Low Complexity SON Coordination using Reinforcement Learning

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*—The continuously increasing traffic demand faces us with increased CAPital EXpenditures (CAPEX) and OPerational EXpenditure (OPEX). Self Organizing Network (SON) functions aim to lower these costs by automating the network tuning. A SON instance is a realization of a SON function which can tune one or a set of cells. Having several uncoordinated SON functions in the network creates a risk for conflicts and instability. This raises the need for a SON COordinator (SONCO) meant to deal with these issues. In this work we consider that on each cell we have one SON instance of each SON function. We present the design of a SONCO which arbitrates conflicts based on weights attributed to the SON functions. The design makes use of Reinforcement Learning (RL) with function approximation. We provide a low complexity approximation of the action-value function based on a number of parameters that scales linearly with the number of cells. We present a study case with the Mobility Load Balancing (tuning the Cell Individual Offset (CIO)) and Mobility Robustness Optimization (tuning the CIO and the handover hysteresis) functions, where the SONCO deals with the conflicts on the CIOs. Numerical results prove that we can orchestrate the SON functions through SONCO configurations that reflect different operator policies.**

***Index Terms*—SON Coordination; MLB; MRO; SON instances; LTE; reinforcement learning; function approximation;**

I. Introduction

To respond to the increasing traffic demand operators have to continuously improve the network capabilities. This comes at the cost of increased CAPital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX). In response Release 8 of 3GPP has introduced the Self Organizing Network (SON) functions that automate the network operations. One category of these functions contains the Self Optimizing Network functions (e.g. Mobility Load Balancing (MLB), Mobility Robustness Optimization (MRO), etc.) that perform run-time optimizations of the network parameters; in the sequel SON refers to them.

In a real network, different types of conflicts (see [1]) can arise if we use several SON instances of the same or different SON functions (e.g. MLB, MRO, Coverage and Capacity Optimization (CCO), etc.), especially in a multi-vendor context. This raises the need for a SON COordinator (SONCO) meant to deal with these problems. In this case the SON instances cannot directly change the parameter configuration. Instead they send requests to the SONCO that decides which requests to accept (and immediately put into effect) and which to deny.

SON coordination was introduced in Release 10 of 3GPP and has been getting a lot of attention lately. In the literature, two different approaches for SON coordination are considered. In the first one the SONCO sees the SON instances as black-boxes (i.e. it does not have the information on the SON algorithm and its inputs). Currently in this category we find: techniques that attribute priorities to the SON functions in [2] and [3], decision trees in [4] and [5], parameter restrictions in [6] and a per user prioritization for handover (HO) in [7]. So far, all these techniques have not taken advantage of the knowledge on the impact of their past decisions. The other approach is to consider the SON instances as white boxes (i.e. the algorithm and input parameters are known,e.g. [8]), which is not always the case from an operator point of view.

In this paper we offer a solution based on the first approach. We build an operator-centric SONCO and we use Reinforcement Learning (RL) to benefit from the knowledge on the effect of our past decisions. RL has been widely used for SON algorithm design (e.g. [9]). Continuing the work in [10] and [11], we provide a solution based on a linear function approximation of the action-value function where the number of parameters scales linearly with the number of cells. As in our previous work:

- our SONCO design is operator-centric, considering the SON instances as black-boxes,
- the requests of the SON instances indicate also how unhappy they are with the current parameter configuration,
- we provide a solution based on RL, where we first simplify the complexity of the optimal policy,
- we provide a study case with 2 SON functions (MLB and MRO) instantiated on each and every cell.

and moreover in this paper:

- for simplifying the complexity of the optimal policy we consider the regret components to be established per update-request (instead of per cell as used in [11]) in order to better approximate the value function,
- we then use a linear function approximation to get a low complexity distributed approximation of the value function, employing a small number of parameters that

DOI: 10.1109/GLOCOM.2014.7037501

scales linearly with the number of cells,
- we use a higher resolution of the network parameters.

The rest of this paper is organized as follows: Section II provides the system description presenting the scenario. Section III outlines the Markov Decision Process (MDP) underlying the RL-solution together with the function approximation and Section IV introduces the RL algorithm with a complexity analysis. Simulation results are included in Section V and Section VI concludes the paper.

II. System Description

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

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 $N_n = |\mathcal{N}_n|$, $\forall n \in \mathcal{N}$.

Network optimization is done using 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.

On each and every cell there are K parameters (e.g. CIO, Hysteresis, antenna tilt, etc.), indexed by $k \in \mathcal{K} = \{1, .., K\}$, that are tuned by the instances of the SON functions. Some parameters, say the parameters $\{1, .., \bar{K}\}$ represent an important source of conflicts. We refer the reader to [1, 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 $P_t = P_{t,\mathcal{N},\mathcal{K}}$, $\mathcal{P} = \prod_{k\in\mathcal{K}}\mathcal{P}_k$ 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 $u \in (0; 1]$ are 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 $\bar{\mathcal{U}} = \prod_{k\in\bar{\mathcal{K}}}\mathcal{U}_k$.

III. SON Coordination.

Instead of directly executing the desired parameter changes in the network, the SON instances send update requests to a SONCO which decides if they are accepted (and immediately executed) or denied. The SON instances are seen as black-boxes by the operator-centric SONCO, so it does not know the algorithm running inside them. It only knows the current update requests (U_t) and the current parameter configuration of the network (P_t). The task of the SONCO is to find a equitable conflict resolution.

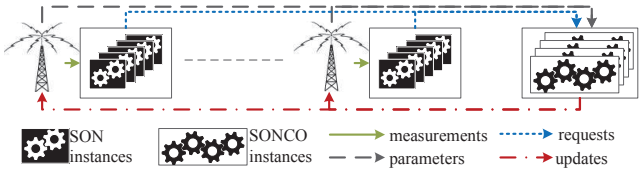

Figure 1. Functional Block Diagram: SONCO↔SON interactions

The SON instances are considered to be synchronized. They send the update requests simultaneously at the end of a periodic time window of size T (Fig. 1). The SONCO is also synchronized with the SON instances. It receives the update requests from all the SON instances and decides which requests are accepted and which are denied. We employ a RL approach as it enables us to use the information on the effect of our past decisions. This information is stored in value functions which reflect how satisfied we were with a given configuration. RL is founded on Markov Decision Process (MDP) as described in the sequel.

A. 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:

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

  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:

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

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

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.

B. Assumptions

Our transition kernel has some particular characteristics. The future parameter 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 exist some deterministic functions $g_k : \mathcal{P}_k \times \mathcal{U}_k^{\mathcal{Z}} \times \mathcal{A}_3 \to \mathcal{P}_k$, $\forall k \in \mathcal{K}$, s.t. $P_{t+1,n,k} = g_k\left(P_{t,n,k}, U_{t,n,k,\mathcal{Z}}, A_{t,n,k}\right)$, $\forall n \in \mathcal{N}$, $\forall k \in \mathcal{K}$.*

Further more 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 of 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 $\bar{\mathcal{K}}$. To be more specific the regret is the sum of local-regrets (per cell, per parameter and per SON function). We denote $\mathcal{M} = \mathcal{N}\times\mathcal{K}\times\mathcal{Z}$ and $\bar{\mathcal{M}} = \mathcal{N}\times\bar{\mathcal{K}}\times\mathcal{Z}$, and we introduce the notation $U_{t,n,k,z} = U_{t,m}$, $\forall t \in \mathbb{N}$, $\forall m = (n,k,z) \in \mathcal{M}$.

Assumption 3 (regret)**.** *There exists $\rho : [-1;1] \to \mathbb{R}$ such that $r(s,a) = \sum_{m\in\bar{\mathcal{M}}} r_m(s,a)$ where $r_m(s,a) = \mathbb{E}\left[\rho\left(U_{t+1,m}\right) | S_t = s, A_t = a\right]$, $\forall m \in \bar{\mathcal{M}}$. One can see this as the sum of local-regrets per triplet of cell, parameters and SON function (i.e. per update request $U_{t+1,m}$).*

We now provide a specific form of ρ, relevant for the SON coordination:

$$\rho\left(x\right) = |x|\,, \tag{4}$$

as the *unhappiness* of the SON functions is reflected in the absolute value of the update requests.

Finally we introduce a regret process: $(R_{t,m})_{(t,m)\in\mathbb{N}\times\bar{\mathcal{M}}}$ s.t. $R_{t+1,m} = \rho\left(U_{t+1,m}\right)$ and thus $\mathbb{E}\left[R_{t+1,m} | S_{0:t}, A_{0:t}\right] = r_m\left(S_t, A_t\right)$. Denote $R_t = \sum_{m\in\bar{\mathcal{M}}} R_{t,m}$, $\forall t \in \mathbb{N}$.

C. Value functions

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

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

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

where $0 \le \gamma < 1$ 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). We first define:

- $\forall m = (n,k,z) \in \bar{\mathcal{M}}$, $\overline{r}_m : \mathcal{P}^{N_n} \to \mathbb{R}$ with $\overline{r}_m(p) = \mathbb{E}\left[\rho\left(U_{1,m}\right) | P_{1,\mathcal{N}_n,\mathcal{K}} = p\right]$, $\forall p \in \mathcal{P}^{N_n}$,
- $\rho_s(u) = \sum_{m\in\bar{\mathcal{M}}} \rho\left(u_m\right)$, $\forall u \in \bar{\mathcal{U}}^{NZ}$, is the instantaneous regret calculated as the summation of the local-regrets
- $g : \mathcal{S}\times\mathcal{A} \to \mathcal{P}^N$ 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 $W_m^\pi : \mathcal{P}^N \to \mathbb{R}$, $\forall m = (n,k,z) \in \bar{\mathcal{M}}$, s.t. for any $(s,a) \in \mathcal{S}\times\mathcal{A}$, $Q^\pi(s,a) = \sum_{m\in\bar{\mathcal{M}}} W_m^\pi\left(g\left(s,a\right)\right)$. Moreover, $W_{\bar{\mathcal{M}}}^\pi$ solves the following fixed point equation:*

$$\begin{aligned} W_m^\pi(p) = \overline{r}_m\left(p_{\mathcal{N}_n,\mathcal{K}}\right) + \gamma \sum_{u\in\mathcal{U}^{NZ}} \mathbb{P}\left[U_1 = u | P_1 = p\right] \cdot \\ \sum_{a\in\mathcal{A}} \pi\left((p,u),a\right) \cdot W_m^\pi\left(g\left((p,u),a\right)\right),\ \forall m \in \bar{\mathcal{M}} \end{aligned} \tag{7}$$

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:*

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

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

D. Function approximation. Sub-optimal policy.

Although Prop. 1 allows to simplify (6) to a set of functions with a reduced state-space $\mathcal{P}^N$ the complexity still scales exponentially with the number of cells N. Therefore in the sequel we set to perform a linear function approximation, which comes at the cost of possible performance loss. Note that W_m^π ($m = (n,k,z)$) 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 $\mathcal{W}_m$ ($\forall m \in \bar{\mathcal{M}}$) be the set of functions $W_m^\pi : \mathcal{P}^N \to \mathbb{R}$ such that $\forall p \in \mathcal{P}^N$, $W_m^\pi(p)$ is a linear function of some features $F_{m,i}$ ($i \in \mathcal{I}_m = \{1,..,I_m\}$) of $p_{\mathcal{N}_n,\bar{\mathcal{K}}}$, namely:

$$\mathcal{W}_m = \left\{ p \mapsto \sum_{i\in\mathcal{I}_m} \theta_i F_{m,i}\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) : \theta_{\mathcal{I}_m} \in \mathbb{R}^{I_m} \right\}. \tag{9}$$

Therefore instead of directly computing $W_{\bar{\mathcal{M}}}^*$ as the solution to (8) we aim to evaluate its projection onto $\mathcal{W} = \mathcal{W}_{\bar{\mathcal{M}}}$ (≡

an approximation of $W^*_{\bar{\mathcal{M}}}$) of the form:

$$\begin{aligned} f\left(p_{\mathcal{N},\bar{\mathcal{K}}}\right) &= \sum_{m\in\bar{\mathcal{M}}} f_m\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) \\ &= \sum_{m\in\bar{\mathcal{M}}}\sum_{i\in\mathcal{I}_m}\theta_{m,i}F_{m,i}\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right). \end{aligned} \tag{10}$$

for some $\theta_{m,i} \in \mathbb{R}$, $\forall m \in \bar{\mathcal{M}}$, $\forall i \in \mathcal{I}_m$. As compared to [11], the function approximations significantly reduces the complexity of the estimate of the action-value function.

IV. Reinforcement learning

A. 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 $\forall m = (n,k,z) \in \bar{\mathcal{M}}$:

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

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, we expect f_t to converge 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 function approximation), this allows us to always accept the requests on these parameters:

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

where this formulation is valid given that there are no conflicts on the parameters in $\check{\mathcal{K}}$.

For practical reasons we use an ϵ-greedy policy:

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

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 :
$\forall m = (n,k,z) \in \bar{\mathcal{M}}$, $\forall i \in \mathcal{I}_m$ initialize $\theta_{m,i} = 0$
Function SONCO :
Observe current parameter configurations p and update requests u, calculate regret $r_m = \rho\left(u_m\right)$, $\forall m \in \bar{\mathcal{M}}$
Calculate $\bar{a} = \arg\min_{a\in\mathcal{A}_3^{N\bar{K}}} f\left(\bar{g}\left(\left(p_{\mathcal{N},\bar{\mathcal{K}}},u_{\mathcal{N},\bar{\mathcal{K}},\mathcal{Z}}\right),a\right)\right)$ 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 $m \in \bar{\mathcal{M}}$,
  $\delta_m = r_m + \gamma f_m\left(\bar{p}_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) - f_m\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right)$
  For all $i \in \mathcal{I}_m$,
    $\theta_{m,i} \leftarrow \theta_{m,i} + \alpha\delta_m F_{m,i}\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right)$
Choose action a using an ϵ-greedy policy ,
Take action a.

B. Complexity analysis

The optimal policy is usually obtained through Q-Learning ([12]). According to the previous section the optimal policy can also be obtained by learning W^* in (7). This allows us to substantially reduce the required state (and action) space size from: $\Psi_V = |\mathcal{S}|\cdot|\mathcal{A}| = |\mathcal{P}|^N|\mathcal{U}|^{NZ}\cdot|\mathcal{A}_3|^{NK}$ to $\Psi_W = \left|\bar{\mathcal{M}}\right||\mathcal{P}|^N$. Still, this solution scales exponentially with N, but as mentioned we perform a linear function approximation coming to an expression using $\Psi_Y = \sum_{m\in\bar{\mathcal{M}}} I_m$ parameters, and one can see that this further reduces the complexity (i.e. required memory) and scales linearly with the number of cells.

V. Simulation Results

A. 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 $\bar{\mathcal{K}} = \{1\}$. 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 ([13]) from eNB n and C_n is the CIO of eNB 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 eNB n_S.

We use the MLB and MRO instances described in [11].

For our study case we use the regret in (4). Regarding the features in (9) we consider (for $m = (n,k,z)$): $\mathcal{I}_m = \{1,..,N_n\}$ and $\tau_n : \{1,..,N_n\} \to \mathcal{N}_n$ to be a bijective function (mapping the indexes in $\mathcal{I}_m$ to cell indexes $\mathcal{N}_n$) s.t. $\forall p \in \bar{\mathcal{P}}^N$, $\forall m = (n,k,z) \in \bar{\mathcal{M}}$, $\forall i \in \mathcal{I}_m$:

$$F_{m,i}\left(p_{\mathcal{N}_n,\bar{\mathcal{K}}}\right) = \begin{cases} p_{\tau_n(i),k} - p_{n,k} & \text{if } \tau_n(i) \neq n, \\ 0.1 & \text{if } \tau_n(i) = n. \end{cases} \tag{14}$$

with which we obtain a linear function of the CIO differences between each cell $n \in \mathcal{N}$ and its neighbors (reflecting to some extent the positions of the cell borders).

1) Scenario: The scenario is built on a network segment of $N = 21$ cells with wraparound (Fig. 2). The simulation details are summarized in Table I. 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 η_G+η_{HS}). The user arrivals rates per area unit can be easily obtained as: $\rho_\bullet\left[UE/s/m^2\right] = \eta_\bullet[Mb/s]/S_\bullet\left[m^2\right]/FS[Mb/UE]$ (S refers to the area and

Table I
SIMULATION PARAMETERS

Category	Parameter	Value
Network	Inter Site Distance	1732 m
Channel modeling	Carrier frequency/ Bandwidth	2 GHz/ 10 MHz
	eNB TX Power	46 dBm
	Propagation Model	3GPP Case 3 [14]
	Channel Model	MIMO 2×2
Mobility	HO Time To Trigger	160 ms
	HO Hysteresis ($\mathcal{H}$[dB])	$\{0, 1, ..., 12\}$
	CIO values ($\mathcal{C}$ [dB])	$\{-12, -8, ..., 0\}$
SON instances	Time window T	5 min
	$(\mathbb{T}^L_{ld}; \mathbb{T}^H_{ld})$	$(0.3; 0.8)$
	$(\mathbb{T}^L_L; \mathbb{T}^H_L)$,$(\mathbb{T}^L_P; \mathbb{T}^H_P)$,$\mathbb{T}^H_{L*}$	$(2; 8)$,$(2.5; 10)$,6
	$(\mathbb{T}^L_E; \mathbb{T}^H_E) = (\mathbb{T}^L_W; \mathbb{T}^H_W)$	$(\infty; \infty) \equiv$ off
	$(g_1; m_1)$, $(g_2; m_2)$	$(30; 0.9)$, $(5; -0.2)$
	$(g_3; m_3)$	$(5; 1.5)$
SONCO	$(\alpha; \gamma; \epsilon)$	$(0.1; 0.8; 0.1)$

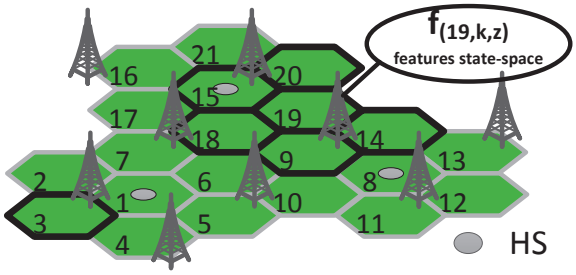

Figure 2. Network topology

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 in cells 1, 8 and 15 as shown in Fig. 2.

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

B. 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.

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, in Figures 3, 4 and 5 respectively. We do not look at the number of too early HOs and the number of HOs to wrong cells because the number of these rare events is not significant.

Fig. 3 shows that assigning a bigger weight to the MLB function makes the load balancing work better, reducing the maximum average load (bellow $\mathbb{T}^H_{ld}$). In our scenario what happens is that the cells with the traffic HSs are allowed to offload more on their neighbors. This off-loading generates more too late HOs as the HO borders are not anymore within

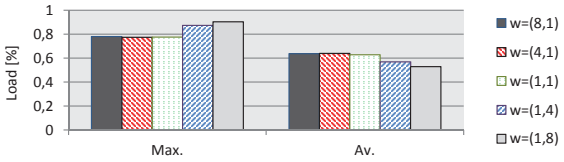

Figure 3. Average Load

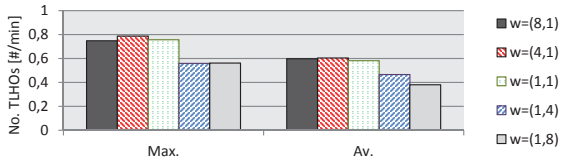

Figure 4. Average Number of Too Late HOs $[\#/\text{min}]$

the actual coverage borders (Fig. 4). An increased number of too late HOs complicates the job of the MRO instances and thus the number of ping pongs also increases (Fig. 5).

On the other hand giving a bigger weight to the MRO prioritizes the reduction of the number of too late HOs (up to 25% reduction in Fig. 4). As expected the price to pay is in terms of load balancing (Fig. 3) where the maximum average load increases (up to 15%). The number of ping pongs is also impacted by the weights, but as it is not directly part of the update request sent by the MRO instances on the CIO we cannot see a trend, but the max is clearly lower than $\mathbb{T}^H_P/T = 2\,[\#/min]$ for any weight.

VI. CONCLUSIONS AND FUTURE WORK

We have presented a RL solution for conflict resolution, where a linear function approximation allows to have distributed value function based on a number of parameters that scales linearly with the number of cells. Using the proposed SONCO we are able to intelligently decide which requests to accept and which to deny. By means of different configurations (SON weights) we are able to favor in any way we want the SON functions. Simulation results show KPI values matching the operator preferences reflected by the SONCO weight parameters. Future work will focus on further improving the scalability of the solution and will direct towards more challenging scenarios.

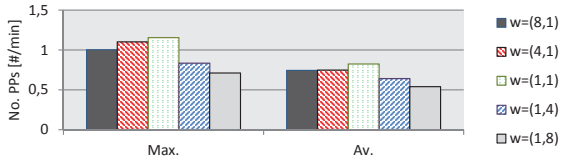

Figure 5. Average Number of Ping Pongs $[\#/\text{min}]$

ACKNOWLEDGMENT

The research leading to these results has been carried out within the FP7 SEMAFOUR project [15] 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] R. Combes, Z. Altman, and E. Altman, "Coordination of autonomic functionalities in communications networks," *CoRR*, 2012.

[9] M. ul Islam and A. Mitschele-Thiel, "Reinforcement learning strategies for self-organized coverage and capacity optimization," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*.

[10] 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.

[11] ——, "SON son conflict resolution using reinforcement learning with state aggregation," in *ACM SIGCOMM 2014 - All Things Cellular 2014 Workshop*, August 2014.

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

[13] 3GPP, "LTE; E-UTRA; Physical layer; Measurements," 3rd Generation Partnership Project (3GPP), TS 36.214, 2012.

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

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

APPENDIX A
PROOF OF PROPOSITION 1

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

$$\begin{aligned}
&\mathbb{P}\left(U_{t+1}=u'|S_0=s,A_0=a,P_1=p'\right)\\
&=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u',U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\
&=\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u''),S_0=s,A_0=a\right)\cdot\\
&\qquad\qquad\mathbb{P}\left(U_1=u''|S_0=s,A_0=a,P_1=p'\right)\\
&\overset{(*)}{=}\textstyle\sum_{u''}\mathbb{P}\left(U_{t+1}=u'|S_1=(p',u'')\right)\cdot\\
&\qquad\qquad\mathbb{P}\left(U_1=u''|P_1=p'\right)\\
&=\mathbb{P}\left(U_{t+1}=u'|P_1=p'\right)
\end{aligned}\tag{15}$$

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

Now, from (6) we have:

$$\begin{aligned}
Q^{\pi}(s,a) &= \mathbb{E}_{\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t R_{t+1}|S_0=s,A_0=a\right]\\
&= \mathbb{E}_{\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t \rho_s\left(U_{t+1}\right)|S_0=s,A_0=a\right]\\
&\overset{A1}{=} \mathbb{E}_{\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t \rho_s\left(U_{t+1}\right)|P_1=g(s,a),\right.\\
&\qquad\qquad\left.S_0=s,A_0=a\right]\\
&\overset{(15)}{=} \mathbb{E}_{\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t \rho_s\left(U_{t+1}\right)|P_1=g(s,a)\right]\\
&= \mathbb{E}_{\pi}\left[\textstyle\sum_{t=0}^{\infty}\gamma^t \sum_{m\in\bar{\mathcal{M}}}\rho\left(U_{t+1,m}\right)|P_1=p\right]\\
&= W^{\pi}(p)=\textstyle\sum_{m\in\bar{\mathcal{M}}}W_m^{\pi}(p)
\end{aligned}\tag{16}$$

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

$$\begin{aligned}
W_m^{\pi}(p) &= \mathbb{E}\left[\rho\left(U_{1,m}\right)|P_1=p\right]+\\
&\qquad\mathbb{E}_{\pi}\left[\textstyle\sum_{t=1}\gamma^t\rho\left(U_{t+1,m}\right)|P_1=p\right]\\
&\overset{A2}{=}\overline{r}_m\left(p_{\mathcal{N}_n,\mathcal{K}}\right)+\gamma\mathbb{E}_{\pi}\left[W_m^{\pi}(P_2)|P_1=p\right].
\end{aligned}\tag{17}$$

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