Cognitive Cellular Networks: A Q-Learning Framework for Self-Organizing Networks

Stephen S. Mwanje, Lars Christoph Schmelz, and Andreas Mitschele-Thiel

***Abstract*—Self-Organizing Networks (SON) aim at simplifying Network Management (NM) and optimizing network capital and operational expenditure through automation. Most SON Functions (SFs) are rule-based control structures which evaluate metrics and decide actions based on a set of rules. These rigid structures are, however, very complex to design since rules must be derived for each SF in each possible scenario. In practice rules only support generic behavior which cannot respond to the specific scenarios in each network or cell. Moreover, SON coordination becomes very complicated with such varied control structures. In this paper, we propose to advance SON towards Cognitive Cellular Networks (CCN) by adding cognition that enables the SFs to independently learn the required optimal configurations. We propose a generalized Q-learning framework for the CCN functions and show how the framework fits to a general SF control loop. We then apply this framework to two functions on Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB). Our results show that the MRO function learns to optimize handover performance while the MLB function learns to distribute instantaneous load among cells.**

***Index Terms*—SON; Cognitive Cellular Networks; MRO; MLB**

I. Introduction

OPTIMAL cell sizes in cellular networks are continuously decreasing, increasing the number and density of cells especially with the introduction of LTE. This has resulted in higher capital and operational expenses, and increased complexity of network operation. Self-Organizing Networks (SON) promise to minimize these challenges through automation of network operations. SON Functions (SFs) have been defined, e.g., in the LTE SON standard [1], with each SF representing a function that can be automated. Examples are Mobility Robustness Optimization (MRO), Mobility Load Balancing (MLB), Coverage and Capacity Optimization (CCO) or Inter-Cell Interference Coordination (ICIC). These functions have traditionally been developed as rule-based controllers which require the designers to have full understanding of the function's behavior. We propose to advance SFs to CCN functions and implement them as cognitive, Q-Learning (QL) based agents that act in the network and use the network's feedback to learn the effects of their actions.

The paper is organized as follows: Section II briefly summarizes the related works, Section III discusses QL and its application to CCN functions while Section III describes the simulation scenario and environment. We apply the CCN concept to two candidate functions on MRO and MLB presented in sections IV and V, and conclude with a summary of the presented work in Section VI.

Stephen Mwanje and L. C. Schmelz are with the Network Management Automation Research team at Nokia research, Munich, Germany (e-mail: stephen.mwanje;christoph.schmelz@nokia.com)

A. Mitschele-Thiel heads the Integrated Communications Group at the Techniche Universität Ilmenau, Germany(e-mail:mitsch@tu-ilmenau.de)

Manuscript received October 8, 2015; revised January 14, 2016.

II. SON Functions: The State of the Art

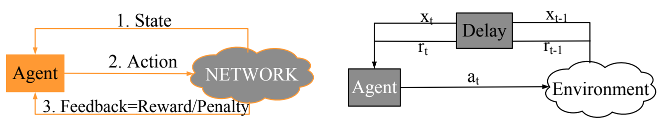
Each SF is characterized by a trigger that initiates the execution of an associated SON algorithm, which in turn configures a set of network parameters in order to optimize a particular metric. The basic SF (in Fig. 1a) is a control agent that: 1) observes the network to evaluate trigger conditions, 2) takes an action to optimize its metrics and 3) gets feedback on the effect of that action on the network. Accordingly, most SFs have been developed as rule-based controllers which select actions by applying defined rules on the observed metrics. Examples controllers are [2]–[5] for Handover (HO) optimization; [6], [7] for tilt optimization and [8], [9] for load balancing. Such an implementation, however, requires that the rules include all the possible scenarios, and that the rule designer fully understands the effects of each action in each of these scenarios. In reality, this is not possible even for a system expert. The biggest challenge is that these control loops create rigid structures that are very complex to design and hard to evolve as more functions are deployed in the network. Furthermore, they only allow for generic behavior which cannot respond to specific contexts in each network or cell.

To counter these challenges, we advance SON to Cognitive Cellular Networks (CCN) by adding cognition to SFs. The CCN concept advances SON beyond the rule based control loops towards artificial-intelligence based cognitive functions that autonomously learn the optimal configurations. Specifically, we propose to design CCN functions as QL agents that act and, using network's feedback, learn from the effects of their actions. QL has been applied in some SFs with positive results, e.g., in [10] [11] [12], but we advance this and propose a QL framework that is applicable to all optimization functions. We demonstrate the benefits of applying the framework with extended versions of the results in [13], [14].

III. Q-Learning Framework for SON Functions

A. Q-Learning (QL)

Multiple approaches have been used to develop SFs. Reinforcement Learning (RL), mainly QL [15], offers the best promise as it allows the network to learn and improve its solutions through experience. QL is a model-free RL algorithm which, using Temporal Difference (TD), solves learning problems even without models. A QL problem is the triple



(a) (b)

Fig. 1. (a) The abstract SON Controller and (b) the QL process

$(X; A; r)$, where, X, A are respectively the sets of all possible states and actions and $r : X{\cdot}A{\cdot}X' \rightarrow \mathcal{R}$ is the reward function.

Assume the learning agent (Q-learner) in Fig. 1b, which applies an action $a_t \in A$ at time t when the environment is in state $x_t \in X$. The environment undergoes a transition into a new state $x_{t+1} \in X$ and the Q-learner receives an (quasi) immediate reward $r_t \in \mathcal{R}$. The learner wishes to choose actions that maximize its discounted cumulative rewards over time.

Assuming π a policy of choosing actions, we define a value function (the Q-value, $Q^{\pi}(x, a)$) for every state action pair, as the expected total discounted reward received when starting with action a in state x and following the policy π thereafter. For the optimal policy π^* [16]:

$$Q^{\pi^*}(x_t, a_t) = E\left[r(x_t, a_t) + \gamma \cdot \max_{a_{t+1}} \{ Q^{\pi^*}(x_{t+1}, a_{t+1}) \} \right] \quad (1)$$

where $\gamma \in (0, 1)$ is the discount factor that balances between the immediate and future rewards. QL maintains estimates of the Q-values, denoted as $\hat{Q}$, and adjusts them based on received rewards using the TD error $e(x_t, a_t)$ [16]. $e(x_t, a_t)$ is the difference between the actual Q-value ($Q(x_t, a_t)$) and its current estimate ($\hat{Q}(x_t, a_t)$). Thus the estimate at time t+1 is updated by adding a small portion (i.e. α) of the error (difference) to the current estimate as:

$$\begin{aligned} \hat{Q}_{t+1}(x_t, a_t) &= \hat{Q}_t(x_t, a_t) + \alpha_t \cdot \Big\{ Q(x_t, a_t) - \hat{Q}(x_t, a_t) \Big\} \\ &= (1 - \alpha_t)\hat{Q}_t(x_t, a_t) + \alpha_t \cdot \Big\{ r(x_t, a_t) + \\ &\qquad \gamma \cdot \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \Big\}. \end{aligned} \quad (2)$$

$\alpha \in (0, 1]$ is the learning rate that balances new information against previous knowledge. $\alpha = 0$ implies no learning while $\alpha = 1$ means that only the latest information is considered and the old knowledge completely disregarded.

B. QL Process and the Exploration of Actions

QL does not specify the exact actions taken in each state. But if each Q-value is updated infinitely often, $\hat{Q}^*$ converges to Q^* with probability 1, provided that α is reduced to 0 at a suitable rate [17]. Multiple approaches for finding the compromise between exploration and exploitation of the state-action space exist, the most common being ϵ-greedy [16]. We use here the ϵ-first strategy which is a special form of ϵ-greedy.

With ϵ-greedy, at each iteration i, the agent takes the optimal action $a_i = arg \max_a \hat{Q}^*(x_i; a)$ with probability $1 - \epsilon$, otherwise it takes a random action. Initially, ϵ must be huge (near 1), but must reduce to $\epsilon = 0$ when $\hat{Q}^* \approx Q^*$, so as to always use the optimal policy. ϵ-first is similar to ϵ-greedy but with a step change between exploration and exploitation. It begins by exploring all the actions without exploiting until all actions have been tested a given number of times. At that point it changes to exploitation of the learned knowledge. The naive (ϵ, δ) algorithm [18] explores all the actions at least N times before exploiting giving, with probability $1 - \delta$, an ϵ-optimal action for each state, with

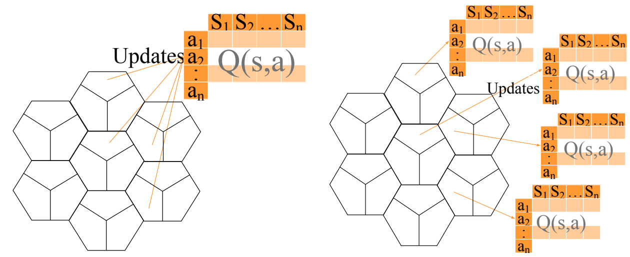
$$N = \frac{2}{\epsilon^2} ln \frac{2n}{\delta}. \quad (3)$$

n is the number of actions; δ the probability that an action is non-optimal; and ϵ is the greed or desired speed of convergence with small ϵ values giving more accuracy at the cost of delayed convergence. In this work, we have selected ϵ=0.4 and optimality probability of 80% ($\delta = 0.2$) which represent mid ground greed while allowing good accuracy. The step-by-step flow of activities is summarized by Algorithm 1.

Algorithm 1: The Q-Learning Algorithm

initialize: Q-value estimates as $\hat{Q}(x_t, a_t) := 0 \ \forall x \in X, a \in A$
1. Observe the current state $x = x_t$
2. Select and execute an action a_t
3. Receive immediate reward r_t
4. Observe new state $x = x_{t+1}$
5. Update estimate $\hat{Q}(x_t, a_t)$ according to Equation 2
6. update time $t \leftarrow t + 1$ and current state $x \leftarrow x_{t+1}$
7. Repeat steps from 2 to 6

until the terminal condition is fulfilled



(a) Cooperative distributed Learning (b) Fully distributed Learning

Fig. 2. Distributed learning strategies

C. Multi Agent Q-Learning

The above QL algorithm focused on a single learning agent, yet distributed SON is a Multi-Agent System (MAS) even for one SF, since optimization must be done for each cell individually. Therein, we must decide whether to apply cooperative or fully-distributed learning. For a given QL problem, if an observed state at one agent A will, at some other time, be observed by another agent B, then A and B should share their observations and learn a shared policy instead of learning independent policies. This results in Cooperative QL where the agents independently select actions but update a single Q-table, as shown in Fig. 2a. The alternative is Distributed QL where each agent learns and updates an independent Q table (as in Fig. 2b) with, as expected, the reverse merits and demerits.

Besides the benefit of enabling cells to share experience, Cooperative QL also speeds up the learning processes. Given the access to the common policy learned by all cells, each cell does not necessarily need to experience each state on its own for it to learn the best action in that state. The challenge here is that the single policy may not be optimal for all agents in all states. Nevertheless, where the QL states are defined in a way that they are fairly consistent across agents (cells),

TABLE I
MAPPING SFS TO QL FRAMEWORK

SF	States	Solution: QL agent adjusts ..	Action to learn	QL agent rewards ..	QL agent penalizes
MRO	Mobility state, e.g., average UE speed in cell	Hys, TTT	Hys-TTT tuple, e.g., (2.5,0.64); ?.. (dB,s)	change in no. of RLFs and ping-pong HOs	
MLB	serving & neighbor cell load, user distribution	CIO by δ	sizes of δ, e.g., [δ =0.5,1.0, ..]	change in load	HO effects
CCO	serving & neighbor cells' spectral efficiency, Tx Power	Antenna Tilt or Tx power by δ	sizes of δ, e.g., [δ =0,1,2, ?]	change in spectral efficiency	User dissatisfaction
ICIC	serving & neighbor cells' spectral efficiency, Tx Power	Tx Power, spectrum allocation policy, ..	absolute or change in Tx power, spectrum allocation, ..	Mean throughput, interference level	HO and Load effects
:	:	:	:	:	:

Legend: Hys - Hysteresis TTT - Time to Trigger CIO - Cell Individual Offset RLF - Radio Link Failure

Cooperative QL should improve the convergence speed. MRO and MLB are example SFs where similar states are observed in different cells. This makes MRO and MLB good candidates for cooperative QL, the challenges above notwithstanding.

D. Generalization of QL for SON functions

As stated earlier, RL has been applied in a number of SON related works, e.g., on CCO [19] [10], Interference Management [11] [12], Load Balancing [20], [21] and HO management [22]. This paper generalizes this approach by considering all SON functions as QL problems some of which are presented in the subsequent sections. As can be seen in Fig. 1, the structure of a typical SF (Fig. 1a) is the same as that for a learning agent (Fig. 1b). As such we consider each SF to be a Q-learning agent given the appropriate elements.

Essentially, with the radio network as the environment for each SF, we define the state(s) for which actions are required. These must be related to the observations that trigger the SFs, e.g., MLB states should consider the degree of cell overload since MLB is triggered by cell overload. Given the states, we define the action set as the different possible parameter values that can be applied in such states. Then, owing to the need to quantify the quality of the actions, a feedback mechanism in form of a reward system for the actions is added. For each action taken, a reward is derived for the SF QL agent from which the agent learns the best actions over multiple interactions with the network. Table I summarizes how some common SFs can be mapped to the QL framework.

The biggest benefit of the framework is that each SF can be designed to learn based on all metrics influenced by its actions. As such each SF will learn not only to optimize its metrics but to also minimize its effects on peers' metrics. In table I, e.g., with knowledge of the dependence of MRO from MLB, the MLB QL agent can be designed to learn based on MLB's metrics (load; user dissatisfaction; ..) and on HO metrics (ping pongs; Radio Link failures; ..). Sections V and VI respectively show the application of the framework to MRO and MLB.

E. Convergence and complexity:

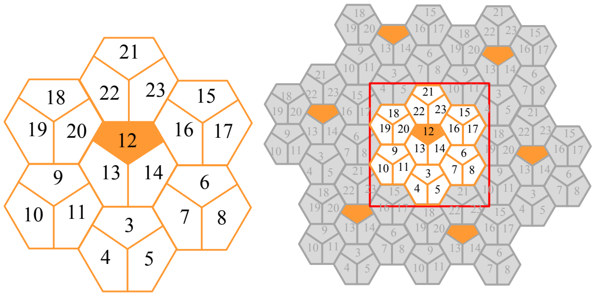
QL converges with probability 1 under the condition of bounded rewards and using, for each update t, a step size α_t so that $\sum_t \alpha_t = \infty, \sum_t {\alpha_t}^2 < \infty$ [17]. In our studies we ensure bounded rewards by design and maintain a small learning rate α. However, due to the use of a non reducing rate, we then forced termination of the algorithms after a long enough learning period as described in section III-B. Authors in [35] state that "although QL performs undirected exploration, its worst-case complexity is polynomial in the number of states n", specifically a sample complexity of $O(nlogn)$ [36]. In our studies, the states are independent to the extent that any action does not change the environment into another state but only responds to it. This translates the state space into a set of distinct QL problems each of space size 1 and k actions, resulting in a sample complexity $O(k)$ per individual QL problem. Then, without loss of generality, the sample complexity of QL on the global problem will be $O(nk)$. Meanwhile, since each cell observes exactly one state at each point in time, the learning agent has to update each of the cells independently.

IV. SIMULATION SCENARIO AND ENVIRONMENT

The QL framework was evaluated using a C++ LTE system-level simulator (tool) based on libraries from Nokia Bell Labs, Germany and the Institute of Communication Networks and Computer Engineering at the University of Stuttgart, Germany (IKR) [23]. The tool simulates the down-link of a 3GPP-compliant LTE radio network with the parameters described in [24]. It was extended with classes that define the required SON functionality, i.e., a separate class is added for MRO and for MLB. Similarly, a separate class *RLagent* that defines a generic QL agent is added so that each SF can instantiate its own QL agent as an object of *RLagent*. Meanwhile, to maintain its local state of the optimization, each cell instantiates its own local object of each SF.

A. Radio Conditions, User Throughput and Cell Load

As shown in Fig. 3, we assume a network with 7 tri-cell LTE base stations (eNBs), with the cell as the coverage area of a single transceiver (Tx). The eNBs are deployed in a regular hexagonal structure with a 500 m inter-site distance and wrap-around implemented for better interference calculation. User throughput defined as the maximum transmission data rate that the user can achieve on a given channel depends on the user's Signal to Interference and Noise Ratio (SINR). For a given SINR S, we use the approximate function in [25] that



(a) Simulation network (b) Network with wraparound

Fig. 3. Simulation network structure with and without wraparound

estimates N_B the number of bits that can be carried per symbol using realistic Modulation and Coding Schemes (MCS) as:

$$N_B = \begin{cases} 0 \; ; \; S < 7.04 \\ -0.0001S^3 + 0.0074S^2 \\ \quad +0.1397S + 0.6218 \; ; \; -7.04 < S < 20.2 \\ N_B(20.2) \; ; \; S > 20.2 \end{cases} \quad (4)$$

We consider the Guaranteed Bit Rate (GBR) traffic model where, as a minimum, a user must be allocated resources ensuring it achieves the specified rate. Then, the number of Physical Resource Blocks (PRBs), N_{PRB}, required for data transmission per scheduling interval T_s will be the ratio of the total data to be transmitted during T_s to the user's achievable rate per PRB. Consequently, the cell's offered load ρ is the ratio of the required PRB count for all the cell's users to the number of available PRBs. Given that the PRB has 7 symbols in a bandwidth B_{PRB} of 180 kHz [24], the offered load is

$$\rho = \frac{B_{PRB}}{B_{sys}} \cdot \sum \frac{GBR \cdot T_s}{7N_B} \quad (5)$$

Cell overload occurs when the offered load ρ exceeds a preset threshold ρ_{max}. Accordingly, ρ can be greater than 1, which represents the case where the total required PRBs exceed the available maximum PRBs within B_{sys}. Each cell allocates PRBs to its associated UEs using a simple round robin scheduler. The scheduler continuously allocates resources to users ensuring that each achieves the desired rate before allocating the next user, and this continues either until all resources are exhausted or until all users are allocated.

B. Mobility and Handover Management

HOs between a serving cell s and a target cell t are triggered according to the A3 condition [26], which, using the Reference Signal Received Power (RSRP) for HO from s towards t, is

$$F_t + O_t^{s,t} - Hys > F_s + O_s^{s,t}. \quad (6)$$

F_t,F_s are respectively the user's RSRP in dBm in t and s cells, without any offsets; $O_t^{s,t}$, $O_s^{t,s}$ are the respective Cell Individual Offsets (CIOs) while Hys, which is uniform for the serving cell, is the Handover Hysteresis (Hys) in dB.

If A3 is fulfilled for a critical time called Time To Trigger (TTT), the UE initiates HO by sending a measurement report of the values F_s and F_t after being filtered by a Layer 1 averaging filter (L1) and a Layer 3 Infinite Impulse Response (IIR) filter (L3). *L3* is implemented as specified in the LTE

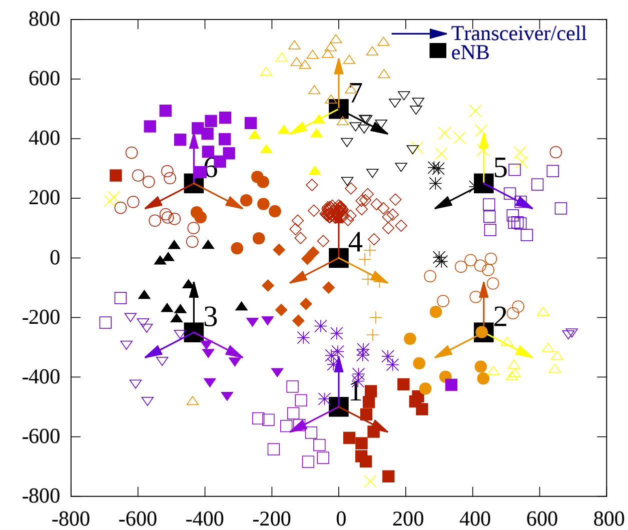


Fig. 4. Initial Network with deployed users colored according to serving cell

Radio Resource Control (RRC) specifications [26] with a filter coefficient of 6. Meanwhile, *L1* averaging which is not standardized and can be vendor-specific, is implemented as a simple moving average of the most recent values. The averaging window, which must be updated every $200ms$ and is required to be long enough to average fast fading yet not so long to affect the results of *L3*, is selected to be 100 samples.

It is evident that HO outcomes mostly depend on Hys and TTT both having the effects of either delaying or advancing HOs. The possible outcomes are either a HO success, a Ping-Pong HO (PP) or a Radio Link Failure (RLF). These are modeled according to the timers defined by 3GPP in [26].

C. Simulation Model

The simulation starts with cell deployment. It then executes multiple "batches" starting each with user deployment. This redeployment ensures that users experience as many of the prevalent radio conditions as possible, since in each batch, each user is placed at a different location and follows a different path during the execution of the batch.

Mobile devices are deployed following a random distribution, in a way that the average number per cell is the ratio of the total number of users to the number of cells. For MLB studies, a hot-spot-induced overload is artificially added by deploying static users (i.e. UEs with velocity = 0 m/s) in a "center cell", specifically, cell 12 in Figure 3a. An example distribution of the users after deployment is shown in Fig. 4.

Test studies showed that statistics for HO events are stable after at most 80 batches of 200 s each. MRO related results are therefore based on simulations of 120 batches each simulating 200 s of operation. Load statistics are, however, stable after at most 20 batches, and so MLB results are based on simulations of 30 batches each simulating 200 s of operation. The crucial simulation parameters are summarized in Table II. The next sections will show performance results of the two CCN based SFs when simulated with the described simulation tool.

TABLE II
SIMULATION PARAMETERS

Parameter	value
System bandwidth	10 MHz
Inter-site distance	500 m
Snapshot interval	50 ms
Number of users	240 mobile, 40 static
User velocity	variable, mean =3, 10, 30, 60 or 120 kmph
Mobility Model	random walk
Pathloss	$A+B \cdot log_{10}[max(d[km], 0.035)]$; A=128.1 and B=37.6
Shadowing	Standard deviation = 6 dB; Decorrelation distance = 50 m
eNB Tx power	46 dBm
eNB Tx antennas	1 per sector, gain 15 dBi, at height = 32 m
UE antennas	1 Omni, gain 2 dBi, at height = 1.5m
Data rate	512 Kbps

V. QL FOR MOBILITY ROBUSTNESS OPTIMIZATION

A major activity in network operations is determining the optimum HO settings, with Hys and TTT as the main control parameters. These need to be configured according to the prevailing user speeds in the cell or network, which translates into both a large state-space and a large parameter-space. Such large spaces can not be effectively evaluated manually which MRO seeks to mitigate. This section discusses our proposed Q-Learning based MRO (QMRO) solution.

For each HO, depending on the Hys-TTT tuple, hereafter called the Trigger point (TP), either a HO success, a Ping-Pong (PP), or a RLF occurs. MRO seeks to optimize the radio link robustness amidst the User Equipment (UE)'s mobility and subsequent HO, i.e. to minimize RLFs and concurrently reduce PPs and unnecessary HOs [1].

Multiple studies have been done on MRO with the major results being reported in [27]–[31]. Virtually all these studies relied on expert knowledge control loops to search through the parameter space. These approaches make two fundamental assumptions that do not hold in real networks:

1) that the mobility profile in the network is static to the extent that a single search is adequate to get the best settings. This is never the case, while the alternative of re-initiating the parameter search each time the velocity profile changes is also impractical;
2) that, when designing rules, the designer understands the underlying dependence of HO metrics on the control parameters. Besides being prone to error in case of wrong assumptions about this dependence, the required rules would be very complex even with the right model.

To counter these challenges, QMRO does not rely on expert knowledge or rules, but learns the Optimum Trigger Points (OTPs) as would be derived from the perfect dependence model. It abstracts UE velocities into a set of mobility states, so as to learn the OTP for each state.

A. HO Performance Metrics

Increase in Hys and/or TTT delays HO triggering, subsequently reducing HOs and PPs. However, when the HO is over delayed, the SINR degrades so much that a RLF occurs, specifically the RLF due to Late HOs (RLFLs). The reverse happens when the Hys and/or TTT are reduced in a bid to trigger the HO earlier. In that case the HO is made to a cell whose signal is not consistently good that the UE re-initiates a HO back to the original cell resulting into a PP. In the extreme case, the SINR in the new cell is so poor that the UE loses its link before or during the reverse HO resulting in a RLF due to Early HO (RLFE). Three metrics should thus be considered for HO performance, i.e., 2 RLF rates and the PP rate.

1) Radio Link Failure Rate (F): A RLF occurs if the UE SINR stays below a threshold for a duration of the critical time (T310) [26]. The RLF rate (denoted by F), due to either a too early HOs (F_E) or a too late HOs (F_L), is the per second number of RLF events evaluated for the cell or the network.

2) Ping-Pong rate (P): A PP or HO oscillation is registered for a user if a HO success from a cell B to another cell A occurs in a time less than the ?PP-Time? after a previous successful HO from A to B. The PP rate (P) is thus the rate of occurrence of PPs per second in the cell or network. The *PP-Time* is not standardized. In this work, it has been set to be approximately equal to the longest TTT (i.e. $PP-Time = 5s$).

3) Number of HO Candidates (NH): During learning, all rates are normalized to NH in the cell to ensure that all cells use comparable statistics in evaluating their actions. This is not required for the network-wide statistics, since in that case we consider the same number of users and mobility patterns. Meanwhile, the actual NH (those who are ready for HO) depends on the two HO parameters, i.e., the Hys and TTT will determine if a UE is due for HO or not. This creates a cyclic dependence as Hys and TTT determine NH yet we need NH to evaluate the right Hys and TTT. To overcome this, we redefine a HO candidate as one who has either initiated a HO or experienced a RLF within the evaluation time interval, ensuring that each user is counted only once even where a single user experiences multiple events within the interval.

4) The HO Aggregate Performance (HOAP): In order to effectively compare trigger points so as to select the best one, a single metric is needed and any comparison using the 3 metrics would be impractical. As such, we translate the 3 metrics P, F_E and F_L into an aggregate metric, the HOAP, as a weighted combination given by Equation 7.

$$HOAP = w_1 P + w_2 F_E + w_3 F_L; \quad \sum w_i = 1 \qquad (7)$$

The following should be noted regarding the HOAP:

1) HOs and HO successes are not directly included in the HOAP since minimizing PPs also minimizes unnecessary HOs and HO successes
2) HO failures being RLFs that occur during the HO process, they are also not directly included, but are assumed to be counted under RLFs.
3) In addition to inappropriate HO settings, RLFs could be due other causes, e.g., coverage problems. We assume

here however, that such do not exist or their effects are so small and can be ignored. This is a justified assumption based on test results which showed that if HO are triggered adequately early (Hys=0 dB and TTT=0 s), RLFs will be eliminated (with excessively high PPs).

4) The selection of weights w_i is subjective. They have here been selected so as to equally balance effects of early HOs (P, F_E) against effects of late HO (F_L), i.e., w_3 equal to w_1 and w_2 combined. Then, since RLFs are less desirable compared to PPs, w_2 should be larger than w_1. The correspondingly selected weight vector is $w = (0.2, 0.3, 0.5)$ and is the one used in all cases where HO performance is evaluated.

B. HO Control Parameters Sensitivity

The core MRO goal is to dynamically select the optimum settings (OTP) even for a network with a dynamic mobility profile. To design an adaptive learning strategy, we investigate the sensitivity of the parameters to UE velocity. We do so by sweeping a selected range of the parameter space for four velocity scenarios. With UEs moving at constant velocity in each scenario, we observe that the OTP changes with velocity as shown in Fig. 5. Fig. 5a gives the linear variation of the HOAP with both Hys and TTT while Fig. 5b describes the detailed variation with TTT using a TTT log scale.

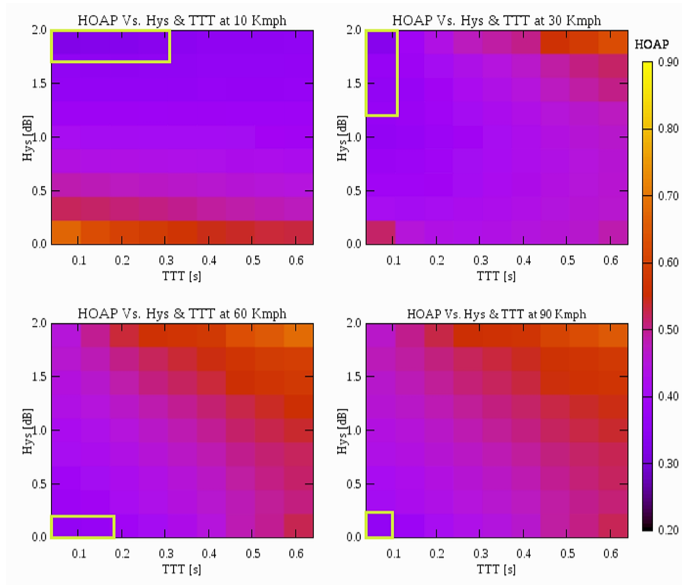
We observe in Fig. 5a that a very high Hys is unacceptable at all velocities, although a combination of moderately high Hys and low TTT could be acceptable. Similarly, a high TTT is only acceptable at low velocities and only in combination with low to medium Hys. Even then, the best settings at low velocity should be medium Hys with low-to-medium TTT, i.e., HOs can moderately be delayed without great penalty since the risk of RLF is low yet even the possibility of PPs is low owing to the low velocity. This is evident in the 10 kmph case where for most TTTs the performance is good at Hys $= 2dB$.

As the velocity increases, the HO delay needs to reduce especially using the TTT. The HOAP is more susceptible to change in TTT, to the extent that the OTP continuously grazes the Hys axis, i.e., the performance changes with TTT but is fairly constant with Hys. At high velocity, even the Hys has major effect and so both parameters should be low. This is evident in the 60 and 90 kmph environments in Fig. 5a where the OTPs are restricted to the lower left corners of the grid, i.e., the part where TTT are within the range of 0-0.64 s. In Fig. 5b we observe that within this small range, although there is major variation in the HOAP with TTT for most Hys, this variation is blurred at points near the optimum point. In that case adjacent TTTs will have practically similar performance.

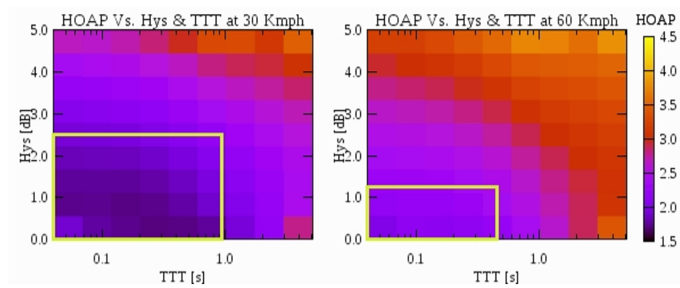
The most obvious conclusion from Fig. 5 is that the OTPs do not lie along any one diagonal for the different velocities as was assumed in [28] and [29]. Any MRO algorithm must thus scan the entire parameter-space or at least more than half the space in order to determine the required trigger point.

C. QMRO: Q-Learning based MRO

QMRO wishes to determine the optimum *Hys-TTT* action that minimizes the HOAP in any mobility state in a cell.



(a) Linear variation of HOAP with Hys and TTT



(b) Detailed (log scale) variation of HOAP with TTT for 30 and 60 kmph

Fig. 5. Handover Control Parameter Sensitivity

We note here that the actions only affect the performance of the cells and do not change the UEs' mobility states. Consequently, it is adequate to learn an action *a* in state *x* that maximizes the expected instantaneous reward *r* at time *t*. As derived in Equation 2, the corresponding Q-update algorithm for instantaneous rewards is Equation (8)

$$Q_{t+1}(x_t, a_t) = (1-\alpha)Q_t(x_t, a_t) + \alpha[r_t(x_t, a_t)] \quad (8)$$

where α is the learning rate previously defined in section III-A. The components of the QMRO algorithm are described below.

1) QMRO State Space: The required HO settings in a cell depend on the mobility of the UEs in the cell as showed in the parameter sensitivity analysis in section V-B. Thus, the states *x* are defined to be the degree of mobility in the cell evaluated as the average velocity over the *SON interval*. The average velocity being a continuous variable, we discretize the states *x* into bands for which the appropriate HO settings must be learned. Table III describes how velocities are grouped into mobility states and, based on the results in Fig. 5, the estimates of the likely default settings that would be used in a manual optimization process. We assume that the velocities are known or at least can be estimated by the cells. A simple estimate can be obtained as the ratio of the approximate cell size to the average time that a UE stays in the cell. It can also be more accurately estimated using the UEs Doppler power spectrum as proposed in [32]. Either way, with a good estimate, QMRO

TABLE III
QMRO MOBILITY STATES AND THEIR DEFAULT ACTIONS

Average Velocity (kmph)	State (x)	Default Hys (dB)	Default TTT (s)
0-4	0	3.0	0.0-5.2
4-8	1	2.5	0.0-2.56
8-12	2	2.0	0.0-1.25
12-17	3	2.0	0.0-1.02
17-22	4	1.5-2.0	0.0-0.64
22-28	5	1.5-2.0	0.0-0.48
28-34	6	1.5-2.0	0.0-0.256
34-41	7	1.5	0.0-0.52
41-48	8	1.0	0.0-0.52
48-56	9	0.5	0-0.48
56-65	10	0.5	0.0-0.256
65-75	11	0.0-0.5	0.0-0.16
75+	12	0.0-0.5	0.0-0.16

can then learn the best configuration for the given cell.

2) QMRO Action Space: Actions are the Hys-TTT tuples signaled by the cells to their associated UEs. Without a SON solution, an operator configures a cell with default parameter settings obtained through trial and error, while with a local search based SON solution, a fixed set of settings similar to those shown in Table III are applied. However, with the observation that OTPs depend on speed, we need to change the settings based on the instantaneous speed in the cell.

It is evident from the parameter-sweep results in Fig. 5 that for all practical speeds, performance at Hys>5 dB is almost always sub optimal. We thus consider Hys values only up to 6 dB. Meanwhile differences in HOAP for some TTT settings are unresolvable especially at low TTT values. For example for most Hys values at all velocities, the performance at TTT = $0.08s, 0.1s, 0.16s$ is practically the same. As such not all TTT values are considered, i.e., the possible TTT actions are the 11 values 0.04, 0.10, 0.128, 0.256, 0.32, 0.48, 0.512, 0.64, 1.02, 1.28, 2.56, 5.12 in s. The resulting action space (total number of actions) for each state is 143 possible combinations of the considered Hys and TTT.

3) QMRO Reward function: We desire to minimize RLFs without excessively increasing PPs and HOs. Since the learner is a rewards-maximizing agent, the reward $r_{x,t}$ should be the negative HOAP evaluated over the SON interval. As stated earlier, the individual rates are normalized to the number of HO candidates, NH as given in Equation 9.

$$r_{x,t} = -(w_1 P + w_2 F_E + w_3 F_L)/NH; \qquad (9)$$

Meanwhile, the weight vector applied during learning may need to be adjusted to enforce particular results especially given the small evaluation period (the SON interval). For example, there may be instances in which no RLFEs are observed during the SON interval which could tilt the result in favor of too many PPs. The vector is thus maintained as $w = (0.2, 0.3, 0.5)$ for the typical results and changed to $w = (0.4, 0.0, 0.6)$ when no RLFEs are observed.

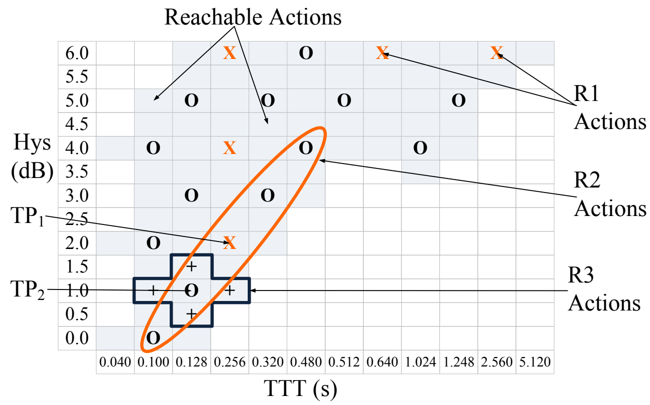


Fig. 6. QHO action space within the three learning regimes R1, R2, R3

4) QMRO Cooperative Learning: HO triggering could be affected by channel conditions that dictate the respective RSRP. However using *A3* minimizes this dependence on the absolute RSRP values, since decisions are made based on RSRP differences among the cells. In that case, HO performance depends only on user mobility and the control parameter values. With mobility-based HO states, it is possible that a state observed in one cell reoccurs in another cell at some other time. As such, cells do not need to learn independent policies but can learn a single policy function based on the abstract mobility states. The result is a cooperative QL problem in which individual cells take actions but update a single Q-table that represents the shared learned policy.

The cooperative learning solution holds if all other crucial parameters are comparable among the cells. For example the cells in the considered network are assumed to be of similar size and applying comparable transmit powers. Other than this, the RSRP profiles at the cell edges may be different resulting in differing behaviors for different cells or cell pairs. Similarly if individual cells concurrently have users with differing behavioral patterns, the solution may fail. For example, the assumption that a state in one cell will be observed in another will not hold for a cell which covers a highway crossing through an office park. Such a cell concurrently has 2 user groups - the slow-moving office users and the fast-moving highway users, each of which will require different settings. Nevertheless, for the majority of networks that do not have such special conditions, cooperative learning as considered in these studies would be fully applicable.

D. Parameter Search Strategy and Optimization Algorithm

As earlier stated, each cell has up to 143 possible actions to consider for each velocity state. Thus even with cooperative learning, evaluating each action multiple times, would still require a long time to converge to the desired solutions.

1) The Parameter Search Strategy: To accelerate convergence, we subgroup the 143 actions so that for any state, we execute 3 learning regimes $R1 - R3$ (shown in Fig. 6) that start with a global exploration and gradually move to a local. For *R1*, actions are selected from different regions of the grid in order to determine the area in which the desired action lies. From the parameter sensitivity analysis, combinations of low

Hys and high TTT are never optimal. As such this region is excluded from the possible candidates denoted by ”R1 actions” in Fig. 6. The outcome trigger point of *R1* (TP_1) specifies the region in which the optimum point lies. This outcome is thus used to define the search space for the next regime *R2*.

During *R2*, actions along the diagonal that goes through TP_1 are explored to obtain the approximate delay that is acceptable for the observed mobility state. In this case, subsequent actions differ in Hys by 1 dB to enable a large enough action space to be explored. As an example, if TP_1 is obtained as $TP_1 = (2.0dB, 0.256s)$, at *R2* the agent explores the region marked *R2 Actions* in Fig. 6. The obtained TP (TP_2) is then used to define the search space for the next regime *R3*.

R3 refines the learned TP_2 by exploring points near TP_2. It compares TP_2 with its four neighbor points to the left, right, top or below. In Fig. 6, ’*R3 Actions*’ shows the exploration region for *R3* assuming that $TP_2 = (5.0dB, 0.128s)$.

2) The SON Interval: Each setting that is applied in a cell is monitored over a period of a SON interval. With different number of users in each cell and in the HO regions, different cells may have different counts of events within the same time period. Consequently, instead of setting the SON interval based on a fixed time period, it is based on a minimum number of HO events that must occur following the application of any action or configuration. This minimum number is the sum of the disjoint HO related events (i.e. HOs, RLFEs and RLFLs). This number is set to 100 events although any value that ensures that comparable counts of all the necessary statistics (i.e. for NHs, PPs and RLFs) are observed would be appropriate.

3) The QMRO Optimization Algorithm: Given the QL elements as discussed above, the optimization algorithm is given in the procedure of *Algorithm 2*, i.e: For each possible state, the action set is initialized with R1 actions and the Q-table entries initialized to 0. Learning is then triggered to be executed after every SON interval t. Each cell c observes its environment over the interval t and at the end of t, the cell determines if an optimization is necessary, i.e., if the cell's velocity state has changed. During the learning phase, c selects an action as described in section V-D1, otherwise it selects the best action that would have been learned. It then signals that action to all its associated UEs and starts collecting the necessary performance statistics for the next interval($t + 1$). At the end of interval $t+1$, c evaluates its HOAP and derives the reward r_t for the action at t. It then updates the learning agent (the Q-table) before repeating the process.

Algorithm 2: QMRO - The Q-Learning MRO Algorithm

Require: UE velocities during *SON* interval, action set a
1. Set Ri=R1; initialize action set $A_{x,R1}$ for regime 1 in all states x
Repeat for each SON interval t
2. **if** HO action was taken at SON interval $t-1$ **do**
3. determine *HOAP* and derive reward $r_{t-1}(x_{t-1}, a_{t-1})$
4. update Q-table according to Equation 8
end if
5. determine current mobility state x_t (from table III)
6. **if** learning complete for state x **do**
7. select $a_{x,t} = a_x^{opt}$, the best action for state x
8. **else if** regime Ri exploration is incomplete **do**
9. select $a_{x,t}$ (sequentially after $a_{x,t-1}$) from $A_{x,Ri}$
10. **else do**
11. select $a_{x,t} = a_{x,Ri}^{opt}$, the optimum value for state x at Ri
12. **if** all learning regimes complete for state x **do**
13. record all regimes complete for state x
14. record $a_{x,t}$ as best action in state x
14. end learning, indefinitely use $a_{x,t}$ in state x
15. **else do**
16. $Ri \leftarrow Ri + 1$
17. use $a_{x,t}$ to set $A_{x,Ri}$ i.e. reconfigure A for Ri
end if
end if
18. Signal selected action $a_{x,t}$ to all UEs in the cell.
19. $t \leftarrow t+1$, monitor, collect statistics, continue at step 2
end loop

TABLE IV
QMRO VELOCITY SCENARIOS

City Area	Initial velocity (kmph)	
	Mean	range
Office park	3	2 - 4
City Center	10	6 - 14
City Edge	30	18 - 42
City Suburb	60	36 - 84
Highway	120	72 - 168

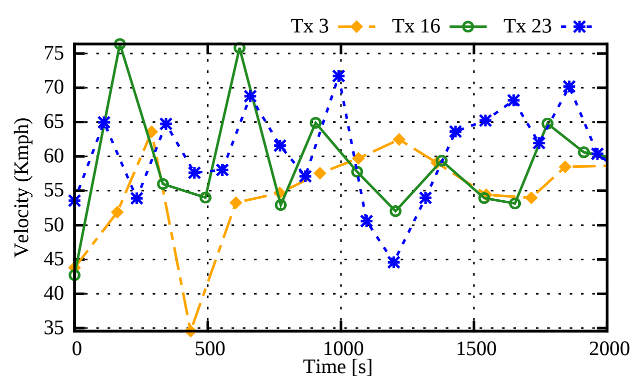


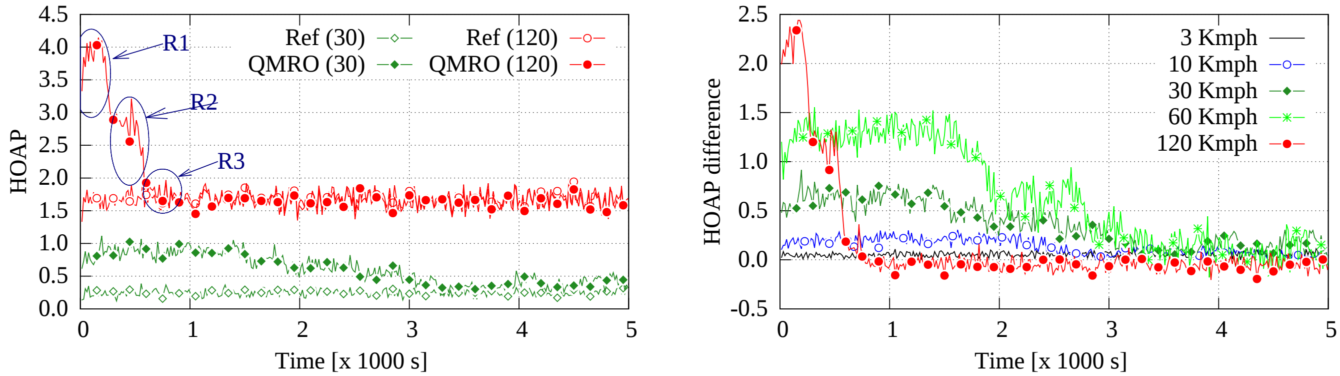
Fig. 7. Typical velocity profile in 3 cells in the $60kmph$ scenario

E. Simulation Results and Discussion

MRO wishes to adjust the HO parameters in line with varying mobility states. We evaluate performance in the 5 different velocity scenarios in Table IV to prove that the algorithm is applicable to any network. The 3 ’normal’ scenarios (*10, 30, 60 kmph*), for example, represent 3 typical city districts - a city center, city edge and residential suburb. We then consider two extreme scenarios (*3 & 120 kmph*) which could respectively represent an office park and a highway.

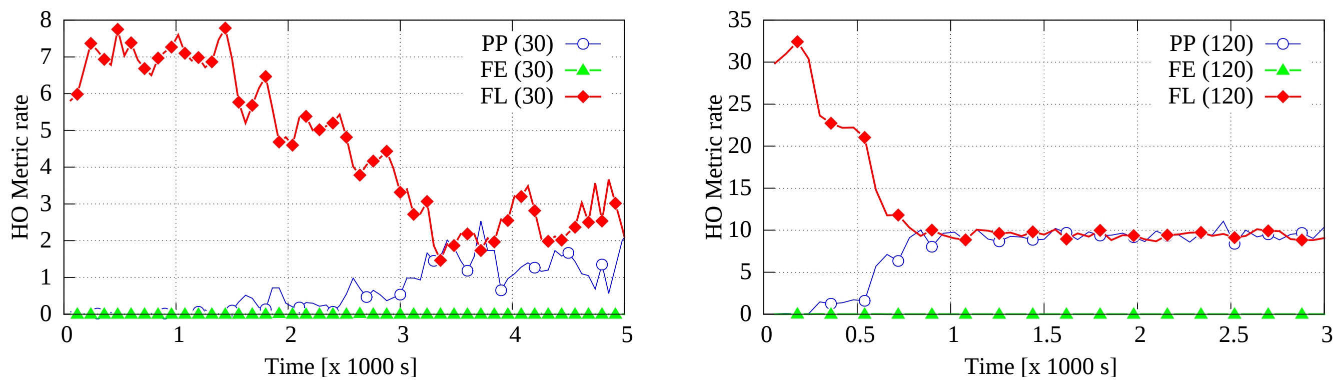
In each mobility scenario, all UEs have independent randomly varying velocities. We implement this by allocating random velocities to the UEs at the start of the simulation and also randomly adjusting the velocities at the start of and during every batch, in each by up to $\pm$ 40%. For example, each of the 240 users in the *city suburb* (60 kmph) network has an individually assigned and continuously changing velocity, as shown in Fig. 7 for the average velocities in three selected cells over a period of 10 batches.

1) Performance in terms of HOAP: To evaluate QMRO, we compare its performance in each velocity scenario against the reference network, *Ref*. This represents the case when all the



(a) HOAP comparison for two velocity scenarios (b) HOAP gains for all velocities

Fig. 8. QMRO Performance:Average network-wide HOAP for QMRO in comparison to the reference network



(a) Performance Trend at 30 kmph (b) Performance Trend at 120 kmph

Fig. 9. QMRO Performance: Variation of average rates for all 3 core metrics in 30 and 120 kmph environment

cells in the network apply the best static settings as obtained from table III in the parameters sensitivity analysis in section V-B. The performance is evaluated in terms of the averages of the metric(s) values throughout the network although cell-specific results would demonstrate the same trends.

Fig. 8 summarizes the results in the different scenarios. Fig. 8a shows the comparison of the average HOAP values for QMRO and Ref in two typical cases, while Fig. 8b describes the differences in performance between QMRO and *Ref* for all the five velocity cases. We observe in both figures that QMRO initially performs poorly as it executes the first learning regime $R1$. The performance then improves in regimes $R2$ and $R3$ as QMRO focuses around the *OTP*, to the extent that it is eventually equivalent to that of *Ref*. Where user velocities are widely spread, QMRO actually performs better since it is able to set the right setting for each velocity range as opposed to a single setting for all velocities. This is evident for the 120 kmph case in Fig. 8b where, after learning, QMRO is consistently better than Ref. Also, in a given time interval, each user undertakes more HOs as the velocity increases, with the effect that cells have a shorter SON interval at a higher velocity i.e the cells reach the minimum event count much faster. This results into shorter convergence times as the velocity increases which is evident in Fig. 8b.

2) QMRO Learning Trend: Fig. 9 evaluates the time variation of the individual metrics (P, F_E and F_L) when applying QMRO in two velocity scenarios (30 and $120kmph$). We observe in both cases, that the agent learns to minimize RLFLs (F_L) by trading them with Ping-pongs (PPs), which have less effect on the user's quality of experience. This is all while ensuring that RLFEs (F_E) remain low. In both velocity scenarios, QMRO suffers from many RLFs at the beginning as it considers settings across a large parameter space. Over time however, the agent continuously reduces F_L by trading such reduction with increase in PP. It then stops this trend as soon as it registers decreasing returns, i.e., when each extra reduction in F_L translates into an excessive increase in PPs or if it instead causes RLFEs to occur.

The foregoing results demonstrate that given a good definition of states that appropriately capture the UEs' mobility and also given adequate learning time, a Cognitive function, e.g., the QL based MRO algorithm is able to learn the appropriate Hys-TTT settings for a given mobility environment.

VI. QL for Mobility Load Balancing

Users are rarely uniformly distributed in cellular networks, but this is critical if a serving cell s is overloaded at a time when free resources exist in neighbor cells. A solution is required to automatically redistribute the load among cells - thus Mobility Load Balancing (MLB). MLB seeks to minimize the number of users who would otherwise not be satisfied, in terms of their data rate, owing to the overload in the serving cell s, i.e., to reduce the ?Number of unsatisfied users? (N_{us}).

To lower the serving cell's load ρ_s, MLB moves some of the edge users in s towards one or more neighbor cells or so called target cells. Let us denote the set of all target cells as

T, any one cell in the set as t and all cells together as T-cells. Then consider the A3 HO entry condition earlier defined in Equation 6. MLB re-distributes the load by virtually shrinking s while concurrently expanding the T-cells. As proposed in the LTE SON standard [1], this can be achieved by adjusting the relative HO margins of the respective cells s, t (and thus the applicable HO boundary between the two cells) using the CIOs ($O_t^{s,t}$ and $O_s^{s,t}; \forall t \in T$ in Equation 6).

The set T for which CIOs are adapted may contain all s's neighbor cells, a subset of them or only a single neighbor. Considering all neighbors, as is done in the network wide Load Balancing (LB) solution in [33], is straightforward. In case of a subset of neighbors, the set could be selected as those cells that fulfill a certain condition, e.g., the neighbors with very low load. Selecting a single neighbor is, however, not as obvious since we would require to select the best candidate among the neighbors. Such a neighbor, as proposed in [34], could be the one that achieves the highest reduction of load in s, without itself getting overloaded. This requires s to have access to load information in all neighbors and also be able to predict the resulting load after users have been transfered to any selected target. Moreover, in a highly dynamic network, such a prediction would very quickly change with the mobility of the users. Consequently, a simple decision to adapt CIOs for all neighbor cells or a subset may achieve better results.

We described here a Reactive MLB algorithm (RLB) that symmetrically adjusts the CIO by a value ϕ between s and all low loaded neighbor cells. However, for a fully Self-Organization (SO) solution, a distributed automated approach that learns the required adaptation of the CIOs would be preferred. We thus apply the QL framework on top of RLB for a distributed fully self organized MLB algorithm. The resulting solution, called QLB learns the best ϕ required for the particular load conditions in the serving cell's environment.

A. Evaluation Parameters and Performance Metrics

When a cell is overloaded, its users are unsatisfied since they are allocated fewer PRBs resulting in lower than expected data rates. We consider a user to be unsatisfied (an un-satisfaction event occurs) if the user's total achieved data rate in a continuous 1 second period is less than the GBR. Thus besides load variations, we evaluate the degree of undesirability of the overload situation, in a cell or the network, in terms of the ?Number of unsatisfied users? (N_{us}). This is defined as the average number of un-satisfaction events in the cell/network per second over the evaluation period. Such an evaluation period could be 5s for the quasi-instantaneous performance in the cell or network or a batch period for the long-term results evaluating the improvements achieved by the optimization algorithm(s). Meanwhile, where actions are taken to reduce overload, we evaluate the effectiveness of the actions in terms of their change in the serving-cell's offered load.

B. RLB: The Static, Reactive MLB Solution

Although it is not so obvious, the minimum MLB action is to select an edge user of cell s for HO to a cell t and then appropriately reducing the offset $O_s^{s,t}$. To avoid the user from moving right back to s, an opposite value (to $O_s^{s,t}$) is applied on $O_t^{s,t}$. This shifts the specific $s-t$ HO boundary which, however, does not guarantee that overload will not quickly reoccur due to a user at a boundary to another cell.

To manage the general load in the cell, the proposed RLB approach adjusts the generic boundary with all low-load neighbor cells. This excludes higher load neighbors as shown in Fig. 10, where boundaries are adjusted for all cells except t_4 which is highly loaded. Adapting boundaries for multiple neighbors ensures that overload does not quickly re-occur in s. Cell s applies the RLB algorithm to adjust CIOs to all T-cells (the low load neighbors) by a fixed value as in Equation 10.

$$\begin{array}{l} O_t^{s,t} = O_t^{s,t} - \phi \\ O_s^{s,t} = O_s^{s,t} + \phi \end{array} \quad all\ t \in T. \tag{10}$$

In the simplest form, CIOs could be gradually changed, each time by a small step over multiple iterations. Although this avoids unnecessary load transfer to the T-cells, it takes too long to remove the s overload. To improve convergence speed, CIOs are adjusted in a single precise step ϕ that removes overload in s without overloading the T-cells. Large CIO changes may, however, cause oscillations, where after load transfer from s to t, t gets overloaded and also initiates LB towards s, causing s to restart the process once again. RLB does not explicitly control this LB induced overload. Instead, we mitigate it using an oscillation control timer T_{oc}, which, following the s-t LB action, has to expire before a LB HO can be triggered from t to s. The size of T_{oc} is set equal to the SON interval although higher values could also be applicable.

Note, however, that minimal T-cell overload after LB HO may be a good result, as it allows the extra load to propagate outwards from the ?center? to outer cells over subsequent LB actions in different cells. This applies particularly when combined with T_{oc}, where the new s (original T-cell) does not move load back to the original s. Instead it moves the load to its other neighbors further away from the original s.

C. Dependence of RLB Gains on Load

As a manual solution, the optimal ϕ could be determined by applying different step sizes and selecting the best. The achieved change in ρ_s, $\Delta\rho_s$ for any applied ϕ is, however, dependent on ρ_s (the load in s) as well as ρ_n (the average load in t). This would make a single ϕ value inapplicable for the different load conditions. To investigate this dependency, we define a set of load scenarios and evaluate E[$\Delta\rho_s$], the expected change in s's load for each combination of scenario Γ and applied CIO change ϕ. Each load scenario Γ is a combination of ranges of ρ_s and ρ_n as shown in Table V, e.g., Γ=3 is the tuple [$0.9 \leq \rho_s < 1.1$; $\rho_n < 0.45$]. Meanwhile, $\Delta\rho_s$ is not deterministic for each combination of ϕ, ρ_s and ρ_n. The dependence can as such only be expressed in terms of the expected outcome E[$\Delta\rho_s$].

For each ϕ in a given Γ, if $\Delta\rho_s$ is observed, E[$\Delta\rho_s$] is updated according to

$$E[\Delta\rho_s] = \gamma \cdot E[\Delta\rho_s] + (1-\gamma)\Delta\rho_s \tag{11}$$

where the forgetting factor γ is set as 0.95 to ensure that any single observation does not unexpectedly skew the average.

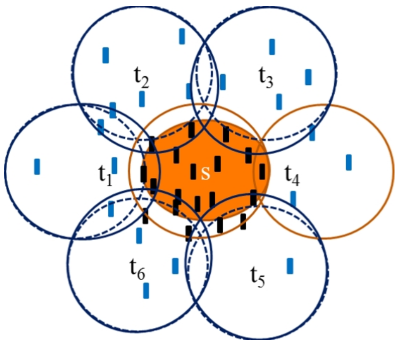


Fig. 10. Reactive change of CIOs

TABLE V
CELL LOAD SCENARIOS (Γ)

ρ_n \ ρ_s	<0.45	[0.45-0.60)	0.60+
[0.0-0.9)	0	1	2
[0.9-1.1)	3	4	5
1.1+	6	7	8

ρ_s - serving cell load
ρ_n - Average T-cell Load

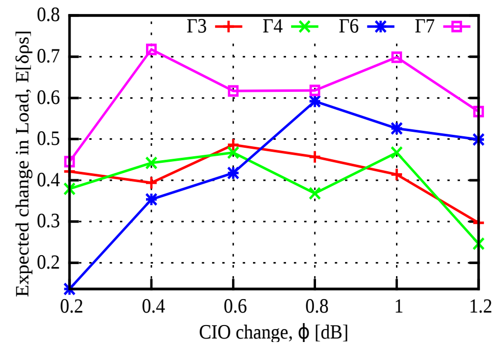


Fig. 11. Dependence of RLB gain on cells load

Fig. 11 shows the dependency of E[$\Delta\rho_s$] on Γ for different ϕ values, evaluated in the network in Fig. 3a with 240 mobile users, and the ?centre cell? initially having 40 static users. We observe that the best ϕ, ϕ_{opt}, is different for each Γ, e.g., ϕ_{opt} is 0.6, 1.0 and 0.4 for the three load scenarios Γ3, Γ4 and Γ7 respectively. This justifies the need for a learning solution that learns the required ϕ_{opt} for each of the load scenario.

D. QLB: Learning the Optimum Actions

With RLB, a fixed ϕ is found that guarantees good average performance, but not the best in each load scenario. For optimal performance, different ϕ values would be required for the different scenarios. Moreover, for any change in CIO, $\Delta\rho_s$ also depends on the user distribution (uD) in cell s, i.e., more load can be offloaded from s if there are more cell edge users. We thus apply Q-Learning based Load Balancing (QLB) in order to learn the required CIO change for each combination of ρ_s, ρ_n and uD. Here, uD describes the fraction of cell s users that are close to the cell edge, such that with more edge users, only a small CIO change is needed for these users to be handed over to neighbor cells.

QLB learns the action that instantaneously removes overload while ensuring that target cells are not overloaded as a result of its actions. Its Q-update algorithm is similar to Equation (8) with the corresponding components as follows:

1) State-space: Since ϕ depends on ρ_s, ρ_n and uD, each state is a vector [ρ_s, ρ_n, uD]. We defined 27 states as the 9 load scenarios in Table V for each of 3 ***uD*** cases: $uD < 20\%$; $uD = [20-35]\%$ and $uD \geq 35\%$. Since the idea is to change the border to all neighbors, uD is evaluated generically within cell s and not specific to any one neighbor.

2) Action-space: Actions are the possible values that ϕ can take, i.e., the values in dBs by which the CIO should be changed. Guided by RLB results, actions are selected as the discrete ϕ values [0.2, 0.4, ?.. 1.0] dB.

3) Rewards: The rewards, as set in Equation 12, consider $\Delta\rho_s$ (the achieved reduction in the serving cell load, ρ_s) and the extra load created in neighbor cells.

$$r = \begin{cases} \Delta\rho_s + 1 & ; \quad \Delta\rho_n = 2 \ \ and \ \ \Gamma < 3 \\ \Delta\rho_s & ; \quad \Delta\rho_n < 1 \\ \Delta\rho_s - 1 & ; \quad otherwise \end{cases} \tag{12}$$

Positive $\Delta\rho_s$ represents reduction in the offered ρ_s that results from users having moved to neighbor cells. Positive $\Delta\rho_s$ is thus rewarded while the reverse is penalized. Since ρ_n is expected to increase as a result of adding users at the very edge of the cells, only $\Delta\rho_n$ of more than 1 is penalized. In general, larger $\Delta\rho_s$ values receive greater reward, but are accompanied by penalties for unrestrained actions taken in LB-states with high ρ_n. This allows high load t cells to overload just enough to propagate the load outwards but not too much to counter-productively cause further un-satisfaction after LB. However, special consideration is taken in cases where a large reduction in ρ_s can be achieved without overloading the target cell. In such cases, e.g., the change from scenario 9 to scenarios 1 or 2, the reward is increased by 1.

Algorithm 3: QLB - The Q-Learning MLB Algorithm

Require: T-List(the list of s neighbors) and action set A
1. **if** LB action taken in previous SON interval i **do**
2. determine $\Delta\rho_s$ **and** derive reward
3. update Q-table according to Equation 2
   end if
4. evaluate ρ_s
5. **if** overloaded **do**
6. determine load state $l = [\rho_s, \rho_n, uD]$
7. **if** exploration complete **do**
8. select $\phi = a_l^{opt}$, the best value for state l
9. **else do**
10. from A, select $\phi = a_l^{i+1}$, i.e., in sequence of last selected value a_l^i in state l
    end if
11. **for** each cell t in T-List with timer TOC expired
12. reduce ${O_t}^s$ by ϕ **and** increase ${O_s}^t$ by ϕ
13. start timer T_{OC} for t LB HO towards s
    end for
    end if
14. Restart SON interval timer , i.e., $t \leftarrow t+1$

4) QLB algorithm: Each load state that is observed in one cell can reappear in any of the other cells in the network. As such, cells do not need to learn independent policies but learn a single shared policy in a cooperative learning process with a single Q-table which is updated by all cells.

During operation, each cell s observes its environment and at the end of a SON interval $\tau = 5s$, s applies the procedure in Algorithm 3 to either take actions that reduce overload, learn from previous actions, or both. If overloaded, s selects an action according to Section VI-D2 and signals the new HO settings (with the revised CIOs) to all its associated UEs. Since CIO changes are symmetric for each $s-t$ boundary,

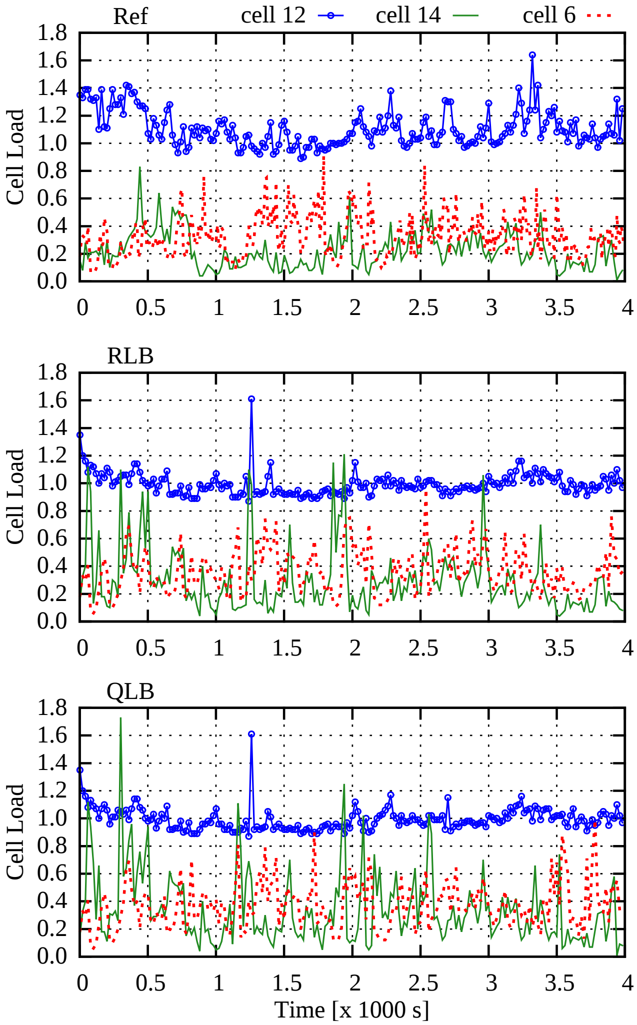


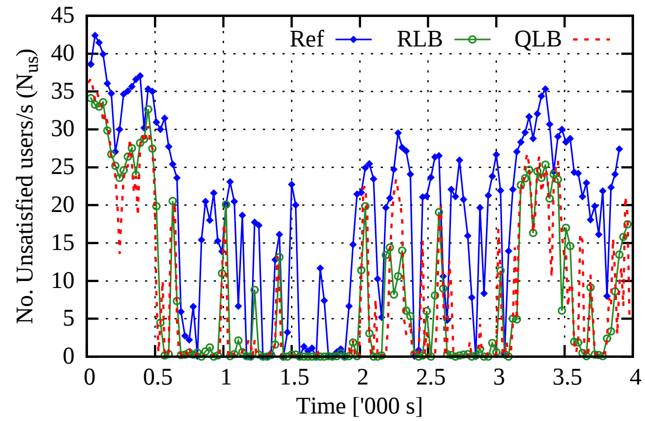
Fig. 12. QLB and RLB load redistribution (load moved from cell 12 to neighbor cells, e.g., 14 and 6)

s also sends the new CIOs to its affected neighbor cells via the X2 interface. At the end of interval $\tau+1$, s evaluates the changes in load, derives the reward and updates the Q-table before repeating the process.

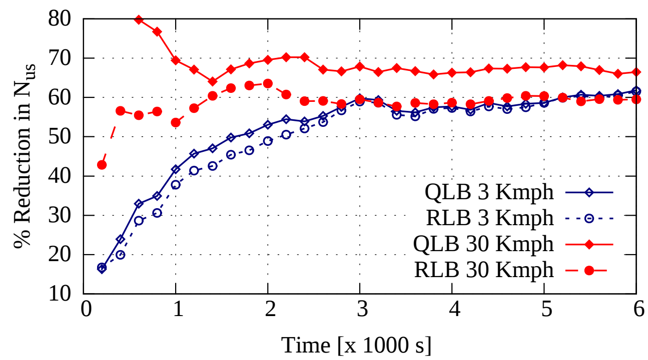
E. QLB Simulation Results

QLB was evaluated using the LTE simulator of section IV. The results, shown in Figs. 12 and 13, consider two perspectives - load variation in individual cells to evaluate the dynamic performance of the solution, and the number of unsatisfied users (N_{us}) which evaluates the global effect of the algorithm and its impact on user satisfaction. We compare the performance of QLB and RLB against *Ref*, the default scenario where the network is operated without any SON function.

1) Learning towards load redistribution: Fig. 12 shows the dynamic behavior of the load balancing solutions in terms of variations of cell load during the simulation, with the load evaluated every second for each cell. For clarity, the figure considers simulations at 3 kmph for which the dynamic behavior is slow enough to be analyzed. Otherwise, performance results at higher velocities are similar but only with much faster variations.



(a) Variation of Nus at 3 kmph



(b) QLB's reduction in Nus

Fig. 13. Effect of QLB on user satisfaction

We observe that both solutions RLB and QLB lower the load in cell 12 by transferring it to neighbors cells (e.g., cell 14) and eventually to the outer cells (e.g., cell 6). After learning however, QLB responds better to overload compared to RLB. This can be seen for the period after 3000 s where cell 12 consistently has slightly lower load for QLB than for RLB. This is the direct consequence of QLB having learned the best CIO change for each load state, which is not the same with RLB. The drawback here is that in a low mobility network, the extra load added to the neighbor cells may take long to move outwards. In that case we need to evaluate performance in terms of the Number of unsatisfied users, N_{us}.

2) QLB Effect on User satisfaction: Fig. 13 compares the performance of RLB and QLB against the reference case (Ref) in terms of the N_{us} in the network for different velocity scenarios. Subfigure 13a shows the quasi-instantaneous variation of network-wide N_{us} in the 3 kmph network scenario for the same period considered in Fig. 12. With user satisfaction evaluated every $15s$ and each point capturing the total unsatisfaction events over the $15s$ interval, we see that both RLB and QLB reduce the N_{us} through the load redistribution.

Subfigure 13b evaluates the global benefit of the solutions for two velocity scenarios across the simulation. Considering the solutions over different batches of the simulation, we evaluate the gains of each of the two solutions measured in terms of the percentage reduction in N_{us} when compared to *Ref*. We observe that for both mobility scenarios, both RLB and QLB improve user Quality of Experience (QoE) by reducing the N_{us}. The reduction in user dissatisfaction is comparable at low velocity since there is not much dynamism to be exploited by varying the CIO change. At higher velocity however, by adjusting CIOs for each instantaneous state that is observed, QLB achieves better user satisfaction.

The results above prove that a cognitive solution, in this a Q-learning based agent, can easily and successfully be applied towards a dynamic autonomous solution for MLB.

VII. Conclusion

In this paper, we have proposed the Cognitive Cellular Networks (CCN) concept as a fully autonomous approach to Self-Organizing Networks (SON). Our contributions and results can be summarized as follows:

Contributions: We (1) proposed a Q-Learning (QL) framework as the method for implementing cognitive SON Functions (SFs); (2) justified how the QL framework can be used for any generic SF; and (3) discussed the application of this framework to two selected SFs: Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB). We showed that, although all SFs use the same framework, special adjustments are required for each SF as dictated by its specific constraints, e.g., each SF required a different strategy on how to explore its action space. In general, however, the positive performance results for the developed solutions proved the benefit of cognitive approaches to SON and that QL provides a good framework for developing such functions.

In particular, Q-Learning based MRO (QMRO) is able to learn the best Handover Hysteresis (Hys) and Time To Trigger (TTT) settings for particular mobility states in the network. Applying cooperative learning, the cells learned a single policy function (single Q-table) that is thereafter exploited. Such an approach is applicable in any environment as demonstrated by the positive performance results obtained in the realistic network scenarios with User Equipments (UEs) having distinct and dynamically varying velocities.

Similarly, Q-Learning based Load Balancing (QLB) learned the best Cell Individual Offset (CIO) settings needed to reduce overload in different load conditions. Starting with Reactive Load Balancing (RLB) which adjusts the Handover (HO) boundary through the CIO, we observed that the required CIO change depends on the load state, characterized by the load in the serving and neighbor cells as well as the user distribution in the serving cell. QLB then learns the different CIO changes that are required for the different load states. Evaluating the solutions in multiple velocity environments showed that QLB achieves better results compared to the rule based reactive RLB. This is especially true in more dynamic environments, e.g., where users move at higher velocities. The subsequent effect of the load re-distribution is that the number of users, which would otherwise be unsatisfied as a result of low-data-rate induced overload, is reduced. ***Convergence and complexity:*** We have described in section III-E that with the same cost for each Q-update, the corresponding computational complexity is linear in the number of cells. In real systems, however, complexity may be a moving target which requires careful consideration of the specific case, i.e. the dynamics of the environment. There will always be a trade-off between speed of adaption and complexity of computation. In the end, it is the deterioration of the system performance during the learning phase that stops us from employing too much learning rather than the complexity of any algorithms.

Future works and extensions: In both SFs, the learning strategy applied the simple approach of trying all the available actions a number of times and thereafter indefinitely applying the learned best action. Although, its convergence was improved by the combination of distributed exploration and centralized cooperative learning, some other variant of the approach would in practice be required since in a large network, cells operate in diverse environments that require specific handling. It may also be necessary to find methods through which states and actions can be automatically derived. This would reduce the design time and possibly the human subjectivity that is typically included in the solutions. On the contrary human intelligence could improve the performance, e.g., by limiting the applicable learning-time parameter space to a range known to have good performance.

In general, however, the results presented here confirm that the cognitive cellular network approach, and specifically QL, provides an outstanding approach to developing SON solutions especially where the dependence of metrics to the control parameters is not perfectly known.

References

[1] 3GPP, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and Self-Optimizing Network (SON) use cases and solutions," 3GPP, Technical Report 36.902 version 9.3.1 Release 9, May 2011. [Online]. Available: http://www.3gpp.org

[2] J. Alonso-Rubio, "Self-optimization for handover oscillation control in lte," in *IEEE/IFIP Network Operations and Management Symposium, Osaka, Japan,*, 2010.

[3] I. Balan, T. Jansen, B. Sas, I. Moerman, and T. Kürner, "Enhanced weighted performance based handover optimization in LTE," in *Proceedings of FNMS, Warsaw, Poland,*, 2011.

[4] G. Hui and P. Legg, "Soft metric assisted mobility robustness optimization in lte networks." Paris, France: International Symposium on Wireless Communication Systems (ISWCS 2012), 2012, pp. 1–5.

[5] V. Capdevielle, A. F. A., and A. Fakhreddine, "Self-optimization of handover parameters in lte networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2013 11th International Symposium on*, May 2013, pp. 133 – 139.

[6] H. Eckhardt, S. Klein, and M. Gruber, "Vertical antenna tilt opti-mization for lte base stations," in *In Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, may 2011, pp. 1–5.

[7] A. Gerdenitsch, S. Jakl, Y. Chong, , and M. Toeltsch, "A rule-based algorithm for common pilot channel and antenna tilt optimization in umts fdd networks," in *ETRI journal*, 2004, pp. 437–442.

[8] M. Garcia-Lozano, S. Ruiz, , and J. Olmos, "Umts optimum cell load balancing for inhomogeneous traffic patterns," in *In Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th, Vol. 2*, sept 2004, pp. 909 – 913.

[9] L. Du, J. Bigham, L. Cuthbert, C. Parini, , and P. Nahi, "Using dynamic sector antenna tilting control for load balancing in cellular mo-bile communications," in *In International Conference on Telecommunications, ICT2002, Beijing, Citeseer*, 2002, pp. 344– 348.

[10] M. N. 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*. IEEE, 2012, pp. 2818–2823.

[11] M. Dirani and Z. Altman, "A cooperative reinforcement learning approach for inter-cell interference coordination in ofdma cellular networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*. IEEE, 2010, pp. 170–176.

[12] M. Bennis and D. Niyato, "A q-learning based approach to interference avoidance in self-organized femtocell networks," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 706–710.

[13] S. Mwanje and A. Mitschele-Thiel, "Distributed cooperative Q-learning for mobility-sensitive handover optimization in LTE SON," in *Computers and Communication (ISCC), IEEE Symposium on*, 2014, pp. 1–6.

[14] S. S. Mwanje and A. Mitschele-Thiel, "A q-learning strategy for lte mobility load balancing." in *PIMRC*, 2013, pp. 2154–2158.

[15] C. Watkins, "Learning from delayed rewards," PhD Thesis, Kings College, Cambridge, England, May 1989.

[16] R. Sutton, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[17] C. J. Watkins and P. Dayan, "Q,-learning," *MachineLearning*, vol. 8, pp. 279–292, 1992.

[18] E. Even-Dar, S. Mannor, and Y. Mansour, "Pac bounds for multi-armed bandit and markov decision processes," in *in Proceedings of 15th Conference on Computational Learning Theory*, July 2002, pp. 258–259.

[19] R. Razavi, S. Klein, and H. Claussen, "Self-optimization of capacity and coverage in lte networks using a fuzzy reinforcement learning approach," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*. IEEE, 2010, pp. 1865–1870.

[20] P. Munoz, R. Barco, I. de la Bandera, M. Toril, and S. Luna-Rami?rez, "Optimization of a fuzzy logic controller for handover-based load balancing," in *Vehicular technology conference (VTC Spring), 2011 IEEE 73rd*. IEEE, 2011, pp. 1–5.

[21] P. Munoz, R. Barco, J. M. Ruiz-Aviles, I. de la Bandera, and A. Aguilar, "Fuzzy rule-based reinforcement learning for load balancing techniques in enterprise lte femtocells," *Vehicular Technology, IEEE Transactions on*, vol. 62, no. 5, pp. 1962–1973, 2013.

[22] N. Sinclair, D. Harle, I. A. Glover, J. Irvine, and R. C. Atkinson, "An advanced som algorithm applied to handover management within lte," *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 62, NO. 5*, pp. 1883–1894, JUNE 2013.

[23] Institute of Communication Networks and Computer Engineering (IKR), "Ikr simulation and emulation library," Universitt Stuttgart, Tech. Rep., Sept 2013, v.4.0. [Online]. Available: http://www.ikr.uni-stuttgart.de/en/Content/IKRSimLib/

[24] 3GPP, "Physical layer aspect for evolved universal terrestrial radio access (utran)," 3GPP TR 25.814, Tech. Rep., October 2006. [Online]. Available: http://www.3gpp.org

[25] A. M. Rao, A. Weber, S. Gollamudi, , and R. Soni, "Lte and hspa+: Revolutionary and evolutionary solutions for global mobile broadband," *Bell Labs Technical Journal*, vol. 13, no. 4, p. 7?34, 2009.

[26] 3GPP, "E-UTRA Radio Resource Control (RRC) Protocol specification (Release 8)," 3GPP, Technical Report 36.331 V8.16.0, December 2011.

[27] T. Jansen, I. Balany, I. Moermanz, and T. Kürner, "Handover parameter optimization in lte self-organizing networks," in *COST2100 TD(10)10068, Athens, Greece,*, 2010, p. February.

[28] T. Jansen, I. Balan, S. Stefanski, I. Moerman, and T. Kurner, "Weighted performance based handover parameter optimization in lte," in *Vehicular Technology Conference (VTC Spring), IEEE 73rd*. IEEE, 2011, pp. 1–5.

[29] I. Balan, B. Sas, T. Jansen, I. Moerman, K. Spaey, and P. Demeester, "An enhanced weighted performance-based handover parameter optimization algorithm for lte networks," *EURASIP Journal on Wireless Communications and Networking*, p. 98, 2011.

[30] S. Mwanje, N. Zia, and A. Mitschele-Thiel, "Self organised handover parameter configuration for lte," in *International Symposium on Wireless Communication Systems (ISWCS 2012), Paris France*, August 2012, pp. 26–30.

[31] G. Hui and P. Legg, "Soft metric assisted mobility robustness optimization in lte networks," in *9th International Symposium on Wireless Communication Systems (ISWCS) Paris France*, August 2012, pp. 1–5.

[32] J. Lorca and A. Sierra, "A Simple Speed Estimation Algorithm for Mobility-Aware SON RRM Strategies in LTE ," in *6th IFIP/IEEE Wireless Days Conference (WD'13)*. Valencia, Spain: Wireless Days 2013, 2013.

[33] H. Wang, L. Ding, P. Wu, Z. Pan, N. Liu, and X. You, "Dynamic load balancing and throughput optimization in 3gpp lte networks," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*. ACM, 2010, pp. 939–943.

[34] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan, "Load balancing in downlink lte self-optimizing networks," in *VTC 2010-Spring, Taipei, Taiwan*, May 2010.

[35] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *AAAI-93: Proceedings of the Eleventh National Conference on Artificial Intelligence*, vol. 11. AAAI Press, 1993, p. 99.

[36] M. Kearns and S. Singh, "Finite-sample convergence rates for q-learning and indirect algorithms," *Advances in neural information processing systems*, pp. 996–1002, 1999.

Stephen S. Mwanje is a Senior Research Engineer at Nokia where is he working on network management automation with a special focus on small cell ultra dense networks. He received his Dr. Ing. degree with the Integrated Communications Research Group at the Ilmenau University of Technology, in Germany, where he worked on multi agent coordination of cognitive self organizing network functions. He holds a BSc Electrical Engineering from Makerere University in Kampala, Uganda and A MS Electrical Engineering from the University of Rochester in Rochester, NY. He has has more than 8 years of mobile network operations, where he managed various projects on GSM/GPRS network planning, deployment, and optimization; microwave radio engineering and spectrum/interference management as well as fiber optic network planning, deployment and operations.

Lars Christoph Schmelz is a Senior Research Engineer at Nokia working on cognitive network management and operation topics. He received his Dipl.-Ing. (M.Sc.) degree in Electrical Engineering / Communication Technologies from Technische Universitaet Muenchen, Germany, in 2000. He worked as a Research Engineer at Siemens Information & Communication Mobile and later Nokia Siemens Networks since 2001, focusing on topics such as mobile network and mobile core evolution, IP-based radio access network, network management architecture and automation, and self-organizing networks (SON). He contributed to multiple European Union funded projects including the EU FP7 SOCRATES and SEMAFOUR projects where he took the lead of work packages on self-configuration / self-healing, and integrated SON management. In SEMAFOUR he furthermore had the role as technical project manager.

Andreas Mitschele-Thiel is a full professor at the Ilmenau University of Technology, Germany, and head of the Integrated Communication Systems group. 2005 to 2009, he also served as Dean for the Faculty for Computer Science and Automation. In addition, he is co-founder of two research spin-offs of the university.

He received a Diploma in Computer Engineering from the Fachhochschule Esslingen in 1985, a M.S. in Computer and Information Science from The Ohio State University in 1989 and a Doctoral degree in Computer Science from the University of Erlangen in 1994. He completed his habilitation in Computer Science at the University of Erlangen in 2000. 1996 to 2000 he served as head of the research group on Design Methodologies at the University of Erlangen. During this period he also served as Associated Rapporteur at the International Telecommunication Union (ITU) where he was responsible for Time and Performance in SDL.

Andreas Mitschele-Thiel has held various positions in the telecommunication industry. 2000 to 2003 he was with the Wireless Advanced Technologies Lab of Lucent Technologies. At Lucent he served as project manager for the IPonAir project, focusing on the evolution of 3G systems. Prior to this he has held various positions at Alcatel. His research focuses on the engineering of telecommunication systems. Special interests are in mobile communication networks, especially self-organization in next generation mobile networks, and IP-based mobile communication systems. His research experience includes communication networks, software engineering, performance evaluation and optimization, and formal description techniques. He has authored two books and numerous papers mostly in the area of communication systems.