SON Conflict Resolution using Reinforcement Learning with State Aggregation

Ovidiu Constantin Iacoboaiea, Berna Sayrac, Sana Ben Jemaa
Orange Labs
38-40 rue du General Leclerc 92130
Issy les Moulineaux, France
{ovidiu.iacoboaiea,berna.sayrac,sana.benjemaa}@orange.com

Pascal Bianchi
Telecom ParisTech
37 rue Dareau 75014
Paris, France
pascal.bianchi@telecom-paristech.fr

ABSTRACT

In future generation networks one of the main focuses is on automating the network optimization. This is done through so called Self Organizing Network (SON) functions. A SON instance is a realization of a SON function that governs one or several cells. Several independent SON instances of one or multiple SON functions are likely to generate conflicts. This raises the need for a SON COordinator (SONCO) meant to solve these conflicts. In this paper we consider that each SON function has one SON instance on every cell and we present the design of a SONCO function for coordinating all these instances. The SONCO solves the conflicts that appear on the update requests arbitrating (i.e. accepting/denying the requests) so that it minimizes a predefined regret. This regret takes into account the weights associated to the SON functions that rank their importance according to the operator policies. We solve the problem in a Reinforcement Learning (RL) framework as it offers the possibility to improve the decisions based on past experiences. We employ a state-aggregation technique to make the state-space of our solution scale linearly with the number of cells. We provide a study case for two SON functions: Mobility Load Balancing (MLB) tuning the Cell Individual Offset(CIO) and Mobility Robustness Optimization (MRO) tuning the CIO together with the handover hysteresis. The proposed SONCO function solves the conflicts on the CIO update requests. Numerical results show how the proposed SONCO is able to favor either MLB or MRO requests according to their associated weights.

Categories and Subject Descriptors

A.m [**General Literature**]: Miscellaneous

Keywords

SON Coordination; MLB; MRO; SON instances; LTE; reinforcement learning; state aggregation;

AllThingsCellular'14, August 22, 2014, Chicago, IL, USA.

http://dx.doi.org/10.1145/2627585.2627591 .

1. INTRODUCTION

The continuous growth of traffic demand is forcing operators to improve their network capabilities by technological upgrades (LTE Advanced [11]), network densification and introduction of Heterogeneous Networks (HetNets). These upgrades lead to increased CAPital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX). In order to reduce the costs, Release 8 of 3GPP has introduced Self Organizing Networks (SON) that replace the costly human intervention with automated mechanisms. These mechanisms are usually classified into 3 categories Self-Configuration, Self-Optimization and Self-Healing. We focus on the second category which provides algorithms that offer a run-time optimization of the network. In the sequel SON will refer to self-optimization.

In a real network we may find more than one SON function (e.g. Mobility Load Balancing (MLB), Mobility Robustness Optimization (MRO), etc.), each of which is trying to optimize some Key Performance Indicators (KPIs) by tuning a set of parameters. This may cause different types of conflicts [6]. We therefore need a SON COordination (SONCO) mechanism to deal with these conflicts. The SON instances will not directly execute the desired parameter changes on the network, instead they formulate requests to the SONCO and it is the SONCO that decides which request will be executed.

The SONCO concept has been introduced fairly recently with Release 10 of 3GPP [6]. There are two tracks followed in finding coordination solutions. On the one hand the SON instances are seen as black-boxes (i.e. there is no or limited information on the algorithm and its inputs). The existing work on this track has so far focused on solving conflicts based on the instantaneous network conditions disregarding the outcome of the past decisions: a coordination mechanism which attributes priorities to the SON functions can be found in [10] and [9]; decision trees are used in [3] and [2]; in [8] the solution is based on restrictions on the value set of the parameters; a per user optimization, is proposed in [4] where priorities are attributed to users for handovers (HOs) based on the weights attributed to the SON functions (MRO and MLB). On the other hand the second track considers the SON instances as white-boxes (i.e. the SONCO knows the used algorithms and their characteristics, e.g. [5]), but this is not always feasible for an operator-centric SONCO especially in a multi-vendor environment.

In our work we follow the first track considering the design of an operator-centric SONCO that sees the SON in-

stances as black-boxes, and unlike current approaches on this track we take advantage of the information on the outcomes of past decisions. Since the SON functions are considered as black-boxes there will be an inevitable uncertainty on the impact of the SONCO's decisions on the SON functions and on the KPIs. In order to ensure a proper conflict resolution and KPI performance, this uncertainty must be minimized. For this purpose in [7] we proposed the Reinforcement Learning (RL) [12] framework where we can use to some extent the information regarding the impact of our past decisions. We employed RL making use of a centralized value function with a state-space composed of all the possible configurations of a restricted set of network parameters. The inconvenient in this case is that the state-space scales exponentially with the number of cells. This paper represents a continuation of the work in [7] where now we focus on a more scalable RL solution by using a distributed value function. We summarize the contribution of this paper as follows:

- we provide an operator-centric SONCO solution where the SON instances are considered as black-boxes,
- we consider that the SON requests include an *unhappiness metric* (an indicator of how unsatisfied the SON instances are with the current parameter configuration),
- we use a RL-based SONCO ,
- we show that in our conditions the action-value function can be simplified and expressed as a function of the parameter configurations,
- we employ a state-aggregation technique to obtain a distributed value function whose state-space scales linearly with the number of cells,
- we present a study case with MLB and MRO instantiated on each and every cell.

The rest of this paper is organized as follows: Section 2 provides the system description presenting the scenario with the SON instances. Section 3 outlines the Markov Decision Process (MDP) underlying the RL-solution together with the state aggregation and in Section 4 we introduce the RL algorithm and we characterize its complexity. Simulation results are included in Section 5 and Section 6 concludes the paper.

2. SYSTEM DESCRIPTION

For simplicity we make the following notation convention: for any variable X (be it parameter, update request, action, etc.) if we index it by a set $\mathcal{I}$ the meaning is $X_{\mathcal{I}} = (X_i)_{i\in\mathcal{I}}$.

We consider a network segment composed of N cells (Fig. 1) indexed by $n \in \mathcal{N} = \{1, ..., N\}$. Let $\mathcal{N}_n \subset \mathcal{N}$, $\forall n \in \mathcal{N}$, be a set containing $\{n\}$ and the neighbors of cell n. Denote $N_n = |\mathcal{N}_n|$, $\forall n \in \mathcal{N}$.

For the network optimization we have Z SON functions (e.g. MLB, MRO, etc.) indexed by $z \in \mathcal{Z} = \{1, .., Z\}$. On each cell we have one SON instance of each SON function. The SON instances create update requests for changing the network parameters that they tune.

We assume that on all cells there are K parameters (e.g. CIO, Hysteresis, antenna tilt, etc.) tuned by the instances

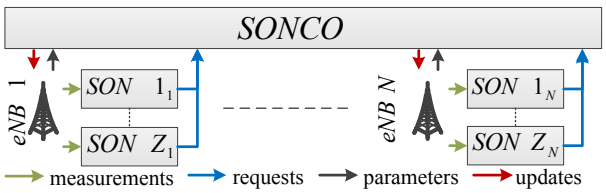

Figure 1: Functional Block Diagram: SONCO↔SON interactions

of the SON functions indexed by $k \in \mathcal{K} = \{1, .., K\}$. Some parameters, say the parameters $\{1, .., \bar{K}\}$ represent an important source of conflicts. We refer the reader to [6, Section 9.1.] for the selection of these parameters in a practical settings. Denote $\bar{\mathcal{K}} = \{1, .., \bar{K}\}$.

Let $P_{t,n,k}$, $\forall (t, n, k) \in \mathbb{N} \times \mathcal{N} \times \mathcal{K}$, be the value of parameter k on cell n at time t and consider $\mathcal{P}_k$ to be the set of possible values of $P_{t,n,k}$. Denote $\mathcal{P} = \prod_{k\in\mathcal{K}} \mathcal{P}_k$ and $\bar{\mathcal{P}} = \prod_{k\in\bar{\mathcal{K}}} \mathcal{P}_k$.

We assume that the SON instances send update requests in order to modify the configurations of the parameters on the hosting cell. An update request is a value $u \in [-1; 1]$ where: $u = 0$, $u \in [-1; 0)$ and $u \in (0; 1]$ is equivalent to a request to maintain, decrease and increase the value of the targeted parameter, respectively; $|u|$ is a measure of how *unhappy* the SON instance is with the current parameter configuration (the closer to 1 the more *unhappy* the SON instance is). If the SON instance does not tune a given parameter than we consider the corresponding request to be `void`. Let $U_{t,n,k,z}$, $\forall (t, n, k, z) \in \mathbb{N} \times \mathcal{N} \times \mathcal{K} \times \mathcal{Z}$, be the update request sent at time t by the instance of the SON function z that runs on cell n, to change parameter k and consider $\mathcal{U}_k \subset [-1; 1] \cup \{\texttt{void}\}$ to be the set of possible values of $U_{t,n,k,z}$. Denote $U_t = U_{t,\mathcal{N},\mathcal{K},\mathcal{Z}}$, $\mathcal{U} = \prod_{k\in\mathcal{K}} \mathcal{U}_k$ and $\bar{\mathcal{U}} = \prod_{k\in\bar{\mathcal{K}}} \mathcal{U}_k$. We leave the details of how the SON instances should quantify their unhappiness for future work.

3. SON COORDINATION

The SON instances do not directly execute the desired parameter changes in the network, instead update requests are sent to a SONCO which decides if they are accepted or denied. The accepted ones are immediately executed. We design an operator-centric SONCO which sees the SON instances as black-boxes (it does not know the input or the algorithm inside). It only knows the update requests (U_t) and the current parameter configuration of the network (P_t). Based on this the SONCO has to find a reasonable solution for conflict resolution.

We consider that all the SON instances are synchronized, i.e. they do their KPI evaluation within a time interval T and they send the requests simultaneously to the SONCO at the end of the time interval (Fig. 1) . The SONCO has the same time granularity T. It decides which requests are accepted and which are denied, so the task of the SONCO is to provide a reasonable conflict resolution. We use RL as it allows us to keep track of our past decisions through value functions which reflect how satisfied we were with a configuration. The RL algorithm is based on Markov Decision Process (MDP) that we describe in the next sub-section.

3.1 MDP. General framework

We define the underlying MDP over all the network:

- **State space:** $\mathcal{S} = \mathcal{P}^N \times \mathcal{U}^{N \cdot Z}$. A state $s \in \mathcal{S}$ is written as $s = (p, u) = (p_{\mathcal{N},\mathcal{K}}, u_{\mathcal{N},\mathcal{K},\mathcal{Z}})$ contains the parameter configurations and update requests.
- **Action space:** $\mathcal{A} = \mathcal{A}_3^{N \cdot K}$, $\mathcal{A}_3 = \{\pm 1, 0\}$. An action $a \in \mathcal{A}$ is composed of one NK-uplet. The action allows to increase/decrease the value of a parameter only if there exists at least one request to do so. Concretely the impact on parameter k ($\forall k \in \mathcal{K}$) of cell n ($\forall n \in \mathcal{N}$) is:

$$\begin{cases} \uparrow, \text{ if } \mathbb{I}_{\{a_{n,k}=1\}} \mathbb{I}_{\{\exists z \in \mathcal{Z} \text{ s.t. } u_{n,k,z}>0\}} = 1 \\ \downarrow, \text{ if } \mathbb{I}_{\{a_{n,k}=-1\}} \mathbb{I}_{\{\exists z \in \mathcal{Z} \text{ s.t. } u_{n,k,z}<0\}} = 1 \\ \updownarrow, \text{ otherwise} \end{cases} \quad (1)$$

  where $\uparrow$, $\downarrow$ and $\updownarrow$ mean increase, decrease and maintain the value of the parameter, respectively. $\mathbb{I}_{\{\bullet\}}$ is the indicator function ($\mathbb{I}_{\{true\}} = 1$, $\mathbb{I}_{\{false\}} = 0$).
- **Transition kernel:** $\mathcal{T}(s'|s,a)$ is the probability of going into state s' when the current state-action pair is (s,a).
- **Regret:** $r(s,a)$ is the regret associated to the state-action pair (s,a).

A policy π is a transition kernel π on $\mathcal{S} \times 2^{\mathcal{A}}$, such that $\pi(s, \{a\})$ represents the probability to take action a when the current state is s.

Consider a stochastic process $(S_t, A_t)_{t \in \mathbb{N}} \in \mathcal{S} \times \mathcal{A}$ where S_t and A_t represent the state and action at time t respectively. Set $S_t = (P_t, U_t)$.

For any policy π introduce a probability $\mathbb{P}_\pi$ such that $(S_t, A_t)_{t \in \mathbb{N}}$ is a Markov chain under $\mathbb{P}_\pi$ thus:

$$\mathbb{P}_\pi \left(S_{t+1} = s' | S_t = s, A_t = a \right) = \mathcal{T}\left(s' | s, a \right), \quad (2)$$

$$\mathbb{P}_\pi \left(A_t = a | S_t = s \right) = \pi \left(s, a \right). \quad (3)$$

Note that we shall simply use the notation $\mathbb{P}$ instead of $\mathbb{P}_\pi$ in eq. (2) i.e. when the probability does not depend on π.

3.2 Assumptions

Our transition kernel has some particular characteristics. The future configuration of parameter $k \in \mathcal{K}$ on cell $n \in \mathcal{N}$ ($P_{t+1,n,k}$) is a deterministic function of: the current value of parameter k on cell n ($P_{t,n,k}$) together with the corresponding update requests ($U_{t,n,k,\mathcal{Z}}$) and action ($A_{t,n,k}$) .

Assumption 1 (kernel). *There exists the deterministic functions $g_k : \mathcal{P}_k \times \mathcal{U}_k^Z \times \mathcal{A}_3 \to \mathcal{P}_k$, $\forall k \in \mathcal{K}$, s.t. $P_{t+1,n,k} = g_k \left(P_{t,n,k}, U_{t,n,k,\mathcal{Z}}, A_{t,n,k} \right)$, $\forall n \in \mathcal{N}$, $\forall k \in \mathcal{K}$.*

Furthermore the update request of the SON instance of the SON function $z \in \mathcal{Z}$ concerning parameter $k \in \mathcal{K}$ on cell $n \in \mathcal{N}$ ($U_{t+1,n,k,z}$) depends only on the parameter configuration on the parameters $\mathcal{K}$ on cell n and its neighbors ($P_{t+1,\mathcal{N}_n,\mathcal{K}}$).

Assumption 2 (kernel). *$\forall u' \in \mathcal{U}^{NZ}$, $\forall p' \in \mathcal{P}^N$, $\forall (n,k,z) \in \mathcal{N} \times \mathcal{K} \times \mathcal{Z}$, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$, we have that: $\mathbb{P}\left(U_{t+1,n,k,z} = u'_{n,k,z} | S_t = s, A_t = a, P_{t+1} = p' \right) = \mathbb{P}\left(U_{t+1,n,k,z} = u'_{n,k,z} | P_{t+1,\mathcal{N}_n,\mathcal{K}} = p'_{\mathcal{N}_n,\mathcal{K}} \right)$.*

Given some state-action pair the regret is a function of the *unhappiness* of the SON instances reflected in the succeeding update requests concerning the conflicting parameters $\bar{\mathcal{K}}$. We define the *instantaneous* regret at time t as the sum of per cell *instantaneous* regrets: $R_t = \sum_{n \in \mathcal{N}} R_{t,n}$ where $\forall n \in \mathcal{N}$, $R_{t,n} = \rho\left(U_{t+1,n,\bar{\mathcal{K}},\mathcal{Z}} \right)$ for some function $\rho : \bar{\mathcal{U}}^Z \to \mathbb{R}$.

Assumption 3 (regret). $r\left(s,a\right) = \sum_{n \in \mathcal{N}} r_n\left(s,a\right)$ *where,* $\forall n \in \mathcal{N}$, $r_n\left(s,a\right) = \mathbb{E}\left[R_{t,n} | S_t = s, A_t = a \right]$.

We now provide a specific form of ρ, relevant for the SON coordination. Consider that ρ is a function of the absolute values of the update requests reflecting a maximum regret per cell , over all SON functions in $\mathcal{Z}$ and parameters in $\bar{\mathcal{K}}$ (say the target is to minimize the sum of the maximum per cell regret). In other words :

$$\rho\left(u_{n,\bar{\mathcal{K}},\mathcal{Z}} \right) = \max_{(k,z) \in \bar{\mathcal{K}} \times \mathcal{Z}} \left| u_{n,k,z} \right|, \ \forall n \in \mathcal{N} \quad (4)$$

3.3 Value functions

For any policy π we introduce the state-value function (V^π) and the action-value function (Q^π):

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s \right] , \quad (5)$$

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s, A_0 = a \right]. \quad (6)$$

where $0 \leq \gamma < 1$ is the regret sum discount factor and $\mathbb{E}_\pi$ is the expectation given that the followed policy is π.

The following proposition allows to simplify (6). In this scope, we first define the following:

- $\forall p \in \mathcal{P}^N$,$\forall n \in \mathcal{N}$, $\overline{r}_n : \mathcal{P}^{N_n} \to \mathbb{R}$ where $\overline{r}_n\left(p_{\mathcal{N}_n,\mathcal{K}} \right) = \mathbb{E}\left[\rho\left(U_{1,n,\bar{\mathcal{K}},\mathcal{Z}} \right) | P_{1,\mathcal{N}_n,\mathcal{K}} = p_{\mathcal{N}_n,\mathcal{K}} \right]$,
- $g : \mathcal{S} \times \mathcal{A} \to \mathcal{P}^N$ where $\forall (p,u) \in \mathcal{S}$, $\forall a \in \mathcal{A}$, $g\left((p,u), a \right) = \left(g_k\left(p_{n,k}, u_{n,k,\mathcal{Z}}, a_{n,k} \right) \right)_{(n,k) \in \mathcal{N} \times \mathcal{K}}$,

Proposition 1. *For any policy π there exists a set of functions $W_n^\pi : \mathcal{P}^N \to \mathbb{R}$, $\forall n \in \mathcal{N}$, s.t. for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, $Q^\pi(s,a) = \sum_{n \in \mathcal{N}} W_n^\pi\left(g\left(s,a\right) \right)$. Moreover, $W_{\mathcal{N}}^\pi$ solves the following fixed point equation:*

$$W_n^\pi(p) = \overline{r}_n\left(p_{\mathcal{N}_n,\mathcal{K}} \right) + \gamma \sum_{u \in \mathcal{U}^{NZ}} \mathbb{P}\left[U_1 = u | P_1 = p \right] \cdot \sum_{a \in \mathcal{A}} \pi\left((p,u), a \right) \cdot W_n^\pi\left(g\left((p,u), a \right) \right), \ \forall n \in \mathcal{N} \quad (7)$$

Proof. See Appendix A. □

Remark 1 (optimal policy). *If a policy π^* minimizes the value function $\forall s \in \mathcal{S}$ then it is said to be* optimal *[12]. Note that π^* is known to be a deterministic policy i.e. (using a small notation abuse) $\pi^* : \mathcal{S} \to \mathcal{A}$ and we have (see [12]): $\pi^*(s) = arg\ max_a Q^*(s,a)$ (Q^* is the* optimal action-value function*). Thus eq. (7) for the optimal policy can be recovered by:*

$$\begin{aligned} & W_n^*(p) = \overline{r}_n\left(p_{\mathcal{N}_n,\mathcal{K}} \right) + \gamma \textstyle\sum_{u \in \mathcal{U}^{NZ}} \mathbb{P}\left[U_1 = u | P_1 = p \right] \cdot W_n^*\left(p^* \right), \ \forall n \in \mathcal{N}, \\ & p^* = g\left((p,u), a^* \right), \\ & a^* = \pi^*(p,u) = \arg\min_{a \in \mathcal{A}} W^*\left(g\left((p,u), a \right) \right) \end{aligned} \quad (8)$$

Note that $(W_n^*)_{n \in \mathcal{N}}$ has to be processed jointly as the policy π^* is centralized, i.e. it is a function of W^*.

3.4 State aggregation. Sub-optimal policy.

Although Proposition 1 allows to simplify (6) to a function with a reduced state-space $\mathcal{P}^N$ it still scales exponentially with the number of cells N. Therefore in the sequel we set to perform a state-aggregation, this one at the cost of possible performance loss. Note that W_n^π depends on p mainly through $p_{\mathcal{N}_n,\bar{\mathcal{K}}}$ (i.e. the values of the conflicting parameters on cell n and its neighbors). Thus we perform a state aggregation as follows: let let $\mathcal{W}_n$ be the set of functions $W_n^\pi : \mathcal{P}^N \to \mathbb{R}$ such that $\forall p \in \mathcal{P}^N$, $W_n^\pi(p)$ depends only on $p_{\mathcal{N}_n,\bar{\mathcal{K}}}$, namely:

$$\mathcal{W}_n = \left\{ p \mapsto F\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) : F \in \mathbb{R}^{\mathcal{Y}_n} \right\} \tag{9}$$

where $\mathcal{Y}_n = \bar{\mathcal{P}}^{N_n}$; therefore instead of directly computing $W_{\mathcal{N}}^*$ as the solution to (8) we aim to evaluate its projection onto $\mathcal{W} = \mathcal{W}_{\mathcal{N}}$ ($\equiv$ an approximation of $W_{\mathcal{N}}^*$) of the form:

$$f\left(p_{\mathcal{N},\bar{\mathcal{K}}}\right) = \textstyle\sum_{n\in\mathcal{N}} f_n\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) \tag{10}$$

for some $f_n \in \mathbb{R}^{\mathcal{Y}_n}$.

4. REINFORCEMENT LEARNING

4.1 Algorithm

First we define $\bar{g} : \bar{\mathcal{P}}^N \times \bar{\mathcal{U}}^{NZ} \times \mathcal{A}_3^{N\bar{K}} \to \bar{\mathcal{P}}^N$ where $\bar{g}\left((p,u),a\right) = \left(g_k\left(p_{n,k}, u_{n,k,\mathcal{Z}}, a_{n,k}\right)\right)_{(n,k)\in\mathcal{N}\times\bar{\mathcal{K}}}$.

We cannot directly calculate f as we only have partial knowledge on the transition kernel, instead we use the flowing recursion $\forall n \in \mathcal{N}$:

$$\begin{aligned} f_{t+1,n}\left(P_{t,\mathcal{N}_n,\bar{\mathcal{K}}}\right) &= (1-\alpha) f_{t,n}\left(P_{t,\mathcal{N}_n,\bar{\mathcal{K}}}\right) + \\ &\quad \alpha\left(R_{t,n} + \gamma f_{t,n}\left(\bar{P}_{t+1,\mathcal{N}_n,\bar{\mathcal{K}}}\right)\right) \\ \bar{P}_{t+1} &= \bar{g}\left(\left(P_{t,\mathcal{N},\bar{\mathcal{K}}}, U_{t,\mathcal{N},\bar{\mathcal{K}},\mathcal{Z}}\right), \bar{A}_t\right) \\ \bar{A}_t &= \underset{a\in\{\pm1,0\}^{N\bar{K}}}{\arg\min}\ f_t\left(\bar{g}\left(\left(P_{t,\mathcal{N},\bar{\mathcal{K}}}, U_{t,\mathcal{N},\bar{\mathcal{K}},\mathcal{Z}}\right), a\right)\right) \end{aligned} \tag{11}$$

where $\bar{A}_t$ represents the optimal action concerning parameters in $\bar{\mathcal{K}}$ based on f_t. As we are using a fixed α, f_t converges in the mean to a fixed point of the so-called Bellman operator *projected onto* $\mathcal{W}$ [12]. Note that the f_t is independent of the parameters in $\check{\mathcal{K}} = \mathcal{K}\backslash\bar{\mathcal{K}}$ (as intended from the state aggregation), this allows us to always accept the requests on these parameters as there are no conflicts on these parameters:

$$\begin{aligned} A_{t,\mathcal{N},\check{\mathcal{K}}} = \check{A}_t = \Big(&\mathbb{I}_{\left\{\exists z\in\mathcal{Z} \text{ s.t. } U_{t,n,k,z}>0\right\}} - \\ &\mathbb{I}_{\left\{\exists z\in\mathcal{Z} \text{ s.t. } U_{t,n,k,z}<0\right\}}\Big)_{(n,k)\in\mathcal{N}\times\check{\mathcal{K}}}. \end{aligned} \tag{12}$$

For practical reasons we use an ϵ-greedy policy:

$$\pi_t\left(S_t,\{a\}\right) = \left((1-\epsilon)\,\mathbb{I}_{\left\{a_{\mathcal{N},\bar{\mathcal{K}}}=\bar{A}_t\right\}} + \frac{\epsilon}{3^{NK}}\right)\mathbb{I}_{\left\{a_{\mathcal{N},\bar{\mathcal{K}}}=\bar{A}_t\right\}} \tag{13}$$

The proposed algorithm is summarized in Alg. 1 . **Function Init** should be called for the initialization of the algorithm and **Function SONCO** should be called every time after receiving the requests of the SON instances.

ALGORITHM 1 (SONCO).
Function Init :
For all $n \in \mathcal{N}$, initialize $f_n(p') = 0, \forall p' \in \bar{\mathcal{P}}^{N_n}$
Function SONCO :
Observe current parameter configurations p and update requests u, calculate regret $r_n = \rho\left(u_{n,\bar{\mathcal{K}},\mathcal{Z}}\right)$, $\forall n \in \mathcal{N}$
Calculate $\bar{a} = arg\ minf\left(\bar{g}\left(\left(p_{\mathcal{N},\bar{\mathcal{K}}}, u_{\mathcal{N},\bar{\mathcal{K}},\mathcal{Z}}\right), a\right)\right)$ and $\bar{p} = \bar{g}\left(\left(p_{\mathcal{N},\bar{\mathcal{K}}}, u_{\mathcal{N},\bar{\mathcal{K}},\mathcal{Z}}\right), \bar{a}\right)$
For all $n \in \mathcal{N}$
$f_n\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) \leftarrow (1-\alpha) f_n\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) + \alpha\left(r_n + \gamma f_n\left(\bar{p}_{\mathcal{N}_n,\bar{\mathcal{K}}}\right)\right)$
Choose action a using an ϵ-greedy policy , Take action a.

4.2 Complexity analysis

The optimal policy is usually obtained through Q-Learning [12] with one SONCO that governs all cells. According to the previous section the optimal policy can also be obtained by learning W^* in (7). This allows us to tremendously reduce the required state (and action) space from a size of: $\Psi_V = |\mathcal{S}| \cdot |\mathcal{A}| = |\mathcal{P}|^N |\mathcal{U}|^{NZ} \cdot |\mathcal{A}_3|^{NK}$ to a size of $\Psi_W = N|\mathcal{P}|^N$. Still, this solution scales exponentially with N, but as mentioned we perform a state aggregation coming to a state space size of $\Psi_f = \sum_{n\in\mathcal{N}} \left|\bar{\mathcal{P}}\right|^{N_n}$, and one can see that this scales linearly with the number of cells.

5. SIMULATION RESULTS

5.1 Simulation scenario

To demonstrate the concept we use 2 SON functions ($\mathcal{Z} = \{1,2\}$): one MLB function ($z = 1$) and one MRO function ($z = 2$). We consider 2 parameters ($\mathcal{K} = \{1,2\}$) of interest the CIO ($k = 1$, tuned by both MLB and MRO instances) and the HO hysteresis ($k = 2$, tuned only by MRO). The parameter on which there are request conflicts is the CIO thus say $\bar{\mathcal{K}} = \{1\}$. For simulation purposes we present their implementation in the sequel.

The CIO and the Hysteresis are two parameters that are used in mobility management as follows: a User Equipment (UE) that wants to transmit data will attach to cell $n_0 = \text{argmax}_{n\in\mathcal{N}}\left(RSRP_n + C_n\right)$ where $RSRP_n$ is the Reference Signal Received Power from cell n and C_n is the CIO of cell n; when attached to a serving cell n_S ($\forall n_S \in \mathcal{N}$) a UE performs a HO to a target cell $n_T \neq n_S$ if $n_T = \text{argmax}_{n\in\mathcal{N}}\left(RSRP_n + C_n + H_{n_S}\mathbb{I}_{\{n=n_S\}}\right)$ where H_{n_S} is the Hysteresis of cell n_S.

5.1.1 Mobility Load Balancing

For the MLB SON function, the input (i.e. the optimized metric) is the cell load (average number of occupied Physical Resource Blocks - PRBs) of the hosting cell (the cell on which the MLB instance runs). The tuned parameter is the CIO (C) of the hosting cell. For cell n, the update request of the MLB instance at time t for the CIO can be expressed as:

$$\begin{aligned} U_{t,n,1,1} = &-\varphi\left(L_{t,n}; g_1, m_1\right)\mathbb{I}_{\left\{L_{t,n}>\mathbb{T}_{ld}^H\right\}} + \\ &\left(\varphi\left(L_{t,n}; g_2, m_2\right) - 1\right)\mathbb{I}_{\left\{L_{t,n}<\mathbb{T}_{ld}^L\right\}} \end{aligned} \tag{14}$$

where $L_{t,n}$is the load on cell n at time t, $\mathbb{T}_{ld}^H$ and $\mathbb{T}_{ld}^L$ are fixed thresholds used by the MLB instance to trigger CIO modification requests ($\mathbb{T}_{ld}^H > \mathbb{T}_{ld}^L$); g is the steepness and m the center of a S-shaped function of x with values in $[0;1]$: $\varphi(x;g,m) = 1/\left(1 + e^{-g\cdot(x-m)}\right)$. The MLB does not send any request concerning the Hysteresis, therefore $U_{t,n,2,1} =$ `void`, $\forall\,(t,n) \in \mathbb{N}\times\mathcal{N}$.

5.1.2 Mobility Robustness Optimization

We consider a distributed implementation of the MRO where the MRO instances running on neighboring cells communicate with each other. At time instant t, the MRO instance running on cell n has the following metrics as input: the number of HO ping pongs ($N^P_{t,n}$), the number of too late HOs ($N^L_{t,n}$), the number of too early HOs ($N^E_{t,n}$) and the number of HOs to a wrong cell ($N^W_{t,n}$) which all originate from cell n, together with the number of too late HOs ($N^{L*}_{t,n}$) where cell n is the HO target cell. Since the MRO tunes two parameters, the CIO ($k = 1$) and the Hysteresis ($k = 2$), the MRO instance of cell n at time t sends 2 simultaneous requests: one for each parameter. The one regarding the Hysteresis can be expressed as follows:

$$\begin{aligned} U_{t,n,2,2} &= \mathbb{I}_{\{q^2_{t,n}=0\}} - \mathbb{I}_{\{q^1_{t,n}>0\}} (\in \{-1,0,1\}) \\ q^1_{t,n} &= \mathbb{I}_{\{N^L_{t,n}>\mathbb{T}^H_L\}} \mathbb{I}_{\{N^E_{t,n}<\mathbb{T}^L_E\}} \mathbb{I}_{\{N^W_{t,n}<\mathbb{T}^L_W\}} \mathbb{I}_{\{N^P_{t,n}<\mathbb{T}^L_P\}} \\ q^2_{t,n} &= \mathbb{I}_{\{N^L_{t,n}\geq\mathbb{T}^L_L\}} + \mathbb{I}_{\{N^E_{t,n}\leq\mathbb{T}^H_E\}} \mathbb{I}_{\{N^W_{t,n}\leq\mathbb{T}^H_W\}} \mathbb{I}_{\{N^P_{t,n}\leq\mathbb{T}^H_P\}} \end{aligned} \quad (15)$$

where $\mathbb{T}^H_{(\cdot)}$ and $\mathbb{T}^L_{(\cdot)}$ are fixed thresholds used to trigger events ($\mathbb{T}^H_{(\cdot)} > \mathbb{T}^L_{(\cdot)}$).

The request regarding the CIO can be expressed as follows:

$$\begin{aligned} U_{t,n,1,2} &= \left(\varphi\left(N^{L*}_{t,n}/\mathbb{T}^H_{L*}; g_3, m_3\right) - 1\right) \mathbb{I}_{\{q^3_{t,n}=1\}} \\ q^3_{t,n} &= \mathbb{I}_{\{\exists i \in \mathcal{N}_{-n} \text{ s.t. } U_{t,i\to n}=1\}} \end{aligned} \quad (16)$$

where $\mathcal{N}_{-n} = \mathcal{N} \setminus \{n\}$. $U_{t,i\to n}$ is an *intra-MRO message*. The MRO instance running on cell i sends to its neighboring MRO instances ($j \in \mathcal{N}_{-i}$) a message requesting a CIO increase if needed:

$$\begin{aligned} U_{t,i\to j} &= \mathbb{I}_{\{q^1_{t,i}=0\}} \mathbb{I}_{\{q^2_{t,i}>0\}} \mathbb{I}_{\{q^5_{t,i}=0\}} \mathbb{I}_{\{j\in\xi(i)\}}, \\ q^5_{t,i} &= \mathbb{I}_{\{N^L_{t,i}\leq\mathbb{T}^H_L\}} \mathbb{I}_{\{N^E_{t,i}\leq\mathbb{T}^H_E\}} \mathbb{I}_{\{N^W_{t,i}\leq\mathbb{T}^H_W\}} \mathbb{I}_{\{N^P_{t,i}\leq\mathbb{T}^H_P\}} \end{aligned} \quad (17)$$

where $\xi(i)$ is the set on cell from which there are Too Late HOs to cell i. Increasing the CIO values of neighboring cells reduces the number of too late HOs originating from them.

5.1.3 Scenario

The scenario is built on a network segment of $N = 21$ cells with wraparound (Fig. 2). The simulation details are summarized in Table 1. We consider an elastic FTP-like traffic with general background traffic arrival rate η_G [Mb/s] together with an additional hotspots (HS) arrival rate η_{HS} [Mb/s] (such that the resulting arrival rate in the HS is $\eta_G+\eta_{HS}$). The user arrivals rates per area unit can be easily obtained as: $\rho_{(\cdot)}\left[UE/s/m^2\right] = \eta_{(\cdot)}[Mb/s]/S_{(\cdot)}\left[m^2\right]/FS[Mb/UE]$ (S refers to the area and FS to the fixed file size). We use Space Poisson Point Processes for the user arrivals. A user arrives in the network, transmits its file and then leaves the network. We consider 3 traffic HSs.

The SONCO and the SON (MLB and MRO) instances are active during the entire simulation.

5.2 Simulation results

We consider a fixed file size of $FS = 16[Mb/UE]$. The user arrival rates are $\eta_G = 189[Mb/s]$ and $\eta_{HS} = 90[Mb/s]$. We try out several sets of weight pairs $w = (w_{MLB}, w_{MRO})$ to see the impact on the KPIs. The total simulation duration is 48 hours; the KPIs in the results are calculated based on the statistics over the last 24 hours period.

Table 1: Simulation parameters

Category	Parameter	Value
Network	Inter Site Distance	1732 m
Channel modeling	Carrier frequency	2 GHz
	Bandwidth	10 MHz
	cell TX Power	46 dBm
	Propagation Model	3GPP Case 3 [1]
	Channel Model	MIMO 2×2
Mobility	HO Time To Trigger	160 ms
SON instances	CIOs ($\mathcal{P}_1$[dB])	$\{-12, -8, ..., 0\}$
	HO Hystereses ($\mathcal{P}_2$ [dB])	$\{0, 1, ..., 12\}$
	Update requests ($\mathcal{U}_{(\cdot)}$)	$[-1; 1] \cup \{\texttt{void}\}$
	Time window T	5 min
	$\left(\mathbb{T}^L_{ld}; \mathbb{T}^H_{ld}\right)$	$(0.3; 0.8)$
	$\left(\mathbb{T}^L_L; \mathbb{T}^H_L\right)$,$\left(\mathbb{T}^L_P; \mathbb{T}^H_P\right)$,$\mathbb{T}^H_{L*}$	$(2; 8)$,$(2.5; 10)$,6
	$\left(\mathbb{T}^L_E; \mathbb{T}^H_E\right) = \left(\mathbb{T}^L_W; \mathbb{T}^H_W\right)$	$(\infty; \infty) \equiv$ off
	$(g_1; m_1)$, $(g_2; m_2)$	$(30; 0.9)$, $(5; -0.2)$
	$(g_3; m_3)$	$(5; 1.5)$
SONCO	$(\alpha; \gamma; \epsilon)$	$(0.2; 0.8; 0.1)$

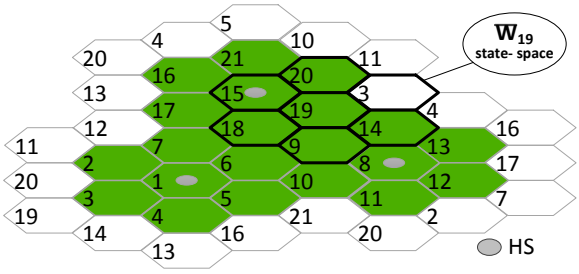

Figure 2: Network topology

In Figures 3, 4 and 5 we plot the maximum and the average (over all cells) of the time-averaged cell load, the time-averaged number of too late HOs and the time-averaged number of ping-pongs, respectively.

In Fig. 3 one can see that a better load balancing is achieved by giving bigger weights to the MLB instances. Basically the overloaded cells (the ones containing the traffic HSs) are allowed to *off-load* more. On the other hand giving a bigger priority to the MRO prevents these cells from *off-loading* as much as the MLB would want (i.e. to decrease the CIO) causing the maximum load to degrade by up to 9.9%. The HO borders are pushed *further away* from the overloaded cells. Thus the number of Too Late HOs towards the overloaded cells from neighboring base stations is reduced at the cost of slightly increasing the number of Too Late HO from these cells towards their neighbors. Overall we reduce the maximum number of too late HOs by up to 28.7% (Fig. 4).

There are very few Too Early HOs and Wrong Cell HO, so they are not significant. Typically the number of ping pongs depends mostly on the Hystereses, but as we can see in Fig. 5 they are also impacted by the weights. If we reduce the number of too late HOs by means of CIO configurations then the MRO has more flexibility in tuning the Hysteresis in order to further reduce the number of ping-pongs (at most 33.6%).

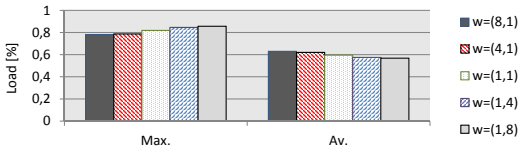

Figure 3: Average Load

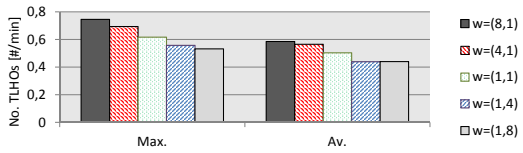

Figure 4: Average Number of Too Late HOs

6. CONCLUSIONS AND FUTURE WORK

In dealing with the conflict resolution between the requests of the SON instances RL proves to have the qualities that allow us to intelligently decide when to accept/deny these requests in order to tune the arbitration according to the operator's preferences (avoiding over-loads for MLB versus decreasing the number of connection failures and ping-pongs due to mobility for MRO). State aggregation allows us to make the state-space scale linearly with the number of cells. Simulation results show that the resulting KPIs reflect the operator preferences. Future work will focus on further improving the scalability of our solution and analyzing other regret functions.

7. ACKNOWLEDGMENTS

The research leading to these results has been carried out within the FP7 SEMAFOUR project and has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 316384.

8. REFERENCES

[1] 3GPP. E-UTRA; Further advancements for E-UTRA physical layer aspects. TR 36.814, 2010.

[2] T. Bandh, R. Romeikat, H. Sanneck, and H. Tang. Policy-based coordination and management of son functions. In *ISINM*, 2011.

[3] T. Bandh, H. Sanneck, and R. Romeikat. An experimental system for son function coordination. In *VTC Spring*, 2011.

[4] J. Chen, H. Zhuang, B. Andrian, and Y. Li. Difference-based joint parameter configuration for MRO and MLB. In *VTC Spring*, 2012.

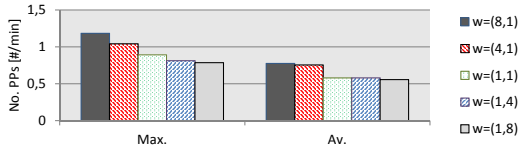

Figure 5: Average Number of Ping Pongs

[5] R. Combes, Z. Altman, and E. Altman. Coordination of autonomic functionalities in communications networks. *CoRR*, 2012.

[6] S. Hämäläinen, H. Sanneck, and C. Sartori. *LTE Self-Organising Networks: Network Management Automation for Operational Efficiency*. Wiley, 2011.

[7] O. Iacoboaiea, B. Sayrac, S. Ben Jemaa, and P. Bianchi. SON coordination for parameter conflict resolution: A reinforcement learning framework. In *WCNC'14 - SONET Workshop*, Apr. 2014.

[8] Z. Liu, P. Hong, K. Xue, and M. Peng. Conflict avoidance between mobility robustness optimization and mobility load balancing. In *GLOBECOM*, 2010.

[9] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan. Coordinating handover parameter optimization and load balancing in LTE self-optimizing networks. In *VTC Spring)*, 2011.

[10] L. Schmelz, M. Amirijoo, A. Eisenblaetter, R. Litjens, M. Neuland, and J. Turk. A coordination framework for self-organisation in LTE networks. In *ISINM*, 2011.

[11] S. Sesia, I. Toufik, and M. Baker. *LTE - The UMTS Long Term Evolution: From Theory to Practice 2nd Edition*. Wiley, 2011.

[12] R. Sutton and A. Barto. *Reinforcement Learning: an introduction*. A Bradford book. 1998.

APPENDIX

A. PROOF OF PROPOSITION 1

PROOF. To simplify (6) we fist calculate ($\forall t \in \mathbb{N}$):

$$\begin{aligned}
&\mathbb{P}\left(U_{t+1}=u'|S_0=s,A_0=a,P_1=p'\right)\\
&=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u',U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\
&=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u''),S_0=s,A_0=a\right)\cdot\\
&\qquad\qquad\mathbb{P}\left(U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\
&\overset{(*)}{=}\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u'')\right)\cdot\\
&\qquad\qquad\mathbb{P}\left(U_1=u''|P_1=p'\right)\\
&=\mathbb{P}\left(U_{t+1}=u'|P_1=p'\right)
\end{aligned}\tag{18}$$

where $(*)$ comes from using the Markov property (on the first term) and Assumption 2 (on the second term).

Define $\rho_c : \mathcal{U}^{NK} \to \mathbb{R}$ where $\rho_c(u) = \sum_{n\in\mathcal{N}} \rho\left(u_{n,\bar{\mathcal{K}},\mathcal{Z}}\right)$. From (6) we have:

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t R_{t+1}|S_0=s,A_0=a\right]\\
&= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t \rho_c(U_{t+1})|S_0=s,A_0=a\right]\\
&\overset{A1}{=} \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t \rho_c(U_{t+1})\,|P_1=g(s,a),\right.\\
&\qquad\qquad\left.S_0=s,A_0=a\right]\\
&\overset{(18)}{=} \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t \rho_c(U_{t+1})\,|P_1=g(s,a)\right]\\
&= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^\infty \gamma^t \sum_{n\in\mathcal{N}}\rho\left(U_{t+1,n,\bar{\mathcal{K}},\mathcal{Z}}\right)|P_1=p\right]\\
&= W^\pi(p) = \textstyle\sum_{n\in\mathcal{N}} W_n^\pi(p)
\end{aligned}\tag{19}$$

where $W_n^\pi(p) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t \rho\left(U_{t+1,n,\bar{\mathcal{K}},\mathcal{Z}}\right)|P_1=p\right]$, $\forall n \in \mathcal{N}$, for $p = g(s,a)$. We can further develop $W_n^\pi(p)$, $\forall n \in \mathcal{N}$, $\forall p \in \mathcal{P}^N$, as follows:

$$\begin{aligned}
W_n^\pi(p) &= \mathbb{E}\left[\rho\left(U_{1,n,\bar{\mathcal{K}},\mathcal{Z}}\right)|P_1=p\right]+\\
&\quad\mathbb{E}_\pi\left[\textstyle\sum_{t=1}^\infty \gamma^t \rho\left(U_{t+1,n,\bar{\mathcal{K}},\mathcal{Z}}\right)|P_1=p\right]\\
&\overset{A2}{=} \overline{r}_n\left(p_{\mathcal{N}_n,\mathcal{K}}\right)+\gamma\mathbb{E}_\pi\left[W_n^\pi(P_2)\,|P_1=p\right].
\end{aligned}\tag{20}$$

The conclusion of Proposition 1 for a policy π follows simply by detailing $\mathbb{E}_\pi$. □