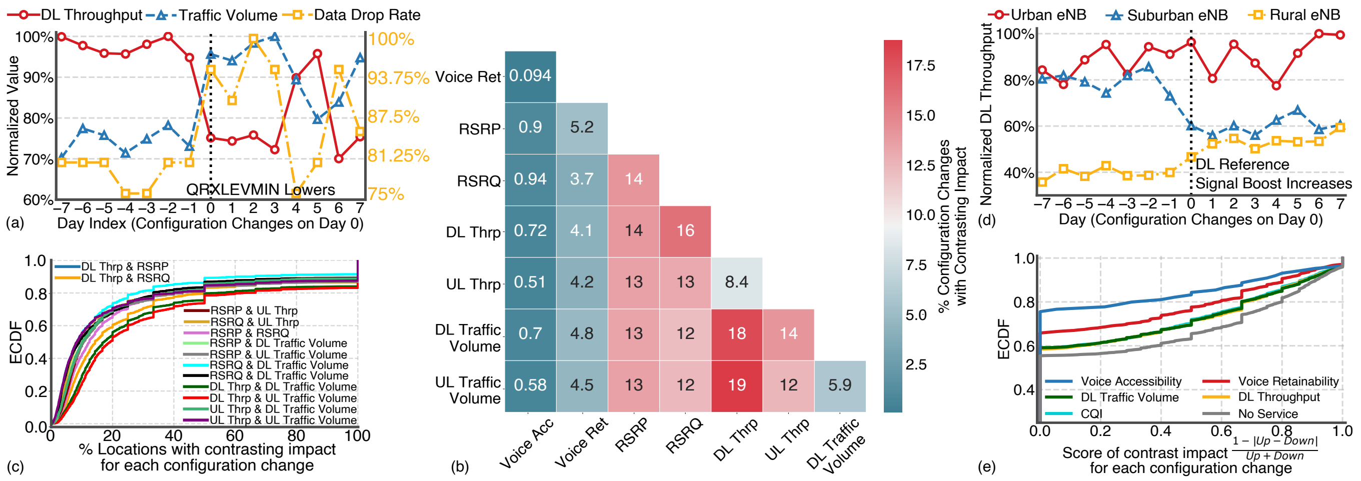


Figure 2: Impact tradeoff widely exists in cellular network. (a) An example of impact tradeoff across KPI metrics. (b) % Configuration changes with contrasting impact across pairs of KPI metrics. (c) % Locations with contrasting impact across pairs of KPI metrics. (d) An example of impact tradeoff across locations. (e) Score of contrast impact on KPIs and CQI

retainability (success rate of call/data completion), downlink/uplink throughput, intra/inter-frequency handover success rate, etc. Meanwhile, to assess the impact tradeoff across KPIs, the service provider uses a composite index CQI, a weighted function of KPIs, to comprehensively reflect the service quality. The CQI is used by the network engineers to continuously monitor the health of the network and tune the network configuration parameters. CQI is calculated as $\sum_i W_i e^{(KPI_i * E_i)}$, where W and E is weight and exponential constant, respectively. The KPIs used to calculate the CQI are as shown in Table 3. Due to proprietary reasons, we omit the values of W and E. We compare the KPIs as well as CQI before and after a configuration change to determine its performance impact. In Chroma, we use CQI as our optimization goal because it captures the tradeoff across multiple performance metrics. One can easily modify the target CQI if needed to adopt to different optimization objectives.

3.2 Motivation and challenges

Impact tradeoff across performance metrics: We observe in the operational network that altering a configuration may improve some key performance indicator (KPI) metrics but negatively impact others. This behavior is wide spread across the network. Although it is not necessarily always detrimental, a configuration change is deemed unfavorable when it disrupts the tradeoff among the performance metrics. Fig. 2(a) presents a typical example wherein the qrxLevMin threshold (minimum RSRP for users to maintain a connection with the base station) was reduced that resulted in improving the coverage and increased traffic volume, but also caused a minor degradation in downlink data throughput and increased data drop rates. The reason for such a tradeoff is increasing the coverage with increase in the number of users implies a higher physical resource block utilization that would then decrease the data throughput attained by the users. Furthermore, for edge users, the chance of a data drop is higher because of the higher distance that the signal has to propagate. This illustrates the importance of seeking a configuration setting that enhances the overall performance impact without impairing individual or multiple metrics.

Fig. 2(b) captures the prevalence of performance impact tradeoff across pairs of metrics observed across all the configuration changes and nodes in the network. The numeric score for each pair of metric is calculated using the percentage of locations that have a contrasting performance impact such as improvement in metric A versus degradation in metric B. The numbers in Fig. 2(b) capture the percentage of changes with a high contrasting impact score. Fig. 2(c) shows a CDF on each variant of configuration change (*e.g.*, parameter X from 5 to 10) for the percentage of locations with contrasting performance impact as shown in Fig. 2(b). As can be seen, different metrics can be impacted in contrasting directions for the same configuration change.

Impact tradeoff across locations: The same configuration change applied at different locations may have different consequences. As illustrated in Fig.2(d), after the same downlink (DL) reference signal boost change launched on 3 eNodeBs at different locations, the DL throughput on urban eNodeB remains the same, but decreases on suburban eNodeB, and increases on rural eNodeB. The reason for this contrasting impact is that the rural locations benefit significantly because coverage gaps are more pronounced, however, while urban and suburban locations can benefit from signal boost as well, the signal boost may have detrimental effects due to increased interference from neighboring cells in a densely populated area. Hence, the distinct characteristics of different locations should also be examined and considered when

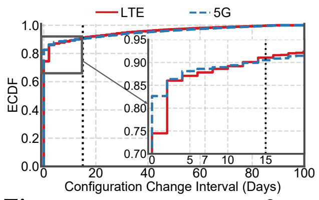


Figure 3: Concurrence of configuration changes.

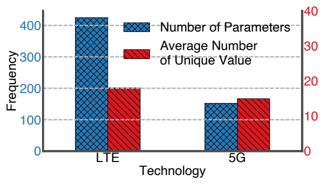


Figure 4: LTE and 5G has a large parameter search space.

we learn the configuration settings to improve service performance. In order to understand if the impact tradeoff across locations is wide spread across network, we show a CDF in Fig.2(e) on each variant of configuration change (*e.g.*, parameter X from 5 to 10) for the contrast impact scores across locations. The contrast impact score for each configuration change is calculated using the number of locations that experience improvement (up) as compared to degradations (down). As can be seen, more than 60% of configuration changes experience different performance impacts across different locations. This highlights the importance of learning which location types (*i.e.*, contexts) experience a better performance impact tradeoff for the configuration setting.

Configuration changes are applied close in time: The performance impact of configuration changes can be hard to identify if configuration changes are applied in close proximity. Fig.3 demonstrates that nearly 90% of both LTE and 5G configuration changes occur with other changes within a 15-day timeframe (7 days before and after the change) on the same cell, making it difficult to pinpoint which configuration change is responsible for the impact. This highlights the importance of grouping related configuration changes and de-confusing the performance impacts.

Large search space to find the best configuration: Finding the optimal configuration is non-trivial. As shown in Fig.4, either LTE or 5G cells feature hundreds of parameters to be tuned, and each one of them with tens of unique values yields an exponentially large search space across parameters. This also re-validates that exhaustively trialing all potential configurations is impossible.

Uneven distribution of trials: Markets vary largely by size and the number of trials (configuration changes during the exploitation phase). Generally, markets with more cells may frequently tune configurations to optimize the service quality, whereas some smaller markets may not have enough trials so that the local network may be sub-optimal or even under-performing. This highlights the need to leverage configuration knowledge from all locations to improve under-trialed markets. Furthermore, it is important to carefully account for the uneven distribution during the learning of network contexts. For example, if trials were conducted only in urban locations and none in rural, then one may incorrectly infer that the improvement may be prevalent across all locations irrespective of the underlying morphology.

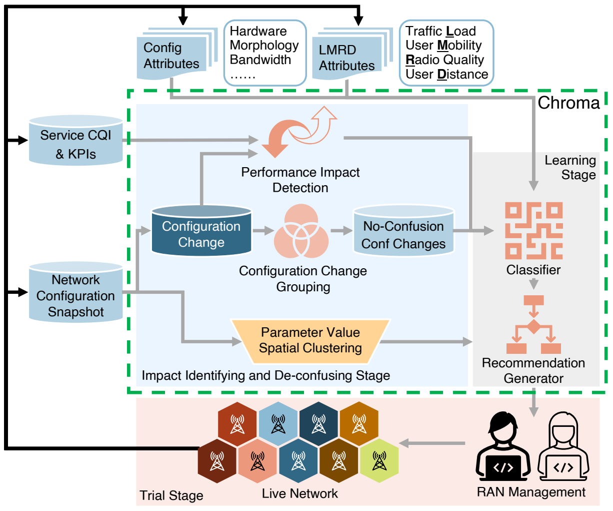


Figure 5: Chroma Workflow

4 Chroma design and implementation

We show the overview of Chroma in Fig.5. We design Chroma to be a closed-loop system with three stages of operation: (i) identifying changes and their performance impacts, (ii) learning and using context to recommend changes across the network, and (iii) trialing the changes after seeking approvals from the network engineers. We introduce each of them in detail in this section.

4.1 Preliminary: key terms

- A cell d works with a set of *configuration parameters* $C = \{C_1, C_2, \cdots, C_i\}$, in which the number of configuration parameters varies by technology and vendor. Each C_i has a set of unique values $\{c_i^1, c_i^2, \cdots, c_i^n\}$, in which the number of unique values varies by C_i. Some C_i may share dependency with other $C_j, j \neq i$. The value of C_i configured on cell d is denoted as $c_{i(d)}$.
- A *configuration change* is defined as "the value of C_i changes from c_i^n to c_i^m", written as $c_i^{n \to m}$.
- A cell can be characterized with a set of *network attributes* $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_i\}$, in which the number of attributes varies by technology. Each attribute $\mathcal{A}_i$ has a set of unique values $\{a_i^1, a_i^2, \cdots, a_i^n\}$, in which the number of unique values varies by $\mathcal{A}_i$.

Relationship between configuration and attribute. Recall that configuration parameters are entities changed at the cells to improve performance, versus attributes can be viewed as characteristics of the cells. Details of configuration parameters and attributes are discussed in Sec 3.1.2 and 3.1.3.

4.2 Configuration change grouping

As discussed in Sec. 3.2, multiple configuration changes may be applied on the same day or very close in time, making it difficult to distinguish their respective effects. Nevertheless, it is observed that (I) some of the configuration changes tend to occur simultaneously, and (II) some configuration changes always trigger other configuration changes. These two observations may be a result of configuration regulations or experience of network engineers. With this knowledge, it is possible to group some configuration changes together and treat them as a single entity. For instance, for (I), reducing the uplink channel bandwidth (ULCHBW) always goes hand-in-hand with increasing downlink channel bandwidth (DLCHBW), and vice versa, while for (II), reducing ULCHBW incurs reducing maximum number of uplink UE (MAXNUMUEUL), but reducing MAXNUMUEUL does not necessarily require reducing ULCHBW. In both examples, we should group the two configuration changes as one.

4.2.1 Grouping two configuration parameters: Suppose there are two parameters C_i and C_j, and $\mathbf{O}(\cdot)$ denotes the set of date-cell pairs on which the parameter is changed. The observation (I) can be expressed as $\mathbf{O}(C_i) = \mathbf{O}(C_j)$ and (II) can be expressed as $\mathbf{O}(C_i) \subseteq \mathbf{O}(C_j)$ or $\mathbf{O}(C_i) \supseteq \mathbf{O}(C_j)$. Note that (I) is a special case of (II) when $\mathbf{O}(C_i) \subseteq \mathbf{O}(C_j)$ and $\mathbf{O}(C_i) \supseteq \mathbf{O}(C_j)$. We quantitatively identify these cases by calculating the posterior probability between C_i and C_j, as expressed below:

$$P(C_i|(C_j) = \frac{|\mathbf{O}(C_i) \cap \mathbf{O}(C_j)|}{|\mathbf{O}(C_j)|}, P(C_j|(C_i) = \frac{|\mathbf{O}(C_i) \cap \mathbf{O}(C_j)|}{|\mathbf{O}(C_i)|}$$

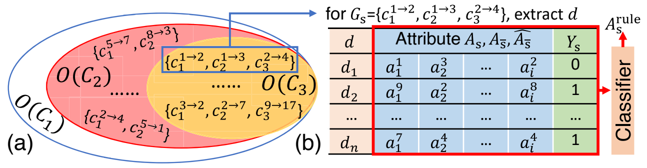
where $|\cdot|$ represents the cardinality, *i.e.*, number of elements. Ideally, C_i and C_j should be grouped together either when $\mathbf{O}(C_i) \subseteq \mathbf{O}(C_j)$, *i.e.*, $P(C_j|C_i) = 1$, or $\mathbf{O}(C_i) \supseteq \mathbf{O}(C_j)$, *i.e.*, $P(C_i|C_j) = 1$. But this method is highly likely to yield false positive when $|\mathbf{O}(C_i)| \gg |\mathbf{O}(C_j)|$ or $|\mathbf{O}(C_i)| \ll |\mathbf{O}(C_j)|$, or false negative when the posterior probabilities are very close but not equal to 1 due to data loss issue. To improve the clustering quality, we also introduce the Jaccard similarity metric between $\mathbf{O}(C_i)$ and $\mathbf{O}(C_j)$ as expressed below:

$$J(\mathbf{O}(C_i), \mathbf{O}(C_j)) = \frac{|\mathbf{O}(C_i) \cap \mathbf{O}(C_j)|}{|\mathbf{O}(C_i)| + |\mathbf{O}(C_j)| - |\mathbf{O}(C_i) \cap \mathbf{O}(C_j)|}$$

which effectively measures the likelihood of tuning C_i and C_j together. Given the potential data issue, we empirically cluster two configuration changes together if they satisfy the following condition

$$J(\mathbf{O}(C_i), \mathbf{O}(C_j)) > 0.3 \wedge (P(C_i|C_j) > 0.9 \vee P(C_j|C_i) > 0.9) \quad (1)$$

4.2.2 Grouping multiple configuration parameters Note that the posterior probability or Jaccard similarity only measures the relationship between two sets, which is insufficient for clustering multiple configuration parameters (> 2) together. Meanwhile, calculating the posterior probability or similarity among multiple sets is non-trivial since the computation time grows exponentially with the number of configurations as $\sum_{k=2}^{n} \frac{n!}{(n-k)!k!} \approx 2^n$. To relax this problem, we approximate it by leveraging the necessary condition that every two member in a group should satisfy the condition in Eq.1. We build an adjacency matrix of which each entry is either 1 or 0 to specify if a pair of configuration parameters satisfies Eq.1. Hence, the multi-configuration change grouping problem becomes to find the maximum cliques[1] based on the adjacency matrix, which is NP-complete and can be solved using standard Bron–Kerbosch algorithm [14] with a complexity of $O(3^{n/3})$. In addition to that, we also add a list of groups by combining the groups sharing members but not including each other, in which the posterior probability of each element given any of the common members is large than 0.9. In this case, a configuration parameter can belong to multiple groups. We denote a set of configuration parameters grouped by their change manner is denoted as $\mathcal{G} = \{C_i, \cdots, C_j\}$.



$\{c_1^{5\to7}, c_2^{8\to3}\}$
$\{c_1^{1\to2}, c_2^{1\to3}, c_3^{2\to4}\}$
$O(C_2)$
......
......
$O(C_3)$
$O(C_1)$
$\{c_1^{2\to4}, c_2^{5\to1}\}$
$\{c_1^{3\to2}, c_2^{2\to7}, c_3^{9\to17}\}$
(a)
(b)
for G_s={$c_1^{1\to2}, c_2^{1\to3}, c_3^{2\to4}$}, extract d
A_s^{rule}
d
Attribute $A_s, A_{\bar{s}}, \widehat{A_{\bar{s}}}$
Y_s
d_1 a_1^1 a_2^3 ... a_i^2 0
d_2 a_1^9 a_2^2 ... a_i^8 1
... ...
d_n a_1^7 a_2^4 ... a_i^4 1
Classifier

Figure 6: Example of subgroup and classification

We give an example as shown in Fig.6(a). Since $O(C_3)$ is a subset of both $O(C_1)$ and $O(C_2)$, and $O(C_2) \subset O(C_1)$, hence, we will have 2 groups $\{C_1, C_2, C_3\}$ and $\{C_1, C_2\}$.

4.3 Performance impact detection and de-confusion

4.3.1 Statistical impact detection To identify the performance impact of configuration changes, we conduct a statistical comparison of performance metrics k-day before and after each change between the study group (locations where the change is implemented) and the control group (locations where the change is not implemented around the same time). By comparing the study and control groups, we can eliminate the effect of external factors. Our statistical impact detection approach is borrowed from [35]. Next, we quantify the performance impact using a relative median difference in percentage as follows: $\delta = \frac{\tilde{\boldsymbol{x}}_{\text{after}} - \tilde{\boldsymbol{x}}_{\text{before}}}{\tilde{\boldsymbol{x}}_{\text{before}}} \times 100\%$, where $\boldsymbol{x}$ can be the time series of any KPI or CQI, and $\tilde{(\cdot)}$ represents the median of the samples observed. We carefully choose $k = 7$ as the observation period to cover the weekly fluctuation. Meanwhile, the outcome of different configuration changes on different cells may vary largely due to the nature of each configuration parameter and the characteristics of the cell configured. Empirical evidence suggests that the configuration change is believed to have an effective impact if $|\delta| > 0.5\%$, where $|\cdot|$ calculates the absolute value. Accordingly, the impact label Y is assigned "Increase" if $\delta > 0.5\%$,

[1]A subset of vertices in an undirected graph such that each pair of two vertices are connected. It is also called complete subgraph.

"Decrease" if $\delta < -0.5\%$, or "No Impact" if $|\delta| < 0.5\%$, which is further mapped to 1, -1, and 0, respectively.

We use CQI as our target metric to effectively capture the performance impact tradeoff. However, we notice that CQI captures no service coverage, but currently does not incorporate traffic volume in bytes. Thus, we assign the final impact label by jointly considering the CQI and traffic volume. Specifically, an "Increase" label is assigned if either CQI or traffic volume improves without any degradation in the other. A "No Impact" label is assigned when there is no impact or when the impact on the two metrics is in contrast. Lastly, a "Decrease" label is assigned when at least one of the two metrics degrades without any improvement in the other. The numerical label is determined by $sgn(Y_{\text{CQI}} + Y_{\text{Traffic}})$, where $sgn(v) = \frac{v}{|v|}$ is the sign function.

4.3.2 Performance impact de-confusion Given the groups of configuration changes, we separate the ***no-confusion dataset*** from the ***all dataset***, in which each cell on a certain day only has configuration changes from group $\mathcal{G}$ and there is no other configuration changes happened 7-day before and after that change. In this way, we ensure that the impact is only caused by $\mathcal{G}$. Thus, we throw away samples with co-occurring configuration changes belonging to different groups for our no-confusion data. These "confused" samples are challenging to use in our impact classification because it is unclear which of the configuration changes actually results in an improvement to performance. Using real-world data, we observe sufficient samples in our no-confusion dataset to enable reliable classification using our network attributes.

4.4 Performance impact classification: learning contexts

To identify which set of configuration changes is favorable for which type of cell, it is essential to leverage the classifier that can find the relationship between the network attributes and impact of configuration change. To achieve this, we first divide each group of configurations $\mathcal{G}$ into several subgroups $\mathcal{G}_\text{s} = \{c_i^{n\to m}, \cdots, c_j^{p\to q} : i \neq j, m \neq n, p \neq q\}$ based on the existing combinations of configuration changes. For each $\mathcal{G}_\text{s}$, we extract a matrix of attributes $\mathcal{A}_\text{s}$ in which each row is a vector of the network attributes of a cell with $\mathcal{G}_\text{s}$ applied, along with their corresponding impact labels Y_s. We then augment the dataset by including the data from its conjugate, *i.e.*, the subgroup with configuration changes of inverse direction as $\mathcal{G}_{\bar{\text{s}}} = \{c_i^{m\to n}, \cdots, c_j^{q\to p}\}$, whose attributes is kept as normal $\mathcal{A}_{\bar{\text{s}}}$ but the impact labels are inversed as $-Y_{\bar{\text{s}}}$. Meanwhile, to prevent the classifier recommend unseen attributes, we generated a batch of missing attribute $\widehat{\mathcal{A}_\text{s}}$ data with "Inconclusive" labels $\widehat{Y_\text{s}}$. Each value of an attribute in $\widehat{\mathcal{A}_\text{s}}$ is picked from all the possible values other than the values exist in $\mathcal{A}_\text{s}$ and $\mathcal{A}_{\bar{\text{s}}}$. Since our goal is to determine the cells that would be benefited from $\mathcal{G}_\text{s}$ being applied to them, *i.e.*, we care more about the "Increase" class, we merge the "Decrease", "No Impact" and "Inconclusive" classes as one. Then, the problem becomes a binary classification task. At this stage, for each $\mathcal{G}_\text{s}$, we train a classifier with $[\mathcal{A}_\text{s}, \mathcal{A}_{\bar{\text{s}}}, \widehat{\mathcal{A}_\text{s}}]$ as the input and impact labels $[Y_\text{s}, -Y_{\bar{\text{s}}}, \widehat{Y_\text{s}}]$ as the output. An example of classification data is shown in Fig.6(b).

Since the input data of the network attributes are discrete values, and the embedding space learned by the classifier should be comprehensible for the sake of generating the configuration recommendations, a straightforward approach is to use the decision tree classifier. However, the decision tree is infamous for over-fitting to training set, biased on the class with more data samples, and instability to noisy data. To mitigate this issue, we also leverage the rule learning algorithm RIPPER [16], which constructs a list of decisions by iteratively adding rules to the list and then removing logic components that are already considered by those rules. For a given $\mathcal{G}_s$, we pick the rule-set generated by either RIPPER or Decision Tree depending on which one yields a higher classification accuracy on all test-set as well as the "Increase" Class. Then, the branches in decision tree or the rule-set from RIPPER which produces "Increase" class is translated to logic clauses in disjunctive normal form (DNF), denoted as $\mathcal{A}_\text{s}^\text{rule}$. A representative example is shown below

$(\texttt{CELLTYPE} = \texttt{CRAN} \land \texttt{DL\_PRB} = \texttt{MEDIUM} \land \texttt{DISTANCE} = \texttt{LOW}) \lor$
$(\texttt{CELLTYPE} = \texttt{VRAN} \land \texttt{TOWER\_HEIGHT} = \texttt{MEDIUM} \land \texttt{PDCCH\_UTIL} = \texttt{MEDIUM}) \lor$
$(\texttt{TOWER\_HEIGHT} = \texttt{HIGH} \land \texttt{SNR} = \texttt{LOW} \land \texttt{MORPHOLOGY} = \texttt{SUBURBAN}) \lor \cdots$

$\mathcal{A}_\text{s}^\text{rule}$ is our context of network attributes for which the group of configuration changes have resulted in a performance improvement across the network. A cell is considered a candidate for applying $\mathcal{G}_\text{s}$ if it possesses network attributes that satisfy any of the inner-nested condition-combinations in $\mathcal{A}_\text{s}^\text{rule}$. We denote these candidate cells as $\mathcal{S}(\mathcal{A}_\text{s}^\text{rule})$.

4.5 Parameter value spatial clustering

Other than clustering configurations based on their change behavior and joint impact on performance, we also run spatial clustering by evaluating parameter value distributions in a single day. The reason for spatial clustering is that certain parameter dependencies cannot be discovered using change clustering alone especially when dependent changes may be implemented spanning longer time intervals such as days or only with new feature activations. We denote $\text{S}(C_i)$ as the set of cells that parameter C_i is configured on. The goal of spatial clustering is to identify parameter clusters $\mathfrak{C} = \{C_i, \cdots, C_j\}$ whose value distribution on the same set of cells exposes strong dependency. As configuration on cells can be quite heterogeneous, i.e, $\text{S}(C_i)$ may be very different from $\text{S}(C_j), j \neq i$. While spatial clustering evaluates parameter values on the same set of cells, parameters will need to go through stage-0 clustering over $\mathrm{S}(C_i)$ with k-means via word2vec[40]. To obtain parent clusters $\mathfrak{C}_{\text{parent}}$, number of clusters in k-means is tuned so that parameters in each $\mathfrak{C}_{\text{parent}}$ has a reasonable intersection of cell sites achieved. Then, for each $\mathfrak{C}^k_{parent}, k \leq K$, where K is the total number of parent clusters achieved, we cluster parameters again based on parameter value distributions on the set of cells in $\mathfrak{C}^k_{\text{parent}}$. To capture the dependency of value distribution among parameters in each $\mathfrak{C}^k_{\text{parent}}$, we introduced mutual information, which measures the mutual dependency between two variables C_i and C_j, i.e., the amount of information we can obtain of C_j from observing value distribution of C_i. Suppose $p(C_i)$ is the probability distribution of C_i. The mutual information between variable C_i and C_j can be defined as the relative entropy between the joint distribution $p(C_i, C_j)$ and the product of their marginal distributions:

$$I(C_i, C_j) = \sum_{C_i}\sum_{C_j} p(C_i, C_j)\log\left(\frac{p(C_i, C_j)}{p(C_i)\cdot p(C_j)}\right) \qquad (2)$$

From which we can see that $I \geq 0$, and when $I = 0$, C_i and C_j becomes independent variables, and the higher I is, the stronger the dependency between C_i and C_j.

Thus, we calculate a similarity matrix $\boldsymbol{I}$ where each entry is the mutual information between a pair of parameters in $\mathfrak{C}^k_{\text{parent}}$, and then a distance matrix with normalization $(1 - \boldsymbol{I}/\max(\boldsymbol{I}))$. At this stage, we are able to obtain child clusters $\mathfrak{C}^{k,l}_{\text{child}}$ via agglomerative clustering. We tune number of child clusters so that the given ground truth of parameters known to be highly dependent on value distributions stay together in the smallest possible cluster.

For each cluster $\mathfrak{C}$, we fetch the list of unique value combinations $[\mathfrak{C}] = [[c^n_i, \cdots, c^p_j], \cdots, [c^m_i, \cdots, c^q_j]]$ and count the occurrence of each combination. Although each unique value combination is an existing configuration on some cells in the live cellular network, but that does not mean every combination is a good one. Hence, we use the occurrence count to infer the rationality that the combination with higher occurrence will be preferred.

4.6 Assertion, recommendation and trial

Now, we have the set of cells $\mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}})$ qualified for the recommendation $\mathcal{G}_{\text{s}} = \{c^{n\to m}_i, \cdots, c^{p\to q}_j\}$. Suppose the current configuration of $\mathcal{G}$ on cell d is denoted as $\mathcal{G}_{(d)}$. We then perform a two-step assertion for the safety of recommendation.

4.6.1 Candidate assertion: We first check the current configurations of each candidate cell $d \in \mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}})$ and remove the candidates whose configurations on $\mathcal{G}$ does not match the initial state in $\mathcal{G}_{\text{s}}$, *i.e.*, the set of candidates after the assertion is $\lfloor\mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}})\rceil = \{d : d \in \mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}}), \mathcal{G}_{(d)} = \{c^n_i, \cdots, c^p_j\}\}$.

4.6.2 Value dependency assertion: As mentioned in Sec.4.5, the values of some configuration parameters have dependencies among them, but they may not exhibit being configured at the same time, *i.e.*, in the same configuration change group. Hence, other than recommending $\mathcal{G}_{\text{s}}$, we may also need to recommend changes on parameters correlated with them.

Since spatial clusters have no overlapping member with each other, $\mathcal{G}$ can be divided into several no-overlapping subsets that the elements in each subset are in the same spatial cluster, *i.e.*, $\{C_i, \cdots, C_j : C_i, \cdots, C_j \in \mathfrak{C}\} \subseteq \mathcal{G}$. For a cell $d \in \lfloor\mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}})\rceil$, for such a subset $\{C_i, \cdots, C_j\} \subseteq \mathcal{G}$, we check the current value of other parameters in the same spatial cluster $\{c_{k(d)} \cdots, c_{l(d)}\}, C_k, \cdots, C_l \in \mathfrak{C}$. Thus, we write the vector $\mathfrak{C}_{(d)} = [c^m_i, \cdots, c^q_j, c_{k(d)}, \cdots, c_{l(d)}]$ as $\{c^{n\to m}_i, \cdots, c^{p\to q}_j\}$ is recommended to cell d. Then, we fetch the unique value combinations of $\mathfrak{C}$ containing $c^m_i, \cdots c^q_i$, denoted as $[\mathfrak{C}]_{c^m_i,\cdots,c^q_j} = [[c^m_i, \cdots, c^q_i, c^r_k, \cdots, c^t_l], \cdots]$. We pick the row in $[\mathfrak{C}]_{c^m_i,\cdots,c^q_j}$ with the minimum number of parameters different from $\mathfrak{C}_{(d)}$. If there is a tie, we pick the row with higher occurrence count. Finally, we add these dependent parameters that are different from the current settings on cell d to the recommendations, denoted as $\mathcal{R}^{(d)}_{\mathcal{G}_{\text{s}}}$.

To sum up, for a cell $d \in \lfloor\mathcal{S}(\mathcal{A}^{\text{rule}}_{\text{s}})\rceil$, we recommend the group of changes $\mathcal{G}_{\text{s}}$ and the supplementary changes $\mathcal{R}^{(d)}_{\mathcal{G}_{\text{s}}}$ which is the minimum effort to satisfy the parameter value dependencies of $\mathcal{G}_{\text{s}}$. The recommendations are then passed to the engineers for review and approval. Finally, the approved recommendations will be trialed on candidate cells.

5 Evaluation

Our goal for evaluating Chroma is to highlight the effectiveness of the individual components in improving the accuracy of change recommendation. We use the classic ML approach for evaluation by dividing the data into training set (70%) to first learn the context, followed by configuration change recommendations and comparisons with testing set (30%) to calculate the recommendation accuracy. We further seek help from the engineers to label our recommendations that they believe would result in a positive performance impact.

We use a very rich dataset collected from a tier-1 cellular service provider in the United States, covering on the order of hundreds of thousands of base stations deployed nationwide, and spanning time interval over a year from October 2021 to January 2023. For configuration data, we use 355.05 million of 5G, and 12.60 billion of LTE daily configuration snapshots. These snapshots are recorded for each cell at the basestation. We do a diff over days to discover configuration changes, yielding around 3.24 million of 5G and 32.27 million of LTE configuration changes. The KPI data is measured every 15-minutes at each cell and aggregated at daily

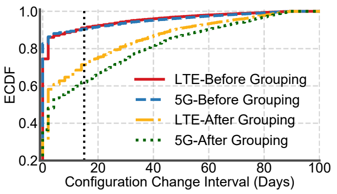


Figure 7: Result of parameter grouping

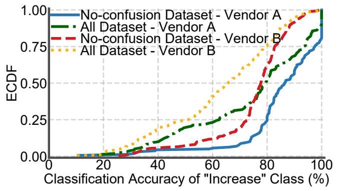


Figure 8: Classification accuracy

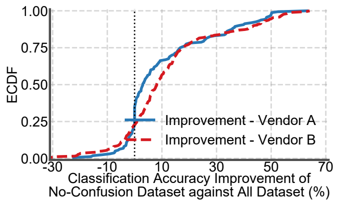


Figure 9: Classification improvement of no-confusion against all dataset

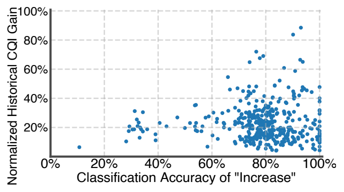


Figure 10: Classification Accuracy VS Historical CQI Percentage Improvement

Duration	Network	Vendor	# Parameters	# Config snapshots	# Changes Captured
May 2022-Jan 2023	5G NR	A	152	249.53M	2.60M
		B	38	105.52M	0.64M
Oct 2021-Jan 2023	4G LTE	A	425	5306.17M	10.63M
		B	113	7301.08M	21.64M

Table 4: Dataset from operational LTE/5G networks

granularity to enable a pre/post impact comparison around the configuration changes. We use network attributes using configuration snapshot as well as time-series data to derive the LMRD variables. The network attributes are recorded at a daily level. We have on the order of hundreds of network attributes. We summarize the details in Table 4.

5.1 Configuration change grouping result

As we discussed in Sec.3.2, configuration changes on the same cell are applied in close proximity. With the posterior-probability and Jaccard similarity based grouping, we successfully reduce the concurrence of configuration changes by 45% for LTE and 50% for 5G network. For a 15-day configuration change interval (the black dotted vertical line), we reduce the confusion by 20% for LTE and 28.5% for 5G dataset, which means we extract such amount of data from dirty raw dataset. This important reduction eliminates the confusion of performance impact arising from unrelated co-occurring configuration changes. Also, with grouping, we ensure that our recommendations can account for the dependency across configuration parameters and enable groups of configuration changes go hand-in-hand.

5.2 Impact classification result

We run the classification task on each group of configuration changes as described in Sec.4.4 using the no-confusion dataset (~40% of all dataset) and complete (or, all) dataset, respectively. We split the datasets with 7:3 for training and testing. As shown in Fig.8, using no-confusion dataset, over 93.75% of configuration changes for vendor A and 87.5% of that for vendor B reach classification accuracies over 60%, which is the threshold for recommending the such configuration changes to cell. In comparison, for the all dataset, only 75% of configurations for vendor A and 60% of that for vendor B reach classification accuracies over 60%. This shows that the no-confusion dataset helps identify more configuration change recommendations (may be better or worse which we will describe next) compared to the all-dataset. Fig. 9 shows the improvement in classification accuracy of using no-confusion dataset against using all dataset. We observe that more than 75% of configuration changes achieve higher classification accuracy with no-confusion dataset. This demonstrates better quality recommendations with no-confusion dataset compared to all dataset.

Next, we ask our network engineers to evaluate the quality of our change recommendations for both no-confusion and all datasets. We provide them with a list of our distinct configuration changes and ask them to color code based on which configuration changes have a higher likelihood of performance improvement - green for positive recommendation, red for negative implying that they would not like this change to implemented, and yellow for changes they are not sure about. Using a threshold of classification accuracy of 60%, we label the configuration change to be a true positive (TP - changes with accuracy ≥ 60% and green label), false positive (FP - changes with accuracy ≥ 60% and red label), true negative (TN - changes with accuracy < 60% and red label) and false negative (FN - change with accuracy < 60% and green label). The reason for the choice of 60% as the threshold was to balance the tradeoff between the high number of recommendations and the potential to capture ones that have higher likelihood of performance improvement. Table 5 shows our results for both vendors across all configuration changes. The accuracy and F1-score with no-confusion dataset is better than all dataset. This confirms that avoiding samples that have confusion of impacts is important for generating good quality configuration change recommendations. Precision is the ratio of TP to the sum of TP and FP. Recall is the ratio of TP to the sum of TP and FN. Precision and recall are also better with no-confusion dataset with the exception of precision on all dataset for vendor *A* (91.67%). Note that losing all the confusion samples might not be good and one could explore better algorithms at dealing with impact confusion in the future.

We now revisit the threshold for impact classification accuracy and the tradeoff compared to the performance improvement opportunity. Fig. 10 shows a scatterplot of classification accuracy for each configuration change versus the historical mean improvement in CQI. We obviously want to pick

Vendor	Data Type	#Config Change types	TP	FP	TN	FN	Not Sure	Precision	Recall	Accuracy	F1-score
A	No-Confusion	160	105	13	0	8	34	88.98%	**92.92%**	**83.33%**	**90.91%**
	All Data		99	9	4	14	34	**91.67%**	87.61%	81.75%	89.59%
B	No-Confusion	259	140	7	1	7	104	**95.24%**	**95.24%**	**90.97%**	**95.24%**
	All Data		74	8	0	73	104	90.24%	50.34%	47.74%	64.63%
Total	No-Confusion	419	245	20	1	15	138	**92.45%**	**94.23%**	**87.54%**	**93.33%**
	All Data		173	17	4	87	138	91.05%	66.54%	62.99%	76.89%

Table 5: Impact classification accuracy using labels from network engineers on the configuration changes. The F1-score and accuracy are better using no-confusion dataset compared to all dataset.

#Market	#Recommendation	#Applied Change	#Cell	#BS	Approval Rate	%BS with Improvements
17	972,436	210,611	37,189	4,236	21.66%	82.81%

Table 6: Trial statistics

configuration changes for recommendations that have high classification accuracy. But as can be seen, there are certain configuration changes with moderate accuracy (*e.g.*, 70-80%) that have a potential for more performance improvement. This can be viewed analogous to risk versus reward. For those changes with lower accuracy but still above our threshold of 60%, one can pick this recommendation for trial with the hope of a better CQI improvement. Thus, the recommendations with lower accuracy (higher risk) may sometimes be worth a trial as long as they have a potential of a better performance impact (higher reward).

6 Operational Experiences

This section presents our operational experiences trialing Chroma in a large scale live LTE and 5G cellular network. We observed improvements in service performance measured using CQI (composite quality index), confirming the effectiveness of Chroma's recommendation in practical settings.

6.1 Summary

Using one-year worth of configuration change, service performance, and network attribute data, we first apply Chroma to learn the contexts and the configuration changes that have the best likelihood of a service performance improvement. We observe tens of thousands of configuration changes across the network during this time interval. This provides us with a rich set of data to conduct our learning. Chroma outputs recommendations on the order of hundreds of distinct configuration changes. We then use the latest configuration snapshot across the network to identify each cell's configuration recommendations. Finally, we share the recommendations with the network engineers to seek their approvals on which changes can be implemented in the operational field.

Table 6 shows a summary of 17 markets (recall, one or several markets combined are equivalent to a state in the US) where we sought approvals and successfully implemented the changes. Adopting a new capability in production setup takes time and based on initial successes, engineers quickly ramp up to sift through the recommendations and start the change implementation. The number of base stations with change implementation in Table 6 is 4,236, which is a subset across all base stations and are selected on different criteria set by the engineers such as the ones that could potentially quickly benefit from configuration tuning, and their controlled region of optimization. As can also be seen, the applied changes are are 210K+ and are a subset of the recommendations because the engineers chose the ones in the first round that could be clear opportunities for improvements versus those that are kept on the back burner and visited later through carefully selected and controlled trials.

Engineer approvals. Our overall approval rate is 21.66%. It will grow over time as the engineers build confidence in Chroma. The remaining recommendations will be vetted by the engineers in subsequent iterations. We also observed a large difference in approval rates across the markets. After seeking feedback from network engineers, we categorize the approvals based on three reasons: (i) engineer's knowledge of the configuration parameters and their comfort level in experimenting the change in production, (ii) selecting base station locations which do not conflict with any other configuration change or software upgrade trials, and (iii) selecting locations that have a poor service experience and thus presents with a good potential to experiment the change and improve the performance.

Performance improvements. As can be seen from Table 6, we notice performance improvements across 82.81% of the base stations captured using the CQI metric. This is a significant result that demonstrates the effectiveness of Chroma in recommending and implementing performance improving configuration changes. These configuration changes were previously not known to the engineers from the corresponding markets, and thus were considered good recommendations by Chroma. Also, so far, we have not captured any unexpected degradation because of Chroma. However, we have a continuous monitoring in place that conducts the performance assessment and recommends a roll-back when there is a significant performance degradation.

6.2 Findings

We present a few interesting use case findings below.

6.2.1 Impact of preamble cyclic shift configuration changes on CQI Preamble cyclic shift (PRACHCS) setting is important to determine how many cyclic shifts are needed to generate

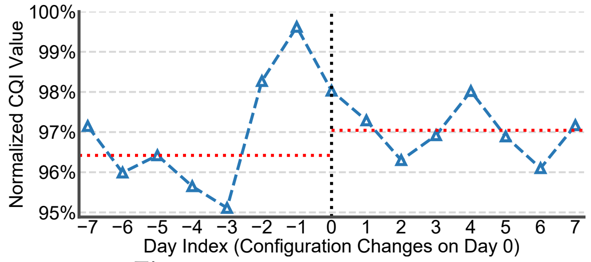


Figure 11: Trial of PRACHCS

the preamble. We recommended increasing the number of shifts, which improved coverage by extending the cell radius. This in turn, helped improve the overall CQI index. We show -7/+7 days pre/post comparison of CQI before and after our recommended configuration change in Fig. 11. A statistical pre/post comparison between the study group and control group yielded a performance improvement.

6.2.2 Impact of EN-DC capability based threshold changes for RSRP/RSRQ on CQI EN-DC (E-UTRAN New Radio - Dual Connectivity) is a non-standalone (NSA) feature enabling users to access both 5G and LTE networks simultaneously. The EN-DC capability based threshold for RSRP and RSRQ controls the filtering of target cells using A5 events based on RSRP and RSRQ values. We learned in Chroma that the configuration setting historically improved performance metrics measured in the LTE network. However, our recommendations in some locations did not improve CQI, which was contrary to our expectation. On further investigation, we discovered that the lack of 5G co-located cells at those locations did not offload the LTE traffic, and thus the improvements in LTE CQI were not immediately observed. However, in several locations with 5G co-located cells, we also observed that EN-DC performance were significantly improved indicating that the users were encouraged to camp on to 5G.

7 Limitations and future opportunities

Insights with confused data: Our current approach in Chroma for performance impact de-confusion (Sec.4.3.2) is to eliminate locations on days that have more than one configuration change group. This has the advantage of reducing false alarms, but has the risk of also losing good samples for true positives. Roll-back information can also shed light on changes that should *not* be recommended. In the future, one could explore algorithms to preserve locations by re-labeling the impacts for configuration changes that have no causal relationships with performance metrics. When there are no roll-backs, but multiple configuration changes (say A and B), one can associate this to other locations and identify if A or B in isolation have similar impacts.

System drift with time: Network evolution is important to tackle in operational environments, and one has to carefully conduct periodic re-training to ensure change recommendations keep up with software upgrades, and technology updates (*e.g.*, dynamic spectrum sharing in 5G). Furthermore, with the recent surge of video conferencing applications, it is crucial to incorporate new sets of performance metrics such as uplink throughput in the impact tradeoffs.

Context definitions and management overhead: A larger number of contexts can potentially result in increase in configuration management overhead and difficulty in troubleshooting unexpected impacts across contexts. As more trials are conducted, Chroma re-learns the contexts where the configuration change has the best chance of improving service performance. Our context learning is data-driven and we believe automated learning of contexts should provide good insights into explaining the contrasting performance impacts. We plan to keep our network attributes up-to-date which should help in better management of contexts.

Adaptation to dynamic configuration changes: The timescales of optimization can be more fine-grained such as every hour or 15-minutes. We anticipate new sets of challenges with fine-grained optimization such as performance impact assessment and de-confusion, availability of datasets in real-time to make faster decisions, and data management overheads such as missing data or delayed data. We believe Chroma can serve as core component to expand to dynamic configuration changes. Intuitively, to handle network events such as congestion or outages in real-time, one can leverage configuration changes from similar contexts in the past that have resulted in restoring performance behaviors.

8 Conclusion

In this paper, we presented the design and implementation of a new solution Chroma that automatically learns and uses contexts based on network attributes to recommend configuration changes that are highly likely to improve performance impact tradeoffs. We motivate the design of Chroma using data collected from a large operational LTE and 5G cellular network and present extensive evaluation results to highlight the classification accuracy of Chroma. Successful implementation of our trials demonstrates the efficacy of Chroma in identifying and recommending performance-improving network configurations.

Acknowledgement

We thank ACM MobiCom 2023 anonymous shepherd, reviewers, Rakesh Sinha, Maulik Shah, Vicki Littlefield, Jennifer Yates and Raghvendra Savoor for their fruitful feedback. We appreciate the collaboration and continuous support from network operation engineers from AT&T Mobility during the trialing of Chroma. Changhan Ge sincerely appreciates the support from Prof. Lili Qiu at UT Austin.

References

[1] 2009. 3rd Generation Partnership Project, TS 21.101 V8.0.0.

[2] 2010. 3rd Generation Partnership Project, TS 36.213 V9.2.0.

[3] 2011. 3rd Generation Partnership Project, TS 32.541 V10.0.0.

[4] 2011. Self-Optimizing Networks: The Benefits of SON in LTE.

[5] 2015. 3GPP LTE TS 32.500. Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements.

[6] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. 2020. AED: Incrementally Synthesizing Policy-Compliant and Manageable Configurations. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*. Association for Computing Machinery, New York, NY, USA, 482–495. https://doi.org/10.1145/3386367.3431304

[7] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 469–482. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/alipourfard

[8] U. Barth. 2009. Self-X RAN: Autonomous Self Organizing Radio Access Networks.. In *IEEE WiOpt*. http://dblp.uni-trier.de/db/conf/wiopt/wiopt2009.html#Barth09

[9] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 155–168. https://doi.org/10.1145/3098822.3098834

[10] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitendra Padhye, and David Walker. 2016. Don't Mind the Gap: Bridging Network-Wide Objectives and Device-Level Configurations. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 328–341. https://doi.org/10.1145/2934872.2934909

[11] Ryan Beckett, Ratul Mahajan, Todd Millstein, Jitendra Padhye, and David Walker. 2017. Network Configuration Synthesis with Abstract Topologies. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Barcelona, Spain) *(PLDI 2017)*. Association for Computing Machinery, New York, NY, USA, 437–451. https://doi.org/10.1145/3062341.3062367

[12] Rüdiger Birkner, Dana Drachsler-Cohen, Laurent Vanbever, and Martin Vechev. 2020. Config2Spec: Mining Network Specifications from Network Configurations. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA.

[13] Simon C. Borst, Arumugam Buvaneswari, Lawrence M. Drabeck, Michael J. Flanagan, John M. Graybeal, Georg K. Hampel, Mark Haner, William M. MacDonald, Paul A. Polakos, Gee Rittenhouse, Iraj Saniee, Alan Weiss, and Philip A. Whiting. 2005. Dynamic Optimization in Future Cellular Networks. *Bell Labs Technical Journal* 10, 2 (2005), 99–119. http://dblp.uni-trier.de/db/journals/bell/bell10.html#BorstBDFGHHMPRSWW05

[14] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* 16, 9 (sep 1973), 575–577. https://doi.org/10.1145/362342.362367

[15] Zhen Cao, Geoff Kuenning, and Erez Zadok. 2020. Carver: Finding Important Parameters for Storage System Tuning. In *Proceedings of the 18th USENIX Conference on File and Storage Technologies* (Santa Clara, CA, USA) *(FAST'20)*. USENIX Association, USA, 43–58.

[16] William W Cohen. 1995. Fast effective rule induction. In *Machine learning proceedings 1995*. Elsevier, 115–123.

[17] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin Vechev. 2018. Netcomplete: Practical Network-Wide Configuration Synthesis with Autocompletion. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation* (Renton, WA, USA) *(NSDI'18)*. USENIX Association, USA, 579–594.

[18] Seyed K. Fayaz, Tushar Sharma, Ari Fogel, Ratul Mahajan, Todd Millstein, Vyas Sekar, and George Varghese. 2016. Efficient Network Reachability Analysis Using a Succinct Control Plane Representation. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, USA.

[19] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation* (Oakland, CA) *(NSDI'15)*. USENIX Association, USA, 469–483.

[20] Aaron Gember-Jacobson, Aditya Akella, Ratul Mahajan, and Hongqiang Harry Liu. 2017. Automatically Repairing Network Control Planes Using an Abstract Representation. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 359–373. https://doi.org/10.1145/3132747.3132753

[21] Aaron Gember-Jacobson, Raajay Viswanathan, Aditya Akella, and Ratul Mahajan. 2016. Fast Control Plane Analysis Using an Abstract Representation. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 300–313. https://doi.org/10.1145/2934872.2934876

[22] S. Hamalainen, H. Sanneck, and C. Sartori. 2012. LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency. *Wiley, 1st edition* (2012).

[23] Haochen He, Zhouyang Jia, Shanshan Li, Yue Yu, Chenglong Zhou, Qing Liao, Ji Wang, and Xiangke Liao. 2022. Multi-Intention-Aware Configuration Selection for Performance Tuning. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 1431–1442. https://doi.org/10.1145/3510003.3510094

[24] Yigong Hu, Gongqi Huang, and Peng Huang. 2020. Automated Reasoning and Detection of Specious Configuration in Large Systems with Symbolic Execution. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 719–734. https://www.usenix.org/conference/osdi20/presentation/hu

[25] Siva Kesava Reddy Kakarla, Ryan Beckett, Behnaz Arzani, Todd Millstein, and George Varghese. 2020. GRoot: Proactive Verification of DNS Configurations. In *SIGCOMM 2020*. https://www.microsoft.com/en-us/research/publication/groot-proactive-verification-of-dns-configurations/ Best Paper Award.

[26] Siva Kesava Reddy Kakarla, Alan Tang, Ryan Beckett, Karthick Jayaraman, Todd Millstein, Yuval Tamir, and George Varghese. 2020. Finding Network Misconfigurations by Automatic Template Inference. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/nsdi20/presentation/kakarla

[27] Konstantinos Kanellis, Ramnatthan Alagappan, and Shivaram Venkataraman. 2020. Too Many Knobs to Tune? Towards Faster Database Tuning by Pre-Selecting Important Knobs. In *Proceedings of the 12th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'20)*. USENIX Association, USA, Article 16, 1 pages.

[28] Ajaykrishna Karthikeyan, Nagarajan Natarajan, Gagan Somashekar, Lei Zhao, Ranjita Bhagwan, Rodrigo Fonseca, Tatiana Racheva, and

Yogesh Bansal. 2023. SelfTune: Tuning Cluster Managers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/nsdi23/presentation/karthikeyan

[29] Peyman Kazemian, George Varghese, and Nick McKeown. 2012. Header Space Analysis: Static Checking for Networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation* (San Jose, CA) *(NSDI'12)*. USENIX Association, USA.

[30] Franck Le, Sihyung Lee, Tina Wong, Hyong S. Kim, and Darrell Newcomb. 2006. Minerals: Using Data Mining to Detect Router.

[31] Zhao Lucis Li, Chieh-Jan Mike Liang, Wenjia He, Lianjie Zhu, Wenjun Dai, Jin Jiang, and Guangzhong Sun. 2018. Metis: Robustly Tuning Tail Latencies of Cloud Systems. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 981–992. https://www.usenix.org/conference/atc18/presentation/li-zhao

[32] Hongqiang Harry Liu, Xin Wu, Wei Zhou, Weiguo Chen, Tao Wang, Hui Xu, Lei Zhou, Qing Ma, and Ming Zhang. 2018. Automatic Life Cycle Management of Network Configurations. In *Proceedings of the Afternoon Workshop on Self-Driving Networks* (Budapest, Hungary) *(SelfDN 2018)*. Association for Computing Machinery, New York, NY, USA, 29–35. https://doi.org/10.1145/3229584.3229585

[33] Ajay Mahimkar, Zihui Ge, Xuan Liu, Yusef Shaqalle, Yu Xiang, Jennifer Yates, Shomik Pathak, and Rick Reichel. 2022. Aurora: conformity-based configuration recommendation to improve LTE/5G service. In *Proceedings of the 22nd ACM Internet Measurement Conference, IMC 2022, Nice, France, October 25-27, 2022*. ACM, 83–97. https://doi.org/10.1145/3517745.3561455

[34] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. 2011. Rapid detection of maintenance induced changes in service performance. In *ACM CoNEXT*.

[35] Ajay Mahimkar, Zihui Ge, Jennifer Yates, Chris Hristov, Vincent Cordaro, Shane Smith, Jing Xu, and Mark Stockert. 2013. Robust Assessment of Changes in Cellular Networks. In *ACM CoNEXT*.

[36] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunasish Biswas. 2021. Auric: Using Data-Driven Recommendation to Automatically Generate Cellular Configuration. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 807–820. https://doi.org/10.1145/3452296.3472906

[37] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunasish Biswas. 2022. Aurora: Comformity-based Configuration Recommendation to Improve LTE/5G Service. In *Proceedings of the ACM Internet Measurement Conference 2022 Conference*. 807–820.

[38] Ajay Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. 2010. Detecting the Performance Impact of Upgrades in Large Operational Networks. In *ACM SIGCOMM*.

[39] Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel Talmadge King. 2011. Debugging the Data Plane with Anteater. *SIGCOMM Comput. Commun. Rev.* 41, 4 (Aug. 2011), 290–301.

[40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[41] W. Mohr. 2009. Self-Organisation in Wireless Networks - Use Cases and Their Interrelation.

[42] Usama Naseer and Theophilus Benson. 2017. Configtron: Tackling network diversity with heterogeneous configurations. In *ACM HotCloud*.

[43] Usama Naseer and Theophilus Benson. 2022. Configanator: A Data-driven Approach to Improving CDN Performance. In *USENIX NSDI*.

[44] Santhosh Prabhu, Kuan-Yen Chou, Ali Kheradmand, P. Brighten Godfrey, and Matthew Caesar. 2020. Plankton: Scalable network configuration verification through model checking. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA.

[45] Chaithan Prakash, Jeongkeun Lee, Yoshio Turner, Joon-Myung Kang, Aditya Akella, Sujata Banerjee, Charles Clark, Yadi Ma, Puneet Sharma, and Ying Zhang. 2015. PGA: Using Graphs to Express and Automatically Reconcile Network Policies. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 29–42. https://doi.org/10.1145/2785956.2787506

[46] Juan Ramiro and Khalid Hamied. 2011. Self-Organizing Networks (SON): Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE. (2011).

[47] Swati Roy, David Applegate, Zihui Ge, Ajay Mahimkar, Shomik Pathak, and Sarat Puthenpura. 2016. Quantifying the Service Performance Impact of Self-Organizing Network Actions. In *Proceedings of the 12th Conference on International Conference on Network and Service Management (CNSM 2016)*. International Federation for Information Processing, 37–45.

[48] Shambwaditya Saha, Santhosh Prabhu, and P. Madhusudan. 2015. NetGen: Synthesizing Data-Plane Configurations for Network Policies. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research* (Santa Clara, California) *(SOSR '15)*. Association for Computing Machinery, New York, NY, USA, Article 17, 6 pages. https://doi.org/10.1145/2774993.2775006

[49] Chong Shen, Dirk Pesch, and James Irvine. 2005. A Framework for Self-Management of Hybrid Wireless Networks Using Autonomic Computing Principles. *3rd Annual Communication Networks and Services Research Conference* (2005).

[50] Samuel Steffen, Timon Gehr, Petar Tsankov, Laurent Vanbever, and Martin Vechev. 2020. Probabilistic Verification of Network Configurations. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 750–764. https://doi.org/10.1145/3387514.3405900

[51] Xudong Sun, Runxiang Cheng, Jianyan Chen, Elaine Ang, Owolabi Legunsen, and Tianyin Xu. 2020. Testing Configuration Changes in Context to Prevent Production Failures. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 735–751. https://www.usenix.org/conference/osdi20/presentation/sun

[52] Yu-Wei Eric Sung, Xiaozheng Tie, Starsky H.Y. Wong, and Hongyi Zeng. 2016. Robotron: Top-down Network Management at Facebook Scale. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 426–439. https://doi.org/10.1145/2934872.2934874

[53] Chunqiang Tang, Thawan Kooburat, Pradeep Venkatachalam, Akshay Chander, Zhe Wen, Aravind Narayanan, Patrick Dowell, and Robert Karl. 2015. Holistic Configuration Management at Facebook. In *Proceedings of the 25th Symposium on Operating Systems Principles* (Monterey, California) *(SOSP '15)*. Association for Computing Machinery, New York, NY, USA, 328–343. https://doi.org/10.1145/2815400.2815401

[54] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, Da Yu, Chen Tian, Haitao Zheng, and Ben Y. Zhao. 2019. Safely and Automatically Updating In-Network ACL Configurations with Intent Language. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. Association for

Computing Machinery, New York, NY, USA, 214–226. https://doi.org/10.1145/3341302.3342088

[55] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-Scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1009–1024. https://doi.org/10.1145/3035918.3064029

[56] Shu Wang, Chi Li, Henry Hoffmann, Shan Lu, William Sentosa, and Achmad Imam Kistijantoro. 2018. Understanding and Auto-Adjusting Performance-Sensitive Configurations. *SIGPLAN Not.* 53, 2 (mar 2018), 154–168. https://doi.org/10.1145/3296957.3173206

[57] Xing Xu, Ioannis Broustis, Zihui Ge, Ramesh Govindan, Ajay Mahimkar, N. K. Shankaranarayanan, and Jia Wang. 2015. Magus: Minimizing Cellular Service Disruption during Network Upgrades. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies* (Heidelberg, Germany) *(CoNEXT '15)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2716281.2836106

[58] Peng Zhang, Aaron Gember-Jacobson, Yueshang Zuo, Yuhao Huang, Xu Liu, and Hao Li. 2022. Differential Network Analysis. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA. https://www.usenix.org/conference/nsdi22/presentation/zhang-peng

[59] Shenglin Zhang, Ying Liu, Dan Pei, Yu Chen, Xianping Qu, Shimin Tao, and Zhi Zang. 2015. Rapid and Robust Impact Assessment of Software Changes in Large Internet-based Services. In *ACM CoNEXT*.

[60] Chenxingyu Zhao, Tapan Chugh, Jaehong Min, Ming Liu, and Arvind Krishnamurthy. 2022. Dremel: Adaptive Configuration Tuning of RocksDB KV-Store. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 2, Article 37 (jun 2022), 30 pages. https://doi.org/10.1145/3530903

[61] Hang Zhu, Varun Gupta, Satyajeet Singh Ahuja, Yuandong Tian, Ying Zhang, and Xin Jin. 2021. Network Planning with Deep Reinforcement Learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 258–271. https://doi.org/10.1145/3452296.3472902

[62] Yuqing Zhu, Jianxun Liu, Mengying Guo, Yungang Bao, Wenlong Ma, Zhuoyue Liu, Kunpeng Song, and Yingchun Yang. 2017. BestConfig: Tapping the Performance Potential of Systems via Automatic Configuration Tuning *(SoCC '17)*. Association for Computing Machinery, New York, NY, USA, 338–350. https://doi.org/10.1145/3127479.3128605