# Optimal configuration determination in Cognitive Autonomous Networks

Anubhab Banerjee[1,2], Stephen S. Mwanje[1], and Georg Carle [2]
[1]Nokia Bell Labs, Munich, Germany
[2]Dept. of Informatics, Technical University of Munich, Germany
Email:*anubhab.banerjee@tum.de, stephen.mwanje@nokia-bell-labs.com, carle@net.in.tum.de*

***Abstract*—Cognitive Autonomous Networks (CAN) promises to raise the level of operational autonomy in mobile networks through the introduction of Artificial Intelligence (AI) and Machine Learning (ML) in the network processes. In CAN, learning based functions, called Cognitive Functions (CF), adjust network control parameters to optimize their objectives which are different Key Performance Indicator (KPI). As the CFs work in parallel, there is often an overlap among their activities regarding control parameter adjustment, i.e., at one point of time, multiple CFs may want to change a single control parameter albeit by different degrees or to different values depending on their respective levels of interest in that parameter. To resolve this dispute, a coordination mechanism is required for sharing the parameter among the independent CFs according to their individual interest levels. In this paper we provide the design of such a Controller in CAN to determine the optimal control parameter value. The Controller first quantifies the impact of that parameter on the objective of each CF, based on which the Controller determines the optimal value using Eisenberg-Gale solution. A numerical evaluation shows that compared to state-of-the-art, the proposed Controller can improve performance by up to 7.7%, while implementation in a simulation environment shows that the proposed Controller is feasible for use in a real life scenario.**

***Index Terms*—Network Automation, Game Theory, Deep Learning, Fisher Market Model**

## I. Introduction

Cognition and autonomy are strongly desired capabilities in future communication networks to serve more connections with increasing demands for throughput, latency and reliability. Using Artificial Intelligence (AI) and Machine Learning (ML) algorithms, new automation functions, called Cognitive Functions (CF), automate specific tasks in the evolved network known as Cognitive Autonomous Network (CAN) [1], thereby raising the levels of operational efficiency and reducing cost. In CAN, each CF is an independent learning agent - it continuously observes and learns from the network states, and acts based on it. As these CFs work in parallel often sharing the same resources, coordination among them is necessary regarding the usage of shared resources. They can coordinate by communicating - either directly with one another, or via a central entity (called a controller). In [2] we showed that in CAN, a centralized coordination mechanism using a controller is more beneficial. Nevertheless, designing such a Controller is not a trivial task, since the behavior of any of the CFs is unknown to the other CFs and the Controller itself. The Controller must act in a dynamic manner based on current available information from the CFs.

In [3] we proposed the design of such a controller whose main functionality is to resolve all types of conflicts that may arise among the CFs. We modeled CAN as a Multi Agent System (MAS), as MAS is a generic and very widely used framework for multi-criteria optimization, and extensively looked for existing multi-criteria optimization studies in MAS. After a very thorough search we did not find any existing research work on a MAS similar to the one we propose. As we elaborated this in [3], we do not reproduce the same background research in this paper and consider [3] as the only existing state-of-the-art.

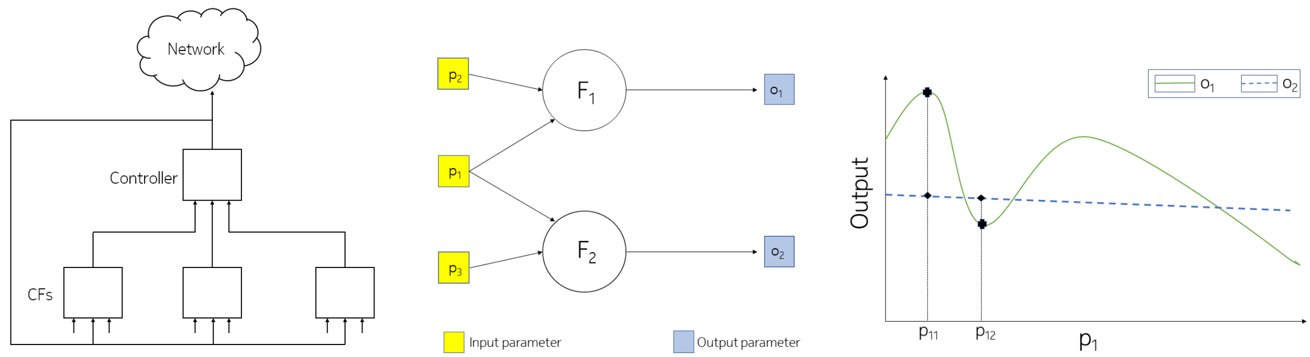
The Controller proposed in [3], [4] selects that particular value of the parameter of conflict for which the product of the utilities of all the competing CFs is maximum. According to Nash's Social Welfare Function (NSWF) [5], this solution is optimal for the combined interest of the CFs. However, it has two major shortcomings -

1) The Controller needs a priori knowledge on the configuration sharing among multiple CFs, i.e., which configuration is used by which set of CFs. So, whenever a new CF is introduced in the system, there has to be an exchange of information between the newly introduced CF and the Controller, which is not discussed in [3].
2) the controller does not consider varying interests among the CFs for a specific configuration. Changing a configuration by a certain amount may change the objective of one CF significantly and objective of another CF merely. Such CFs should not be treated equally even if both have interest in the configuration.

Our main contribution in this paper is to design a new Controller which accounts for the importance of individual CFs while determining optimal configurations in CAN. Our contributions in this paper are two-fold:

- we design a Controller using which both the aforementioned disadvantages of [3] are overcome. While determining the optimal value of a shared configura-

978-3-903176-32-4 © 2021 IFIP



(a) CAN abstraction (b) example CAN model (c) Output variations of $\mathrm{F}_1$ and $\mathrm{F}_2$ w.r.t. $p_1$

Fig. 1: CAN abstraction and example CAN model

tion, the Controller takes the influence of it on the output of each CF into account, so that the calculated configuration gives more priority to those CFs whose outputs heavily depend on that configuration.
- We quantify the importance of a configuration on the output as *config-weight*. When a configuration is shared among multiple CFs, these config-weight values help to visualize the importance of that configuration to each CF so that based on the values, actions of CFs can be prioritized.

Our proposed method gives a significant improvement over existing solution [3] and using a real life scenario, we also discuss on the implementation of the proposed solution.

## II. Problem Description

A typical network is characterized by many control parameters (CP) and Key Performance Indicators (KPI). For example, a Base Station (gNB), has several CPs like Transmission Power (TXP), Antenna Tilt (RET), and, multiple observable KPIs like downlink throughput, Radio Link Failures (RLF), latency. In CAN, each CF focuses on optimizing of one or more KPIs as its objective. The CF continuously observes and learns the variation in its performance w.r.t. changes in the CPs or the external environments, and based on that, it determines the values of CPs for which its objective is optimal in a certain state. In 5G, the environment can change very rapidly [6], and the CFs may detect frequent changes in the optimal CP values. If each CF changes the CPs according to its own will, stability of the whole system may be compromised. This is why we assume, for sake of stability issue, only the Controller can change the CPs. Without going into details, we abstract a CAN as shown in Fig. 1a in a hierarchical overview where Controller stays one level above the CFs. Before changing any network configuration, the Controller takes feedback from all the CFs and makes a decision which is beneficial for the collective interest of all the CFs.

Most significant challenge for the Controller is to determine the optimal value of a shared CP (configuration). The challenge is described as follows: consider a CAN with two CFs - $\mathrm{F}_1$ and $\mathrm{F}_2$, and one Controller (Fig. 1b). Both the CFs share the input CP $p_1$, so both of them should have a say on determining the final value of $p_1$, but how much say each of them should have is a matter of utmost importance. Let us assume that we plot the variations of objectives of the two CFs with respect to $p_1$ in Fig. 1c. It is evident from Fig. 1c that when $p_1$ changes, $o_1$ varies more than $o_2$. For example, when $p_1$ changes from $p_{11}$ to $p_{12}$, $o_1$ decreases from its maximum to its minimum, yet $o_2$ remains almost constant. So, for $\mathrm{F}_2$ it does not matter if $p_1$ is set at $p_{11}$ or $p_{12}$, but for $\mathrm{F}_1$ it is clearly a matter of utmost importance. From this observation, it is evident that $\mathrm{F}_1$ should be given more importance while determining the value of $p_1$ than $\mathrm{F}_2$, or, to express it mathematically, while determining $p_1$, if the Controller gives importance $w_{F_1}^{p_1}$ to $\mathrm{F}_1$ and importance $w_{F_2}^{p_1}$ to $\mathrm{F}_2$, then $w_{F_1}^{p_1}$ should be greater than $w_{F_2}^{p_1}$, i.e., $w_{F_1}^{p_1} > w_{F_2}^{p_1}$. At the same time we assume in our CAN model, for the Controller, $o_1$ and $o_2$ have equal importance ($w_{o_1} = w_{o_2}$), i.e., by changing a configuration value if the output $\mathrm{F}_1$ decreases by x% and output of $\mathrm{F}_2$ increases by y% with y > x, then the new configuration is selected by the Controller.

In this paper, our target is to propose the design of a Controller, which takes these $w_{F_i}^{p_i}$ values into account while calculating optimal configurations in CAN, with the following properties -

- The Controller can be used to determine the optimal values for all configurations in CAN.
- The Controller has to calculate configuration of the system in a way that is optimal for the combined interest of all the CFs.
- While calculating the optimal value for the system, the Controller should take individual $w_{F_i}$ values into account, and, at the same time should give equal importance to the output of each CF.
- The calculation mechanism of the Controller has to be very generic and dynamic to be used in real life.

## III. Proposed Solution

In this Section we provide end-to-end workflow of the whole system along with the design of the Controller which satisfies all four properties listed in Section II.

### *A. Requirements for Optimal Configuration Calculation*

During optimal configuration calculation, the Controller requires three pieces of information -

*a) optimal-config-range set:* Being a learning agent, a CF can generate its most favorable configuration set based on its learning, called **optimal-config-range set** (OCRS). An OCRS has the following structure: $[min_{config}, max_{config}]$ where $min_{config}$ and $max_{config}$ denote the lower and upper bound of the set respectively. For any configuration value between $min_{config}$ and $max_{config}$, the objective of the CF always lies within a certain percentage (value of which to be decided by the Mobile Network Operator (MNO)) of its maximum objective value, called *cf-return-size*.

*b) utility function:* In CAN, for the Controller to understand and compare between objectives of different CFs which have different dimensions, it is recommended to convert them in some identical predefined scale. This scale can be provided by the MNO or Controller beforehand. Example of such a scale is [0:10], where 0 means the lowest and 10 means the highest achievable value by the CF.

A CF can generate the utility function based on its learning. To convert its output to this scale, each CF generates a function which is called **utility function** (UF). Let us assume that while using a configuration $p$, maximum and minimum achieved performance of the CF are $c_{max}$ and $c_{min}$ respectively. A utility function, denoted by $f(p)$, maps the objective obtained from $p$ (which is $c$) in the predefined ([0:10]) scale -

$$f(p) = \frac{c - c_{min}}{c_{max} - c_{min}} = \frac{c - 0}{10 - 0} = \frac{c}{10} \quad (1)$$

Similarly, if for a value $p_1$ objective of the CF is $c_1$, then the utility value corresponding to $p_1$ is $\frac{c_1}{10}$.

*c) Config-weight:* In general, different input configurations have different levels of importance (weight) in determining a CF's objective. The relative importance of a configuration on the output is called **config-weight** (CW). For example, in Fig. 1b we see that $F_1$ has two input configurations $p_1$ and $p_2$. If we assume that $w_1$ and $w_2$ are their config-weight values respectively, we have to calculate $w_1$ and $w_2$ in such a way so that $w_1 + w_2 = 1$, so that these $w_i$ values generated by different CFs for the same configuration are expressed on the same scale and become comparable. This value can be calculated either by the CF itself (based on its learning) or set by the MNO (based on previous knowledge). Also, MNO can always override the value calculated by the CFs.

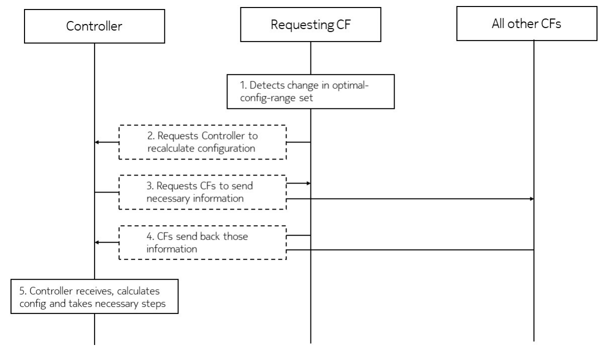



Fig. 2: End-to-end workflow of CAN

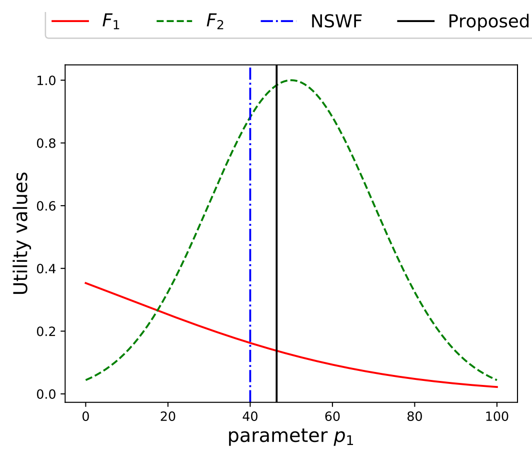



Fig. 3: Comparison between NSWF and proposed solution

### *B. Workflow of Optimal Configuration Calculation*

Here we give an overview on how a CAN works. All the CFs are trained with real life data (or some simulator generated data in case real life data is unavailable) before they become operational. This is a standard practice used in many Self Organizing Networks (SON) cases. Main purpose of the training is to determine the config-weight values which remain fixed unless MNO changes them.

CAN starts with initial configurations set by the MNO manually from previous experience. The end-to-end workflow of an optimal configuration calculation consists of five steps (Fig. 2) -

*a) Step 1:* After the system becomes operational, each CF begins its learning process. After every certain time interval, the CF calculates the OCRS and compares it with the one calculated in the last cycle. If these two are identical, the CF continues its learning until the next cycle, otherwise, it proceeds to Step 2.

*b) Step 2:* After a CF finds a change in its OCRS, it requests the Controller to recalculate that configuration. This CF is denoted as Requesting CF (Fig 2). There can be one or multiple Requesting CFs at a cycle for same or different configurations. However, the Controller calculates optimal value of a configuration only once in a cycle.

*c) Step 3:* After the Controller receives a request for configuration recalculation, it sends a message to all CFs in the system, asking to send their OCRS, UF and CW.

*d) Step 4:* After all the CFs receive the request from the Controller, each CF sends its latest (i) OCRS, (ii) UF and (iii) CW values to the Controller.

*e) Step 5:* In the final step, the Controller calculates the optimal configuration (discussed in Section III-C) based on information received from CFs and makes necessary changes in the network.

### C. Optimal Configuration Calculation

*1) Fisher Market Model (FMM):* In Game Theory, Fisher Market Model (FMM) [7] provides optimal solution for the combined interest of all while taking individual interests into account. To calculate the optimal configuration for the combined interest of all interested CFs while taking their individual interests into account, in this Section we model CAN as an FMM. An FMM $\mathcal{M}$ consists of a set of buyers $\mathcal{D} = \{d_1, d_2, .., d_d\}$ and a set of items $\mathcal{C} = \{c_1, c_2, .., c_c\}$, where every buyer $d_i$ has:

- An initial budget ($b_i$) which can be visualized as money that can only be utilized to purchase the items and has no intrinsic value to the buyer.
- A utility function $u_i$:$[0,1]^c \to \mathbb{R}$, that maps a quantity vectors of the c items to some real predefined scale.

$u_i(x_i)$ represents the buyer's utility when receiving $x_i$ amount of items. The set of budgets is denoted by $\mathcal{B} = \{b_1, b_2, .., b_d\}$. Without any loss of generality, the supply of each good is assumed to be one unit and the total budget of all buyers is normalized to one, i.e., $\sum_{i=1}^{d} b_i = 1$ [8]. In an FMM game, each agent first reports its preference to some central entity and the central entity then determines a market equilibrium according to the budgets of the agents and their reported preferences on the items of C, i.e., based on the budget $b_i$ of a buyer $d_i$, the target is to determine the set of items ($c_i$) the buyer should possess for an optimal allocation of items [8].

*2) CAN as a FMM:* We introduce an FMM game in CAN to calculate optimal value of a configuration. The Controller can be visualized as the central entity and CFs as buyers. Considering example of Fig. 1b, while determining optimal $p_1$, $\mathcal{D} = \{\mathrm{F}_1, \mathrm{F}_2\}$ and $\mathcal{C} = \{p_1\}$. Now, we visualize the $w_{F_i}^{p_1}$ values as their respective budgets, e.g., $w_{F_1}^{p_1} = w_1$, $w_{F_2}^{p_1} = w_2$ and respective UFs are $f_1(p_1)$ and $f_2(p_1)$. Now, the game model is complete as: $\mathcal{D} = \{\mathrm{F}_1, \mathrm{F}_2\}$, $\mathcal{C} = \{p_1\}$ and $\mathcal{B} = \{w_1, w_2\}$ and the target is to find optimal $p_1$.

*3) Solution to FMM:* It is important to find the equilibrium solution in an FMM game to obtain the optimal value of $p_1$.It can be captured by the Eisenberg-Gale convex program [9] if the utility functions of the buyers belong to the same class in Constant Elasticity of Substitution (CES) family. Utility functions in the CES family take the form of

$$u_i(x_i) \;=\; (\sum_{j=1}^{c} a_{ij} \cdot x_{ij}^{\rho})^{\frac{1}{\rho}} \tag{2}$$

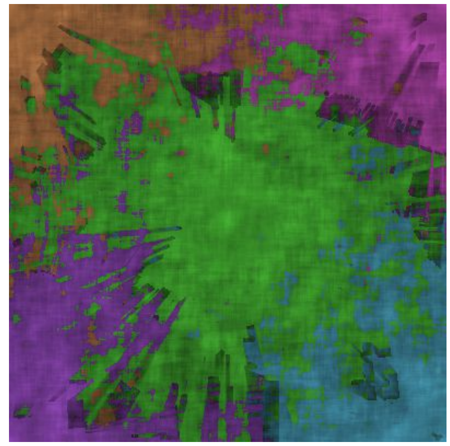


Fig. 4: Radio coverage of the cells

where $\rho$ parameterizes the family, $-\infty < \rho \le 1$, $\rho \ne 0$. The Leontief, Cobb-Douglas, and linear utility functions are obtained when $\rho$ approaches $-\infty$, 0, and equals 1 respectively [8]:

$$Leontief : u_i(x_i) = \min_{j \in [c]} \left\{ \frac{x_{ij}}{a_{ij}} \right\} \tag{3}$$

$$Cobb - Douglas : u_i(x_i) = \prod_{j \in [c]} x_{ij}^{a_{ij}} \tag{4}$$

$$Linear : u_i(x_i) = \sum_{j \in [c]} a_{ij} \cdot x_{ij} \tag{5}$$

For these three types of utility functions, the Eisenberg-Gale solution takes the following form:

$$\max \prod_{i=1}^{d} u_i^{b_i} \tag{6}$$

$$\text{s.t. } u_i = (\sum_{j=1}^{c} a_{ij} \cdot x_{ij}^{\rho})^{\frac{1}{\rho}}, \forall i \in [d]$$

$$\sum_{i=1}^{d} x_{ij} \le 1, \forall j \in [c] \;\&\; x_{ij} \ge 0, \forall i \in [d], j \in [c]$$

For some values of $\rho$, e.g., $\rho = 1$, the objective function of this convex program is not strictly concave, which means that there can be multiple market equilibria [8].

*4) Eisenberg-Gale solution for CAN:* When we use Eq. 6 for optimal $p_1$ calculation,we see that $u_i(x_i)$ becomes $f_i(p_1)$ and $b_i$ becomes similar to $w_i$ but not equivalent as $b_i$ values are normalized ($\sum_{i=1}^{d} b_i$) but $w_i$ values are not. So, $w_1$ and $w_2$ are normalized to $w_1^{'}$ and $w_2^{'}$ such that $w_1^{'} + w_2^{'} = 1$ and $w_i^{'}$ become equivalent to $b_i$. From Eq. 1 we see that CF utility functions are linear w.r.t. CPs and they are members of Linear CES family (Eq. 5, $\rho = 1$).

*5) Optimal Configuration Calculation by Controller:* Here we describe how the optimal configurations are calculated by the Controller. Considering CAN model of Fig. 1b, let us assume that when the Controller calculates $p_1$, the values of OCRS, UF and CW, for $\mathrm{F}_1$ are - $\{[p_{1min}^{F_1}, p_{1max}^{F_1}], w_{p_1}^{F_1}, f_1(p_1)\}$ and for $\mathrm{F}_2$ are - $\{[p_{1min}^{F_2}, p_{1max}^{F_2}], w_{p_1}^{F_2}, f_2(p_1)\}$.

The Controller combines two OCRSs into a final-optimal-config set (FOCS) taking minimum of ($p_{1min}^{F_1}$, $p_{1min}^{F_2}$) and maximum of ($p_{1max}^{F_1}$, $p_{1max}^{F_2}$) and selects the $p_1^*$ from FOCS, for which $f_1(p_1^*)^{w_1^{'}} \cdot f_2(p_1^*)^{w_2^{'}}$ is maximum.

In case, there are multiple values of $p_1$ for which $f_1(p_1)^{w_1^{'}} \cdot f_2(p_1)^{w_2^{'}}$ is maximum, the one for which the weighted utility values are closest to one another is selected. The closeness of the values can be measured using standard deviation (SD), the lower the SD value, the closer the values are to one another. For example, let us assume $p_{11}$ and $p_{12}$ are two values of $p_1$, which belong to the FOCS and $f_1(p_{11})$ = 10, $f_2(p_{11})$ = 6, $f_1(p_{12})$ = 6, $f_2(p_{12})$ = 5.83, $w_1^{'}$ = 0.3, $w_2^{'}$ = 0.7. For both $p_{11}$ and $p_{12}$, value of $f_1^{w_1^{'}} \cdot f_2^{w_2^{'}}$ is 6.992. As the final value is equal for both $p_{11}$ and $p_{12}$, we measure the SD of [$f_1^{w_1^{'}}$, $f_2^{w_2^{'}}$] for both cases. For $p_{11}$, the SD value is 0.755 and for $p_{12}$, the SD value is 1.186 and so, $p_{11}$ is selected as the optimal value of $p_1$.

## IV. Implementation

Here we show the advantages of using our proposed solution over [3] using an example. Let us consider the CAN model shown in Fig. 1b and assume that the CFs are capable of generating the UFs and CWs based on their learning. As both $F_1$ and $F_2$ share $p_1$, we discuss how the optimal value of $p_1$ is calculated taking individual interests of $F_1$ and $F_2$ into account. This is to be noted that, values of all the parameters in this Section have been assumed in accordance with the values assumed in [3].

Let us assume that the UFs generated by $F_1$ and $F_2$ are modeled as Gaussian distributions:

$$f_1(p_1, p_2) = 0.5e^{\frac{-(p_1+50)^2}{2p_2^2}} \tag{7}$$

$$f_2(p_1, p_3) = e^{\frac{-(p_1-50)^2}{2p_3^2}} \tag{8}$$

As we discuss calculation of optimal $p_1$, we keep $p_2$ and $p_3$ constant. Both $F_1$ and $F_2$ have been trained on $p_1$: [0, 100] and their UFs are expressed on a [0:1] scale. The reason behind assuming Gaussian functions as UFs is - in real life, distributions of KPIs resemble Gaussian Distribution to a great extent [10]. We assume cf-return-size as 50% for $p_1$, so that from Eq. 7, $F_1$ calculates OCRS as [0, 36] and from Eq. 8, $F_2$ calculates OCRS $p_1$ as [27, 73]. Based on learning, $F_1$ generate CW values as $\zeta_{p_1} = 0.339$, $\zeta_{p_2} = 0.661$. As $\zeta_{p_1}$ + $\zeta_{p_2} = 1$, no further normalization is needed. For $F_2$, input parameters are $p_1$ and $p_3$. Based on learning, $F_2$ generate CW values as $\zeta_{p_1} = 1$, $\zeta_{p_3} = 0$. As $\zeta_{p_1} + \zeta_{p_3} = 1$, there is no need for further normalization.

### A. Optimal $p_1$ Calculation

After the CFs send OCRS, UF, CW to the Controller, the Controller calculates the optimal configuration as described in Section III-C5. Value of $w_1$ (config-weight of $p_1$ sent by $F_1$) is 0.339 and value of $w_2$ (config-weight of $p_1$ sent by $F_2$) is 1. After normalization, $w_1$ becomes $w_1^{'}$ = 0.25 and $w_2$ becomes $w_2^{'}$ = 0.75. When the Controller combines the OCRSs provided by $F_1$ and $F_2$, the FOCS becomes [min(0, 27), max(36,73)] or, [0, 73]. As the CFs have been trained when $p_1$ is varied in steps of 1, the Controller also samples values in [0, 73] in steps of 1 and selects the sample value for which $f_1^{w_1^{'}} \cdot f_2^{w_2^{'}}$ is maximum.

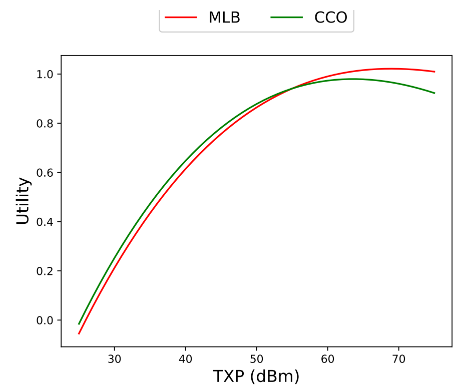



Fig. 5: Utilities of CFs vs TXP

### B. Comparison with NSWF solution

In Fig. 3 we plot the variations of utility values with respect to $p_1$. The red plot shows the utility of $F_1$ and the green plot shows the utility of $F_2$. According to [3], optimal $p_1$ for the system is 40 and according to our proposed solution, optimal $p_1$ is 46. However, we see that, when we use our proposed solution, utility of $F_1$ decreases by 2.4% but utility of $F_2$ increases by 10.1%, so the overall system performance improves by 7.7% which is quite significant.

## V. Real life Scenario

### A. Simulator Description

Here we discuss how the optimal configurations are calculated in a real life scenario based on a system level simulator that uses and extends the emulator in [11] and has already been used extensively in previous research works like [12], [13]. The environment of the simulations is an authentic recreation of a small part of the city of Hamburg. In an area of 4 sq. kilometer we deploy 5 cells as shown in Fig. 4. All the cells are 1-sector macro cells (2GHz) using realistic radio propagation models (the WINNER+ model [14]). We place 100 users across all cells randomly for 20 times and collects data for all 20 cases. As user mobility greatly affects in depicting the variation of the KPIs w.r.t. control parameters, we make the users static so that the relationship between each KPI and control parameter can be clearly observed. Although we only use radio related measurements in this evaluation, the simulator implements full flow-level simulation, such as FTP or video traffic, and could also provide flow related data if needed.

### B. CF Implementation

In all our simulations, we focus on the central cell shown in Green color in Fig. 4. We assume a CAN deployed at the this cell with two CFs - Mobility Load Balancing (MLB) and Capacity and Coverage Optimization (CCO). MLB tries to reduce the load in the cell and CCO tries to maximize the coverage and capacity of the cell. For MLB, the CPs are - Time To Trigger (TTT), Cell Individual Offset (CIO), Downlink Transmission Power (TXP) and Antenna Tilt or Remote Electrical Tilt (RET) and for CCO the CPs are - TXP and RET. Objectives and the input parameters of these CFs [15] are listed in Table I. Although we simulate with 2 CFs, proposed concept can be extended to any finite number of CFs in a CAN. We use two neural network (NN) to implement these two CFs. Each CF has 5 fully connected layers, and each layer has 50 nodes. We use MSE as the loss function and Adaptive learning rate optimizer or Adam optimizer [16]. Each CF has the inputs and outputs as listed in Table I. We collect data from 20 different scenarios and use them for training the NNs,so that after the training is done, each CF can predict the output corresponding to a set of input configurations. As TXP is shared between both the CFs, in this paper we describe how optimal values for TXP can be calculated using the proposed Controller.

TABLE I: Inputs and Outputs of CCO and MLB

| Name | Input | Target |
|---|---|---|
| MLB | TTT, CIO, TXP, RET | Minimize Load |
| CCO | TXP, RET | Maximize downlink throughput |

We generate the training data from the simulator and range of TXP, used for training, is 25 dBm to 75 dBm. CW values for TXP, which MLB and CCO generate based on their learning, are 0.35 and 0.08 respectively. We use a linear function to convert output of each CF into a utility value using Eq. 1. For generating the OCRS for a particular parameter, each CF varies TXP in steps of 1 and selects those values for which its utility value is within 50% of its maximum utility value (cf-return-size).

### C. Utility Interpretation

When a CF generates UF for TXP all the other parameters (TTT, CIO, RET) are kept constant. In case of CCO, utility value is directly proportional to the CF output, i.e., the higher downlink average user throughput, the higher the utility value whereas, in case of MLB, utility value is inversely proportional to the output, i.e., the higher the load, the less favorable is that configuration for the system and the lower is the utility value.

In Fig. 5 we plot the utility functions of MLB and CCO when TXP is varied. We see that when the TXP is increased, utility value of MLB gradually increases, and remains constant after a certain value of TXP, and, utility value of CCO gradually increases, reaches the maximum and then starts decreasing. This happens because when TXP is kept increasing, interference from neighboring cells also keep increasing and downlink throughput value starts decreasing because of that interference.

### D. Optimal Configuration Calculation

In this section we describe how optimal configuration for TXP is calculated. The same approach has to be followed for calculating the optimal configurations for RET, CIO and TTT. As already stated earlier, we assume -

- The system starts with some pre-loaded configuration: TXP = 46 dBm, RET = 0 deg, CIO = 0 dB.
- The CFs also have loaded OCRS, UF and CW values. CW and UFs are the same as defined in last section. However, for OCRS, MLB has [43 - 50] and CCO has [43 - 50]. These values have been chosen deliberately so that the CFs trigger the optimal configuration calculation process

After the system starts, in an interval of 120 seconds, both the CFs check for their OCRS. After recalculation, OCRS for MLB is [37, 75] and for CCO is [36, 75]. Both of them send request to the Controller to recalculate TXP value. After exchanging necessary information, the Controller generates the FOCS as [min(37,36), max(75,75)] = [36, 75]. After this the Controller calculates the optimal value of TXP as described in this paper and the optimal value of TXP is 66 dBm. We also calculate the optimal TXP by the method proposed in [3] (in which case the optimal TXP is 64 dBm), and found that our proposed method gives a 2% overall improvement.

### E. Time Complexity

In this Section we discuss about the time complexity of our proposed design. Our proposed solution is beneficial to use in real life if another configuration recalculation request does not arrive before previous configuration recalculation is complete. We run the simulations to determine the time consumed while calculating each of the control parameters (TXP, TTT, CIO, RET) and the result is shown in Table II. We see that time consumed for calculating one optimal configuration varies between 0.2 - 0.35 ms, which is negligible compared to the frequency of request arrival (120 s in our simulation set-up).

TABLE II: Time consumed in each parameter calculation

| Param | Time (ms) | Param | Time (ms) | Param | Time (ms) | Param | Time (ms) |
|---|---|---|---|---|---|---|---|
| TXP | 0.35 | TTT | 0.22 | CIO | 0.22 | RET | 0.32 |

## VI. Conclusion and Future Direction

In this paper we designed a Controller for dynamically calculating optimal configurations in a CAN. Our proposed Controller overcomes the existing problems and provides up to 7.7% improvement in overall system performance over state-of-the-art. Along with that we also proposed a new metric called config-weight to quantify the importance of a control parameter on a KPI in a cellular network. We also implemented our solution in a simulation environment and showed the benefits of using our proposed solution. As a next step we plan to implement our solution in a mobile network environment and make necessary adjustments.

## References


[1] S. S. Mwanje and C. Mannweiler. *Towards Cognitive Autonomous Networks: Network Management Automation for 5G and Beyond*. John Wiley & Sons, 2020.

[2] A. Banerjee, S. S. Mwanje, and G. Carle. On the necessity and design of coordination mechanism for cognitive autonomous networks. arXiv:2001.07031, 2020.

[3] A. Banerjee, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In *2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE, 2020.

[4] A. Banerjee, S. S. Mwanje, and G. Carle. Ran cognitive controller. arXiv:2010.10278, 2020.

[5] S. Ramezani and U. Endriss. Nash social welfare in multiagent resource allocation. In *Agent-mediated electronic commerce. Designing trading strategies and mechanisms for electronic markets*, pages 117–131. Springer, 2009.

[6] A. Alnoman and A. Anpalagan. Towards the fulfillment of 5g network requirements: technologies and challenges. *Telecommunication Systems*, 65(1):101–116, 2017.

[7] W. C. Brainard, H. E. Scarf, et al. *How to compute equilibrium prices in 1891*. Citeseer, 2000.

[8] S. Brânzei, Y. Chen, X. Deng, A. Filos-Ratsikas, S. K. S. Frederiksen, and J. Zhang. The fisher market game: equilibrium and welfare. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[9] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[10] I. Marsh, B. Grönvall, and F. Hammer. The design and implementation of a quality-based handover trigger. In *International Conference on Research in Networking*, pages 580–591. Springer, 2006.

[11] Nokia Siemens Networks. White paper: Self-organizing network (son): Introducing the nokia siemens networks son suite-an efficient, future-proof platform for son. Technical report, October, 2009.

[12] S. S. Mwanje, M. Kajó, S. Majumdar, and G. Carle. Environment modeling and abstraction of network states for cognitive functions. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

[13] J. Ali-Tolppa and M. Kajó. Mobility and qos prediction for dynamic coverage optimization. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–2, 2020.

[14] J Meinilä, P Kyösti, L Hentilä, T Jämsä, EK Essi Suikkanen, and M Narandzic. Wireless world initiative new radio winner+, d5. 3: Winner+ final channel models. *CELTIC Telecommunication Soultions, Tech. Rep*, 2010.

[15] S. Hämäläinen, H. Sanneck, and C. Sartori. *LTE self-organising networks (SON): network management automation for operational efficiency*. John Wiley & Sons, 2012.

[16] D. P Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.