Coordinating SON Instances: Reinforcement Learning with Distributed Value Function

Ovidiu 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*—With the emergence of Self-Organizing Network (SON) functions network operators are faced with a practical problem: coordination of SON instances. The SON functions are usually designed in a standalone manner, i.e. they do not take into account the possibility that other instances of the same or different SON functions may be running in the network. This creates the risk of conflicts and network instability. Therefore a SON COordinator (SONCO) is needed. In this paper we design an operator centric SONCO that sees the SON instances as black-boxes, i.e. it does not know the algorithm inside the SON functions. Our aim is to improve the network stability (i.e. number of parameter changes) for SON instances of the same SON function. We employ Reinforcement Learning (RL) in order to profit from the information on the past SONCO decisions. We simplify the expression of the action-value function and we use state aggregation to further reduce the required state space, making it scale linearly with the number of coordinated cells. We provide a study case with the Mobility Load Balancing (MLB) function independently instantiated on every cell. The results show that the proposed SONCO improves the network stability.**

***Index Terms*—SON; Coordination; SON instances; LTE; MLB; reinforcement learning; state aggregation**

I. Introduction

Release 8 of 3GPP has introduced the Self Organizing Networks (SONs) as a means to reduce the operators CAPital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX). Self-Optimizing Networks constitute one category of SON functions that perform run-time optimization of the network, like for example Mobility Load Balancing (MLB), Mobility Robustness Optimization (MRO), etc ([1]). In the sequel SON will refer particularly to this category.

A SON instance is a realization of a SON function that governs (optimizes) one or a set of cells. Typically SON functions are built in a standalone manner, i.e. they do not take into account the existence of other SON instances in the network. So coordination is not implicit especially in a multi-vendor network and therefore conflicts and network instabilities are likely to occur. To deal with this issue Release 10 of 3GPP has introduced the SON COordinator (SONCO) [1]. We distinguish 2 dimensions of the coordination problem:

- the instance dimension: coordination between instances of the same SON function (e.g. MLB), most-likely to be needed if the instances come from different vendors,
- the function dimension: coordination between instances of different SON functions (e.g. MLB, MRO etc.)

In the literature the coordination problem is addressed in several papers. There are mainly two tracks. The first one assumes that the SON instances are black boxes, i.e. the algorithm inside is unknown. Most of the current work on this tack does not profit from the information on the past experience: in [2] and [3] the coordination is done by attributing priorities to the SON functions; in [4] and [5] the authors use decision trees; [6] focuses on conflict avoidance through creating restrictions for the value set of the parameters; in [7], the authors propose a per user optimization that attributes priorities to users for handovers (HOs) based on the weights attributed to the SON functions. So far, exploiting the information on the outcome of past decisions has only been used in [8], employing Reinforcement Learning (RL) for parameter conflict resolution. The second track considers the SON functions as white boxes (i.e. the algorithm inside is known), e.g. [9], but this is not always a plausible approach for an operator-centric SONCO.

In this paper we follow the first track and we make use of the past experience by employing RL to design the SONCO but, unlike in [8], we focus on network stability and on improving the scalability of the value-function. We coordinate instances of the same SON function running on neighboring cells. The contributions of this paper can be summarized as follows:

- we design an operator-centric SONCO considering the SON instances as black-boxes,
- we target the network stability for independent SON instances of the same SON function running on each cell,
- we use a RL-based SONCO solution to find the best parameter configuration, eliminate changes that divert us from this configuration and still explore occasionally other potentially good configurations.
- we simplify the expression of the value-action function reducing the state space.
- we also perform a state-aggregation, to get the required state-space to scale linearly with the number of cells.
- we provide a case study with MLB instances.

The remainder of this paper is structured as follows: Section II provides the system description presenting the scenario, the SON instances and SONCO ; Section III outlines the Markov

DOI: 10.1109/PIMRC.2014.7136431

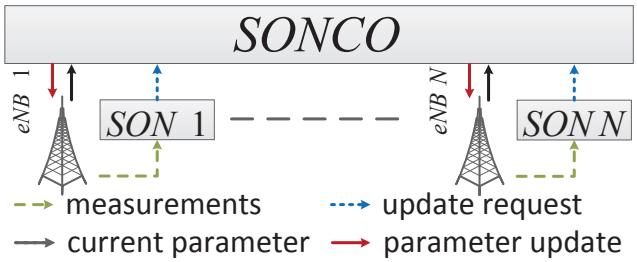

Figure 1. Functional Block Diagram: SONCO↔SON (MLB) interactions

Decision Process (MDP) underlying the RL solution; Section IV presents the RL algorithm; simulation results are included in Section V and Section VI concludes the paper.

II. System Description

To simplify notations we consider a variable X indexed by a set $\mathcal{I}$ to have the following meaning: $X_{\mathcal{I}} = (X_i)_{i\in\mathcal{I}}$.

We consider a network sector composed of N cells. On each cell $n \in \mathcal{N} = \{1, ..., N\}$ we have only one independent SON instance (or even a group of SON instances tuning different parameters, that can be seen as one merged SON instance) indexed also by n (see Fig. 1). Assume that on each cell there are K tuned parameters (e.g. Cell Individual Offset (CIO), antenna tilt, etc.) indexed by $k \in \mathcal{K} = \{1, ..., K\}$.

Let $P_{t,n}$ be a K-tuple representing the parameter configuration on cell n at time t and $\mathcal{P}$ be the set of possible values of $P_{t,n}$, $\forall\,(t, n) \in \mathbb{N} \times \mathcal{N}$. Denote $P_t = P_{t,\mathcal{N}}$.

Each SON instance sends an update request targeting to modify the parameters $\mathcal{K}$ of the cell it runs on. The update requests consist of a K-tuple $u \in \mathcal{U} = \{\pm 1, 0\}^K$ where $(\forall k \in \mathcal{K})$ $u_k = -1$, $u_k = 0$ and $u_k = 1$ means decrease, maintain and increase the value of parameter k, respectively. Let $U_{t,n}$ be the update requests sent at time t by the SON instance running on cell n. Denote $U_t = U_{t,\mathcal{N}}$.

Having several independent SON instances in the network creates the risk for instability, i.e. unnecessary parameter changes due to the interdependence of the SON instances. This raises the need for a SONCO meant to deal with this issue. Therefore the SON instances do not directly make the desired parameter changes in the network, instead they send update requests to the SONCO. The SONCO has to decide which requests to accept and which to deny, eliminating as many as possible of the unnecessary parameter changes.

We assume that all SON instances are synchronized and they all send the update requests to the SONCO at the end of a periodic time window of size T. The SONCO is synchronized with the SON instances; after receiving all the requests it takes its decision and ensures the accepted parameter changes are executed immediately.

The purpose of the SONCO instances is to find the most rewarding configuration (the reward is defined in the sequel), to steer the network towards this configuration and to prevent as much as possible getting diverted from it. To do so we use RL trough which we attribute value functions to network configurations. The value functions are learned online using the Temporal Difference (TD) learning method [10]. The underlying Markov Decision Process (MDP) is describe in the next section.

III. Markov Decision Processes

A. General framework

We present how our problem maps to an MDP:

- **State space** $\mathcal{S} = \mathcal{P}^N \times \mathcal{U}^N$. A state $s \in \mathcal{S}$ is composed of 2 NK-tuples $s = (p, u)$: the current parameter values and update requests .
- **Action space** $\mathcal{A} = \mathcal{A}_2^{NK}$, $\mathcal{A}_2 = \{0, 1\}$. An action $a \in \mathcal{A}$ is a NK-tuple of binary variables. (1=accept, 0=deny)
- **Transition kernel**. $\mathcal{T}(s'|s, a)$ the probability of going to state s' when the current state-action pair is (s, a).
- **Reward.** $r(s, a)$ is the reward 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 $\mathbb{P}_\pi$. Namely, for $\mathbb{P} = \mathbb{P}_\pi$:

$$\mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a) = \mathcal{T}(s'|s, a), \tag{1}$$

$$\mathbb{P}(A_t = a|S_t = s) = \pi(s, a). \tag{2}$$

B. Assumptions

The transition kernel has some particular characteristics. The future parameter configuration on cell $n \in \mathcal{N}$ $(P_{t+1,n})$ is a deterministic function of: the current value of parameters on cell n $(P_{t,n})$ together with the corresponding update requests $(U_{t,n})$ and action $(A_{t,n})$. Moreover the current update requests of the SON instance on cell n $(U_{t,n})$ depend only on the current parameter configurations $(P_{t,\mathcal{N}})$ of the network.

Assumption 1 (kernel)**.**

1) *There exists the deterministic functions* $g : \mathcal{P} \times \mathcal{U} \times \mathcal{A}_2^K \to \mathcal{P}$, *s.t.* $P_{t+1,n} = g(P_{t,n}, U_{t,n}, A_{t,n})$, $\forall n \in \mathcal{N}$.
2) $\forall u' \in \mathcal{U}^N$, $\forall p' \in \mathcal{P}^N$, $\forall n \in \mathcal{N}$, $\forall\,(s, a) \in \mathcal{S} \times \mathcal{A}$, *we have that* $\mathbb{P}(U_{t+1,n} = u'_n|S_t = s, A_t = a, P_{t+1} = p') = \mathbb{P}(U_{t+1,n} = u'_n|P_{t+1} = p')$.

Given some state-action pair the reward is a function of the *happiness* of the SON instances reflected in the succeeding update requests. We define the *instantaneous* regret at time t as $R_t = \sum_{n\in\mathcal{N}} R_{t,n}$ where $R_{t,n} = \rho(U_{t,n})$ for some function $\rho : \mathcal{U} \to \mathbb{R}$.

Assumption 2 (reward)**.** $r(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$.

As an example, we now provide a specific form of ρ, relevant for the SON coordination. For any $u \in \mathcal{U}$:

$$\rho(u) = \max_{k\in\mathcal{K}} \sum_{i\in\{\pm1,0\}} r_i \mathbb{I}_{\{u_k=i\}} \tag{3}$$

where $(r_i)_{i\in\{\pm1,0\}} \in [0,1]$ (the closer to zero the more the probability of receiving the corresponding request is reduced) and $\mathbb{I}_{\{.\}}$ is the indicator function ($\mathbb{I}_{\{true\}}=1$, $\mathbb{I}_{\{false\}}=0$). The use of the max function over $k \in \mathcal{K}$ targets to improve the *fairness* w.r.t. the parameters.

C. Optimal policy

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

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\,|S_0=s\right]\ , \tag{4}$$

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\,|S_0=s, A_0=a\right]. \tag{5}$$

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

The following proposition simplifies (5). We first define:

- $\forall n\in\mathcal{N}$, $\bar{r}_n : \mathcal{P}^N \to \mathbb{R}$ where, $\forall p\in\mathcal{P}^N$, $\bar{r}_n(p) = \mathbb{E}\left[\rho\left(U_{1,n}\right)|P_1=p\right]$,
- $g : \mathcal{S}\times\mathcal{A}\to\mathcal{P}^N$ where $\forall(p,u)\in\mathcal{S}$, $\forall a\in\mathcal{A}$, $\bar{g}\left((p,u),a\right) = \left(g\left(p_n,u_n,a_n\right)\right)_{n\in\mathcal{N}}$,
- the cumulated reward: $\rho_c(u) = \sum_{n\in\mathcal{N}}\rho(u_n)$, $\forall u\in\mathcal{U}^N$.

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 $(s,a)\in\mathcal{S}\times\mathcal{A}$, $Q^\pi(s,a) = \sum_{n\in\mathcal{N}} W_n^\pi\left(\bar{g}(s,a)\right) = W^\pi\left(\bar{g}(s,a)\right)$. Moreover, $W_{\mathcal{N}}^\pi$ solves the following fixed point equation:*

$$\begin{aligned} W_n^\pi(p) = \bar{r}_n(p) + \gamma \sum_{u\in\mathcal{U}^N}\mathbb{P}\left[U_1=u|P_1=p\right]\cdot \\ \sum_{a\in\mathcal{A}}\pi\left((p,u),a\right)\cdot W_n^\pi\left(\bar{g}\left((p,u),a\right)\right),\ \forall n\in\mathcal{N} \end{aligned} \tag{6}$$

Proof: To simplify (5) we fist calculate ($\forall t\in\mathbb{N}$) :

$$\begin{aligned} &\mathbb{P}\left(U_{t+1}=u'|S_0=s,A_0=a,P_1=p'\right)\\ &=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u',U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\ &=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u''),S_0=s,A_0=a\right)\cdot\\ &\qquad\mathbb{P}\left(U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\ &\overset{(*)}{=}\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u'')\right)\cdot\\ &\qquad\mathbb{P}\left(U_1=u''|P_1=p'\right)\\ &=\mathbb{P}\left(U_{t+1}=u'|P_1=p'\right) \end{aligned} \tag{7}$$

where $(*)$ comes from using the Markov property (on the first term) and Assumption 1.2 (on the second term).

Now, from (5) we have:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^{\infty}\gamma^t R_{t+1}|S_0=s,A_0=a\right]\\ &= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^{\infty}\gamma^t\rho_c\left(U_{t+1}\right)|S_0=s,A_0=a\right]\\ &\overset{A1.1}{=} \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^{\infty}\gamma^t\rho_c\left(U_{t+1}\right)|P_1=\bar{g}(s,a),\right.\\ &\qquad\left.S_0=s,A_0=a\right]\\ &\overset{(7)}{=} \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^{\infty}\gamma^t\rho_c\left(U_{t+1}\right)|P_1=\bar{g}(s,a)\right]\\ &= \mathbb{E}_\pi\left[\textstyle\sum_{t=0}^{\infty}\gamma^t\sum_{n\in\mathcal{N}}\rho\left(U_{t+1,n}\right)|P_1=p\right]\\ &= W^\pi(p) = \textstyle\sum_{n\in\mathcal{N}} W_n^\pi(p) \end{aligned} \tag{8}$$

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

$$\begin{aligned} W_n^\pi(p) &= \mathbb{E}\left[\rho\left(U_{1,n}\right)|P_1=p\right] +\\ &\qquad\mathbb{E}_\pi\left[\textstyle\sum_{t=1}^{\infty}\gamma^t\rho\left(U_{t+1,n}\right)|P_1=p\right]\\ &= \bar{r}_n(p) + \gamma\mathbb{E}_\pi\left[W_n^\pi\left(P_2\right)|P_1=p\right]. \end{aligned} \tag{9}$$

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

Remark 1 (optimal policy)**.** *If a policy π^* minimizes the value function $\forall s\in\mathcal{S}$ then it is said to be* optimal *[10]. 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 [10]): $\pi^*(s) = arg\ max_a Q^*(s,a)$ ($Q^*$is the* optimal action-value function*). Thus eq. (6) for the optimal policy can be recovered by:*

$$\begin{aligned} &W_n^*(p) = \bar{r}_n(p) + \gamma\textstyle\sum_{u\in\mathcal{U}^{NZ}}\mathbb{P}\left[U_1=u|P_1=p\right]\cdot\\ &\qquad\qquad W_n^*\left(p^*\right),\ \forall n\in\mathcal{N},\\ &p^* = \bar{g}\left((p,u),a^*\right),\\ &a^* = \pi^*(p,u) = \arg\max_{a\in\mathcal{A}} W^*\left(\bar{g}\left((p,u),a\right)\right) \end{aligned} \tag{10}$$

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 $W^*(\sum_{n\in\mathcal{N}} W_n^*)$.

D. State aggregation. Sub-optimal policy.

Although Proposition 1 allows to simplify (5) to a function with a reduced domain $\mathcal{P}^N$ (reducing thus the required state-space) it still scales exponentially with the number of cells N. Therefore in the sequel we set to perform a state-aggregation, at the cost of possible performance loss. Note that W_n^π depends on p mainly through $p_{\mathcal{N}_n}$ (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 on $\mathcal{P}^N\to\mathbb{R}$ that depend on $p\in\mathcal{P}^N$ only through $p_{\mathcal{N}_n}$, namely:

$$\mathcal{W}_n = \left\{p\mapsto F\left(p_{\mathcal{N}_n}\right) : F\in\mathbb{R}^{\mathcal{Y}_n}\right\} \tag{11}$$

where $\mathcal{Y}_n = \mathcal{P}^{N_n}$; therefore instead of directly computing $W_{\mathcal{N}}^*$ as the solution to (10) we aim to evaluate its projection onto $\mathcal{W}=\mathcal{W}_{\mathcal{N}}$ ($\equiv$ an approximation of $W_{\mathcal{N}}^*$) of the form:

$$f\left(p_{\mathcal{N}}\right) = \textstyle\sum_{n\in\mathcal{N}} f_n\left(p_{\mathcal{N}_n}\right) \tag{12}$$

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

Note that if for each and every cell we consider that all the other cells are neighbors, i.e. $\mathcal{N}_n=\mathcal{N}$, $\forall n\in\mathcal{N}$, than the state aggregation does not have any effect. Thus in this case $f_{\mathcal{N}}\equiv W_{\mathcal{N}}$, and we can evaluate the optimal policy like this.

IV. Reinforcement learning

A. Algorithm

We cannot directly calculate f as we only have partial knowledge on the transition kernel, instead we use the flowing recursion:

$$\begin{aligned} &\forall n\in\mathcal{N},\ f_{t+1,n}\left(P_{t,\mathcal{N}_n}\right) = (1-\alpha)\,f_{t,n}\left(P_{t,\mathcal{N}_n}\right) +\\ &\qquad\qquad\alpha\left(R_{t,n}+\gamma f_{t,n}\left(\bar{P}_{t+1,\mathcal{N}_n}\right)\right)\\ &\bar{P}_{t+1} = \bar{g}\left(\left(P_{t,},U_t\right),\bar{A}_t\right)\\ &\bar{A}_t = \arg\max_{a\in\mathcal{A}} f_t\left(\bar{g}\left(\left(P_t,U_t\right),a\right)\right) \end{aligned} \tag{13}$$

where $\bar{A}_t$ represents the optimal action based on f_t.

We use an ϵ-greedy policy:

$$\pi_t\left(S_t,\{a\}\right) = \left((1-\epsilon)\,\mathbb{I}_{\left\{a=\bar{A}_t\right\}} + \frac{\epsilon}{2^{NK}}\right) \tag{14}$$

Algorithm 1 SONCO

Function Init :

For all $n \in \mathcal{N}$, initialize $f_n(p') = 0, \forall p' \in \mathcal{P}^{N_n}$

Function SONCO :

Observe current parameter configurations p and update requests u, compute regret $r_n = \rho(u_n), \forall n \in \mathcal{N}$

Calculate $\bar{a} = \arg\max_{\check{a}} f(\bar{g}((p,u),\check{a}))$ and $\bar{p} = \bar{g}((p,u),\bar{a})$

For all $n \in \mathcal{N}$

$f_n(p_{\mathcal{N}_n}) \leftarrow (1-\alpha) f_n(p_{\mathcal{N}_n}) + \alpha(r_n + \gamma f_n(\bar{p}_{\mathcal{N}_n}))$

Choose action a using an ϵ-greedy policy , Take action a.

Table I
SIMULATION PARAMETERS

Category	Parameter	Value
Network	Inter Site Distance	500 m
Channel modeling	Carrier frequency/ Bandwidth	2 GHz /10 MHz
	cell TX Power	46 dBm
	Propagation Model	3GPP Case 1 [11]
	Channel Model	MIMO 2×2
	Radio Resource Control	3GPP [12]
SON	CIO values ($\mathcal{C}$ [dB])	$\{-9,-6,-3,0\}$
	Time window T	2.5 min
	$(r_0; r_1; r_{-1})$	$(1; 0.9; 0)$
	$(\mathbb{T}_L; \mathbb{T}_H)$	$(0.5; 0.8)$
SONCO	Learning rate α	0.05
	Discount factor γ	0.8
	Epsilon greedy parameter ϵ	0.1

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* $\mathcal{W}$ [10].

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.

B. Complexity analysis

The optimal policy is usually obtained through Q-Learning [10] with one SONCO that governs all cells. According to the previous section the optimal policy can also be obtained by learning W^* in (6). This allows us to reduce the required state (and action) space from a size of: $\Psi_V = |\mathcal{S}| \cdot |\mathcal{A}| = |\mathcal{P}|^N |\mathcal{U}|^N \cdot |\mathcal{A}_2|^{NK}$ to a size of $\Psi_W = N|\mathcal{P}|^N$. Still, this solution scales exponentially with N, but after we perform the state aggregation coming to a state space size of $\Psi_f = \sum_{n\in\mathcal{N}} |\mathcal{P}|^{N_n}$, and one can see that this scales linearly with the number of cells.

V. Simulation Results

A. Simulation scenario

To demonstrate the concept we use the MLB function described in this sub-section. The MLB instances tune a mobility parameter: the Cell Individual Offset (CIO), for optimizing the cell load [13]. The users that wish to transmit data get attached to cell $n_0 = \text{argmax}_{n\in\mathcal{N}}(RSRP_n + C_n)$ where $RSRP_n$ is the Reference Signal Received Power ([14]) from cell n and

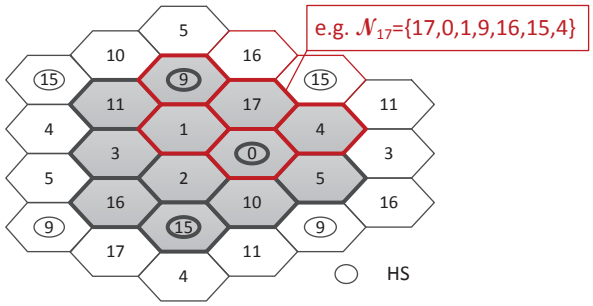

Figure 2. Network topology

C_n is the CIO of cell n. The input of an MLB instance is the cell load ($L_{t,n}$) calculated as the average number of occupied Physical Resource Blocks (PRBs, [15]) of the hosting cell (the cell on which the MLB instance runs) during the time interval $(t-1, t]$. The output is a request to change the CIO of the hosting cell ($P_n \leftarrow C_n$): $U_{t,n} \in \{\pm 1, 0\}$. For simulation purposes we use the following algorithm as a typical MLB implementation:

$$U_{t,n} = \begin{cases} -1 & \text{, if } L_{t,n} > \mathbb{T}_H \\ 1 & \text{, if } L_{t,n} < \mathbb{T}_L \\ 0 & \text{, otherwise} \end{cases} \quad (15)$$

where $\mathbb{T}_H$ and $\mathbb{T}_L$ are fixed thresholds used by the MLB instance to trigger CIO update requests ($\mathbb{T}_H > \mathbb{T}_L$). The MLB instance sends: *off-load* requests ($U_{t,n} = -1$) when the cell is overloaded and *on-load* requests ($U_{t,n} = 1$) to get back to the default configuration when the traffic is no longer imbalanced (and thus no need to balance the cell loads).

The reward is calculated using (3). The coefficients $((r_i)_{i\in\{\pm1,0\}})$ are fixed such that we favor the configurations where we do not receive *off-load* requests. Simulation details are summarized in Table I.

We use a hexagonal topology with $N = 12$ and wraparound (see Fig. 2); 4 different cases are analyzed:

- SFoff: MLB instances are deactivated,
- SCoff: MLB instances are activated, SONCO is off,
- SCOP: Optimal policy: $\forall n \in \mathcal{N}$, $\mathcal{N}_n = \mathcal{N}$, i.e. each cell considers all other cells as neighbors.
- SCSP: Sub-optimal policy (with state aggregation): $\forall n \in \mathcal{N}$, $\mathcal{N}_n$ includes cell $\{n\}$ and its 6 direct neighbors.

We consider a general background traffic arrival rate η_G [Mb/s] together with an additional hotspot (denoted by HS in Fig. 2) arrival rate η_{HS} [Mb/s] so that the resulting HS arrival rate is η_G+η_{HS}. The user arrival rates per area unit can be easily obtained as: $\rho_{(\cdot)}[UE/s/m^2] = \eta_{(\cdot)}[Mb/s]/S_{(\cdot)}[m^2]/FS[Mb/UE]$ (S refers to the area and FS to the 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. User scheduling is done in a Proportional Fair (PF) manner.

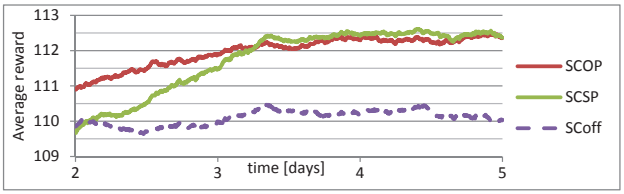

Figure 3. Average reward ($48h$ sliding window)

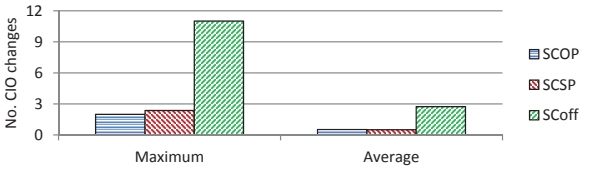

Figure 4. Average no. of CIO changes ([#/h]) over last 48h

B. Simulation results

We use a file size of $FS = 16[Mb/UE]$ and the traffic arrival rates $\eta_G = 72[Mb/s]$ and $\eta_{HS} = 63[Mb/s]$. We analyze data from 5 simulated days.

Fig. 3 shows that the average reward is increased when the SONCO is on. Moreover the sub-optimal policy (SCSP) converges to the same reward as the optimal policy (SCOP). One can see that the convergence of the algorithm remains slow. This is due to the fact that the algorithm learns from scratch and needs to visit all the possible states a sufficient number of times. Once the learning algorithm is trained offline on a realistic network simulator, it can be implemented in the real network. The algorithm should then be capable of tracking and adapting dynamically to the traffic variations.

As mentioned our aim is to reduce the number of parameter changes. In Fig. 4 we plot the maximum and the average (over all cells) of the time-averaged number of CIO changes. One can see that he proposed approach has eliminated a big amount of the unnecessary configuration changes (77-82%) and implicitly the accompanying signaling. By steering the network towards the most rewarding CIO configuration we also reduce the frequency of overload events (concretely the number of off-load requests) by 33-40%.

In Fig. 5 the maximum and the average (over all cells) of the time-averaged loads are plotted. We can see that for the SONCO-on cases (SCOP, SCSP) the MLB instances still accomplish their task (keeping the load below $\mathbb{T}_H$) meaning that the SONCO does not prevent the MLBs from balancing the cell loads.

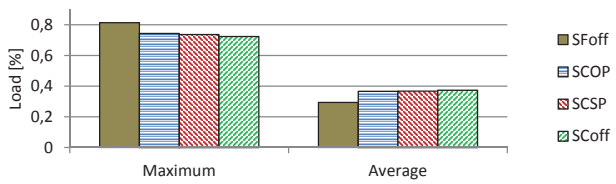

Figure 5. Average loads over last 48h

VI. Conclusions and Future Work

We have proposed an operator-centric SONCO that coordinates independent SON instances seen as black-boxes. Our design is based on RL allowing us to make use of the information on the impact of our past actions. We simplify the expression of the value-action function (smaller state-space) and afterwards we apply a state-aggregation to make the required state-space scale linearly with the number of coordinated cells. We provide a study-case with independent MLB instances and we show that the sub-optimal policy (obtained by state-aggregation) reaches performances close to the optimal policy. We manage to reduce the number of CIO changes by 77-82% while still allowing the MLBs to accomplish their task, i.e. keep the cells from becoming overloaded. Future work will focus on dynamic traffic scenarios in order to evaluate the tracking capabilities of the algorithm.

Acknowledgment

The research leading to these results has been carried out within the FP7 SEMAFOUR project [16] and has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 316384.

References

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

[2] L. Schmelz, M. Amirijoo, A. Eisenblaetter, R. Litjens, M. Neuland, and J. Turk, "A coordination framework for self-organisation in lte networks," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 193–200.

[3] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan, "Coordinating handover parameter optimization and load balancing in lte self-optimizing networks," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, 2011, pp. 1–5.

[4] T. Bandh, H. Sanneck, and R. Romeikat, "An experimental system for son function coordination," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, 2011.

[5] T. Bandh, R. Romeikat, H. Sanneck, and H. Tang, "Policy-based coordination and management of son functions," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011.

[6] Z. Liu, P. Hong, K. Xue, and M. Peng, "Conflict avoidance between mobility robustness optimization and mobility load balancing," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010.

[7] J. Chen, H. Zhuang, B. Andrian, and Y. Li, "Difference-based joint parameter configuration for mro and mlb," in *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, 2012, pp. 1–5.

[8] O. Iacoboaiea, B. Sayrac, S. Ben Jemaa, and P. Bianchi, "SON coordination for parameter conflict resolution: A reinforcement learning framework," in *IEEE WCNC 2014 - Workshop on Self-Organizing Networks (WCNC'14 - SONET Workshop)*, Istanbul, Turkey, Apr. 2014.

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

[10] R. Sutton and A. Barto, *Reinforcement Learning: an introduction*, ser. A Bradford book. A Bradford Book, 1998.

[11] 3GPP, "E-UTRA; Further advancements for E-UTRA physical layer aspects," 3rd Generation Partnership Project (3GPP), TR 36.814, 2010.

[12] ——, "LTE; E-UTRA; Radio Resource Control; Protocol specification," 3rd Generation Partnership Project (3GPP), TS 36.331, 2012.

[13] ——, "LTE; E-UTRA and E-UTRAN; Overall description," 3rd Generation Partnership Project (3GPP), TS 36.300, 2013.

[14] ——, "LTE; E-UTRA; Physical layer; Measurements," 3rd Generation Partnership Project (3GPP), TS 36.214, 2012.

[15] ——, "LTE; E-UTRA; Physical channels and modulation," 3rd Generation Partnership Project (3GPP), TS 36.211, 2013.

[16] SEMAFOUR project web page http://fp7-semafour.eu/.