STS: Space-Time Scheduling for Coordinating Self-Organization Network Functions in LTE

Stephen S. Mwanje, Andreas Mitschele-Thiel
Integrated Communications Systems
Techniche Universität Ilmenau
Email: {stephen.mwanje, andreas.mitschele-thiel}@tu-ilmenau.de

***Abstract*—Self-Organizing Networks (SON) and a number of SO functions (SFs) have been proposed, e.g. in the LTE standard. Since SFs operate on the same network, adjusting the same set of parameters, conflicts arise. Mechanisms are thus required to resolve or minimize these conflicts. We propose Space-Time scheduling procedures that allow for separating the execution of SFs at different space and time points so as to minimize negative cross effects among the SFs. Using two Q-learning based SFs, our results show that the combined scheduling in space and time ensures that SFs learn optimal behaviors that are only due to their own actions and not the peers' actions. In doing so, they maintain good performance even within the shared environment.**

Keywords- LTE; SON; Coordination; Conflicts;

I. INTRODUCTION

The number and density of cellular base stations have increased as a result of increase in desired user throughput and the subsequent the network traffic. This has resulted in increases in networks' capital and operational expenses as well as their complexity of operation. To handle these challenges, SON has been proposed ([1] [2]) and a number of SFs defined, e.g. in the LTE SON standard [3] [4]. A SF in this case is any network function that can be automated, e.g. Mobility Robustness Optimization (MRO) or Mobility Load balancing (MLB). Since SFs operate on the same network adjusting the same or related parameters, conflicts arise during their operation. Mechanisms are thus required to resolve or minimize these conflicts.

In [5] and [6] two ideas were presented for managing SON conflicts. The first suggests classifying SFs using Functional Parameter Groups, created in a way that parameters in any one group contribute to the satisfaction of the same goal(s) and are decoupled from other groups. However, most parameters were found to fall under the same group which would lead to impractical results, as many SFs would need to be implemented together and possibly concurrently. The second idea introduced a control plane to decide the activation of triggers and a coordination plane to process parameter changes proposed by concurrent SFs. However, the need for control and coordination rules for every pair of SFs renders the solution impractical for a large set of SFs. Meanwhile, a coordination function that separates SFs in time so as to minimize race conditions is presented in [7]. However, deactivating an SF *i* throughout the network at all execution times of the other SF(s) is not acceptable as it might lead to sub optimal performance in *i*.

In this paper we present Space-Time Scheduling (STS) procedures that separate the execution of different SFs in space and time so as to minimize the cross effects among SFs. We evaluate the performance of the STS procedures using two Q-learning (QL) based SON functions published in [8] and [9].

We begin with a discussion of the QL SFs in Section II and describe our proposed approaches in Section III. In Section IV, we describe the simulation scenario, the results both with and without coordination and the envisaged performance limits. We conclude with a summary and research outlook in section V.

II. Q-LEARNING SON FUNCTIONS

As shown in Fig. 1, each SF acts as a control agent that: 1) observes the network to evaluate its trigger, 2) takes an action and 3) gets feedback on the effect of that action at the end of a monitoring period called the SON interval. The action is the adjustment or configuration of the associated parameters. Meanwhile the feedback measures how well or badly the target metrics have been affected by the action. A rule-based SF determines the next action based on defined rules. On the other hand, a learning-based SF explores the actions, using the feedback to learn the effect of each action. The two SFs considered in our study are the Q-learning functions for MLB (QLB) [8] and for MRO (QMRO) [9].

Q-Learning (QL) is a model free Reinforcement learning technique which, using Temporal Difference (TD) methods can solve learning problems that do not have explicit behavioral models. To do so, QL estimates a value function Q, for each state-action pair as the expected reward of taking an action $a \in A$ when starting in state x and thereafter acting according to a fixed policy π. Both QMRO and QLB require actions that achieve instantaneous results on the set objectives as explained in the next sections. Each solution maintains a Q-table whose entries are the Q-values for the corresponding state-action pairs. Each Q at time i+1 is estimated iteratively using the TD update method [8] modified for instantaneous rewards as

$$Q_{i+1}(x_i, a_i) = (1 - \beta)Q_i(x_i, a_i) + \beta[r_i(x_i, a_i)] \quad (1)$$

The learning rate β in range [0,1] defines the influence of new information on previous knowledge. β=0 implies no learning while β=1, means considering only the latest information.

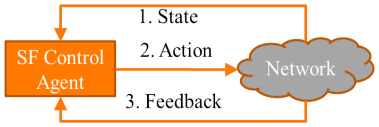


Fig. 1. Abstract SON Controller

A. HO Modeling

Both MRO and MLB rely on the handover (HO) procedure. *HOs* are triggered using the A3 event [10] which for HO from serving cell *s* to the target cell *t* is

$$F_t + O_t^s - Hys > F_s + O_s^t. \quad (2)$$

F_t,F_s are respectively the Reference Signal Receive Power (RSRP) in dBm of the *t* and *s* cells, without any offsets. O_t^s, O_s^t are the Cell Individual Offsets (CIO) in dB respectively for HO from *t* to *s* and from *s* to *t*. Hys is the hysteresis in dB which is uniform for the serving cell. 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.

B. QMRO: Q-Learning for MRO

QMRO learns the best Hys-TTT configuration for a given mobility state in a cell. Since the mobility state observed in one cell is likely to re-occur in another, the cells learn cooperatively by sharing the Q-table. The elements of QMRO are [9]:

1) Performance metrics: MRO aims to minimize Radio Link Failure (RLF) rates without increasing the Ping-pong (PP) rate. A RLF occurs if the SINR stays below a threshold for the critical time T310 [10]. The RLF rate (F), either due to too late HOs (F_L) or due to too early HOs (F_E), is the rate of RLFs per second, calculated either for the cell or the entire network. On the other hand, a PP occurs when a successful HO from a cell A to B is followed by another successful HO from B back to A in a time less than PP time. Then the PP rate (P) is the number of PPs per second in the cell or the network. We evaluate HO performance in terms of a HO Aggregate Performance (HOAP) metric defined as

$$HOAP = w_1 P + w_2 F_E + w_3 F_L; \qquad \sum(w_i) = 1 \quad (3)$$

We select the weights w_i to equally balance effects of early HOs (P and F_E) against effects of late HO (F_L). Since for positive Hys, $F_E \approx 0$, we set $w_1 > w_2$ so that early HOs are not disproportionately favored. The applied weight vector is thus $w = (0.3, 0.2, 0.5)$. Note that this selection is subjective and could be done differently but we assume in general that such a definition would be the input of operator policy.

2) State-action space: The states capture the average velocity in the cell, which here are defined by the states in Table I [9]. An action is the (Hys-TTT) tuple each cell signals to all its users in any state. Specifically, we consider the 49 tuples defined by Hys = (0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0) in dB and TTT = (100, 128, 256, 320, 480, 512, 640) in msec.

3) Rewards: The reward function aims to minimize RLFs, without increase in PPs and HOs. The reward $r_{x,i}$ is thus the negative HOAP in (3) evaluated over the SON interval and applying the same weight vector.

$$r_{x,t} = -(w_1 P + w_2 F_E + w_3 F_L); \quad (4)$$

TABLE I. *QMRO* MOBILITY STATES

Average Velocity (kmph)	0-4	4-8	8-12	12-17	17-22	22-28	28-34	34-41	41-48	48-56	56-65	65-75	75+
State (x)	0	1	2	3	4	5	6	7	8	9	10	11	12

C. QLB: Q-Learning for MLB

MLB seeks to move users from an overloaded cell to a set T of less loaded neighbor cells (the target cells). From (2), if O_t^s is positive and/or O_s^t is negative, the HO condition is fulfilled in advance. Then s to t HOs will be executed earlier, effectively forcing users out of the serving cell.

Assume a cell c has U users and user $u \in U$ requires $X_{c,u}$ Physical Resource Blocks (PRBs). With PRB bandwidth B_{PRB} (=180 KHz [10]) and c bandwidth B_{cell}, the cell load ρ is

$$\rho = \frac{\sum_{u \in U} X_{c,u}}{B_{cell}/B_{PRB}}. \quad (5)$$

ρ can be greater than 1, representing the case when the total required PRBs exceed the maximum possible PRBs within B_{cell}, but c is overloaded when ρ exceeds a threshold ρ_{max}.

The LB algorithm adjusts the generic boundary between the s and t cells by changing the *CIOs* according to

$$\begin{matrix} {O_t}^s = {O_t}^s - \phi \\ {O_s}^t = {O_s}^t + \phi \end{matrix} \quad all\ t \in T. \quad (6)$$

The size of the change in boundary (i.e. the size of ϕ) depends on the load status in the serving cell and its neighbors. As such, different ϕ values need to be learned for the different load conditions. Moreover because the load states can reappear in any cell in the network, QLB also applies cooperative learning among the cells. The elements of the QLB algorithm are [8]:

1) Performance metrics: When the cell is overloaded, users are unsatisfied, i.e. they are allocated fewer PRBs resulting in lower data rates than expected. Due to scheduling variations, a user is considered unsatisfied (an un-satisfaction event occurs) only if it's total achieved data rate in a continuous 1 second period is less than the Guaranteed Bit rate (GBR). We evaluate performance in terms of the Number of unsatisfied users (Nus), which is the average number of un-satisfaction events in the cell/network within the evaluation period.

2) State-actions space: The size of ϕ in (6) depends on: ρ_s - the load in the serving cell s; ρ_n - the average load in the T-cells; and uD - the user distribution in cell s. Each state is thus a vector [ρ_s ρ_n uD], with entries discretized as described in [8]. Actions are the possible values that ϕ can take, which were selected as the set [0.2, 0.4, 0.6, 0.8, 1.0] in dB.

3) Rewards: QLB aims to determine the action ϕ that instantaneously removes overload from the serving cell, but without overloading the target cell(s). As such the rewards, as set in Table II consider δ_{ρ_s} the achieved reduction in ρ_s and the extra load created in neighbor-cells. Positive δ_{ρ_s} (reduction in ρ_s) is rewarded while the reverse is penalized. Since ρ_n is expected to increase, only δ_{ρ_n} of more than 1 is penalized. In general, larger δ_{ρ_s} receive greater reward, but are accompanied

TABLE II. *QLB* REWARD FUNCTION

		Change in serving cell Load δ_{ρ_s}				
		-2	-1	0	1	2+
change in neighborhood load, δ_{ρ_n}	<1	-2	-1	0	1	2
	1+	-3	-2	-1	0	1

Notes:
1. $\delta_{\rho_s} < 0$ represents a reduction in serving cell Load δ_{ρ_s} and vise versa
2. Extra reward is allocated to encourage large steps that do not overload target cells

by penalties for unrestrained actions taken in LB-states with high ρ_n. This allows high load target cells to overload just enough to propagate the load outwards but not so much as to counter-productively cause further dissatisfaction after LB.

III. Proposed SON Coordination Approaches

The individual SON solutions assume that each agent acts separately within the environment i.e. that the observed effects are caused only by its actions and not by any actions of other agents. In reality however, multiple agents act in and observe the same environment, either within a single cell or as neighbor cells. Without coordination, the agents then learn based on effects that are due to their actions as well as the actions of their peers. Coordination provides a mechanism for accounting for each SFs effects on other SFs or a mechanism of separating such effects. In the following sections we describe 3 approaches to scheduling the execution of the SFs so as to minimize these effects: 1) Temporal Separation during Learning (TSL), Concurrent learning with Spatial Separation (CSS), and Space-Time Scheduling (STS).

A. Temporal Separation during Learning (TSL)

If two conflicting SFs concurrently take actions in the same cell, they cannot fully differentiate the effects of their actions on the cell. For this reason a number of studies have proposed separation of the SFs such that at any time only one is allowed to act. However, any SFs will register poor performance during the time when it is deactivated. For example consider that MLB is deactivated to wait for MRO which requires time for at least 1000 HOs to obtain adequate statistics on HO metrics. During this period, even if free resources exist in the neighbor cells, any affected cell will not take any action even if it experiences excessive overload. This is even worse if the SF is deactivated not only in one cell but throughout the whole network.

To counter this, we propose that time separation is executed only during the learning period, and that all SFs are reactivated thereafter. Effectively, only 1 SFs learns at a time. Prioritizing which SF learns first could be an operator policy but it should in general consider which SF is likely to be more affected by the other. For MLB and MRO, since MLB adjusts the HO boundary between cells, it should do so in an environment of optimum HO performance. As such, QMRO is selected to learn first as shown in Fig. 2.

The challenge with TSL is that learning is actually executed concurrently in multiple cells. There is thus a possibility that the effects observed in one cell are not only due to the actions taken by that cell but could also have been contributed by other cells. Moreover this poor performance is made worse when, after learning, the SF acts concurrently with other SFs that are also continuously changing parameters. The alternative approach then is to consider separating SFs in space as we describe in the next section.

B. Concurrent learning with Spatial Separation (CSS)

In neighbor cells, actions of one SF in one cell can also affect a similar or different SF in another cell. For example MRO affects load distribution (MLB) in both the acting cell as well as its neighbors. In that case, the two conflicting SFs (similar or not) should not concurrently take actions in two neighbor cells. Although it is possible that there could be

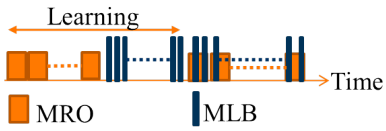


Fig. 2. TSL between QMRO & QLB

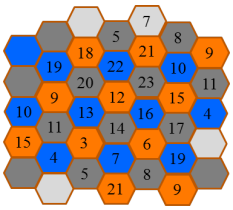


Fig. 3. Cell clustering for spacial scheduling

Algorithm 1:* CSS *Clustering Algorithm

Require: *Seed cell* a; *sets* $X_i, i = 1, 2, 3$ *for the 3 colors*

1. add seed a to X_1
2. Select c = any neighbor of a with ***exactly*** 1 colored neighbor
3. add c to X_2
4. **Repeat while not all cells colored / allocated**
5. **for** each uncolored cell c with ***exactly*** 2 colored neighbors; **do**
6. given neighbors' colors as $X_{i=j}, X_{i=k}$
7. Allocate c to third color $X_{i=l}$

done

some non conflicting SFs, we only consider the generic case in which all SFs are assumed to have degree of conflict however small that may be. Thus, actions are only taken concurrently in two cells, if there is at least 1 other cell in between the two. The result is a reuse-3 concurrency structure among the cells as shown in Fig. 3. To that respect we have designed a clustering scheme that allows one of any 3 neighbor cells to act. We assume that the clustering is executed centrally as part of the network configuration. Essentially each time, a new eNB is added, the clustering scheme of *algorithm 1* is executed to allocate cluster indices (or colors) to the cells. The algorithm requires a seed which can be selected randomly or by the operator. The seed is allocated to the first cluster index and the subsequent cells colored depending on their neighbors colors. In the uniform hexagonal grid as here considered, the clustering scheme achieves the same cell distribution of Fig. 3 among the cells regardless of the applied seed. Each cluster of cells is allocated a time-frame within a single multi-frame as shown in Fig 4. Each cell then schedules SFs within the frame according to its local requirements.

C. Space - Time Scheduling (STS)

The clustering solution above ensures that SFs in any two cells do not conflict with each other. It is however possible that SFs conflict with one another within a single cell, degrading the expected performance. Such degradation is worse in learning functions since the SFs will learn effects which are due to the combination of their actions and their peers actions. To counter the degradations, we combine time separation and special scheduling thus Space-Time scheduling (STS). As earlier proposed, we apply time separation only during the learning. This ensures that SFs learn functions that are only due to their actions and that after learning no SF under performs owing to being deactivated as it waits for other SFs.

Each cluster is allocated a time frame as shown in Fig. 4, such that each cell has 1 out of the 3 frames of the multi-frame within which to take actions. For example, in Fig. 5, cluster 1 cells a and k always take action in frame 1. Within each frame, the cells apply a time division scheduling to allocate time to each of the active SFs. It is imaginable that all cells in a cluster will have the same SF acting. We however, do

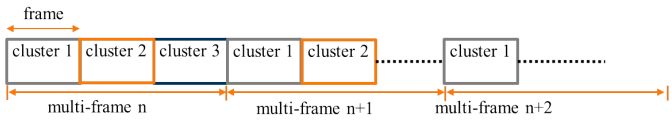


Fig. 4. CSS Multi-frame

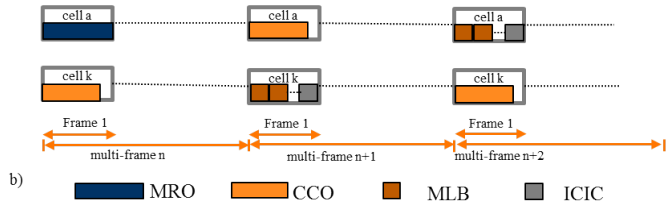


Fig. 5. STS Frames in CSS Multi-frame

not consider this. We instead assume the generic case that cells could have different requirements at different times, so that each cell schedules SFs according to its requirements. As such, each cell will determine how to schedule SFs within its allocated time-slot based on those requirements but also based on the timing requirements of the different functions. For example in Frame 1 in Fig. 5, cells a and k respectively schedule MRO and Coverage-Capacity Optimization (CCO). Contrarily, cell k schedules multiple SFs (e.g. twice MLB and once Inter-Cell Interference Coordination (ICIC)) at a time when cell a schedules CCO.

IV. Performance Results and Discussion

Simulation studies were done using a C++ event based LTE downlink system level simulator based on software libraries provided by Alcatel-Lucent Bell Labs Germany and the University of Stuttgarts Institute of Communication Networks and Computer Engineering [11]. The network scenario consists of 7 eNBs each with 3 cells. A wrap-around is implemented for reliable interference and SINR calculations. Mobile users that are initially randomly placed in the coverage area, move with a random walk mobility model. To implement overload in some cells, a hot-spot area is created by adding a number of fixed users in a cell in the middle of the network. Simulations are repeated over multiple batches, specifically 50 batches each simulating 200 s of operation. In each batch users are re-deployed in the same way i.e. mobile users randomly placed anywhere in the network and fixed users placed in the center cell. All users are connected to the network throughout the simulation interval. In case of a RLF, a user is reconnected to the best available cell as expected for LTE. Further details of the simulation parameters are given in Table II.

First, we consider the scenario where mobile users move at an average velocity of 60 kmph (i.e. individually between 36 and 84 kmph), but consider the effect of speed by evaluating another scenario where users move at 30 kmph. Similarly, we start with a network having a hot-spot of 50 m in radius but also evaluate the network with a smaller hot-spot of 20 m. As mentioned, the performance metric is HOAP for QMRO while QLB is evaluated in terms of the load ρ and the Number of Unsatisfied users ($N_u s$).

A. Performance of Q-learning SON Functions

Fig. 6 to 9 show the performance of the QL functions. We begin with a reference environment to which we apply

TABLE III. Simulation Parameters

Parameter	value
System bandwidth	10 MHz
Inter-site distance	500 m
Time between snapshots	50 ms
Number of users	420 mobile, 40 static
User velocity	variable with mean 30 and 60 Km/h
Mobility Model	random walk
Pathloss	128.1 + 37.6 $log_{10}[max(dKm, 0.035)]$
Shadowing standard deviation,	50m
Decorrelation distance	50m
eNB Tx power	46 dBm
eNB Tx antennas	1 per sector at height = 32 m
UE receive antennas	1 Omni at height = 1.5m
eNB max. antenna gain	15 dBi
UE max. antenna gain	2 dBi
Data rate	512 Kbps

the QL functions, first considering each individually and then evaluating the case when both act concurrently but without coordination. Figs. 6 and 7 evaluate the per-cell averages over an entire batch of 200 s while Figs. 8 and 9 evaluate the instantaneous performance per cell. In particular Fig. 7 attempts to combine the two plots in Fig. 6 to compare the steady state performance after learning has been completed.

1) Reference performance(Ref): The Reference Performance (denoted by Ref in the results) represents the performance of the network when configured with the best manual settings. Since we can not predict the cells that are likely to be in overload, Ref is configured with the default CIO of 0dBs, so that HO triggering depends only on HO parameters - Hys and TTT. The HO settings on the other hand are obtained by manually sweeping the Hys-TTT parameter space at the average velocity of 60 kmph. The best settings, obtained as [Hys=0.5dB, TTT=0.1s], are then applied to the network as the reference settings. This is important since we need to evaluate the performance of QLB based on a network with optimal HO settings. The resulting performance, for this reference scenario that does not apply any SFs, represents the starting point that should be improved by the SFs. In that case, since HO settings are optimal or nearly optimal, QMRO should achieve similar or better performance as Ref. Conversely, QLB should by all means achieve better results as compared to Ref.

2) QMRO performance: QMRO needs to autonomously learn the HO settings that achieve the same or even better HO performance as Ref. We observe in Figs. 6 and 8 that QMRO achieves this. It starts out with poor performance but improves over time such that the steady state performance is comparable to Ref as shown in Fig. 7.

3) QLB performance: QLB aims to redistribute load and minimise user dissatisfaction. We observe in Figs. 6 and 7 that QLB is able to reduce the number of unsatisfied users by almost 50 %. On a micro scale, Fig. 9 evaluates the instantaneous load in 3 cells -the overloaded cell 12, a tier one neighbor 14 and an outer tier two neighbor cel 6. We observe that QLB re-balances the load distribution by transferring the load not only to the neighbors (e.g. cell 14) but eventually to the outer cells (e.g. cell 6).

4) QLH: Uncoordinated operation of QMRO and QMLB: Without coordination we observe that the performance of both SFs degrades as denoted by QLH in Figs. 6 and 7. Effectively, during learning, each SFs observes effects that are due to the combination of its actions and the peer's actions. As such,

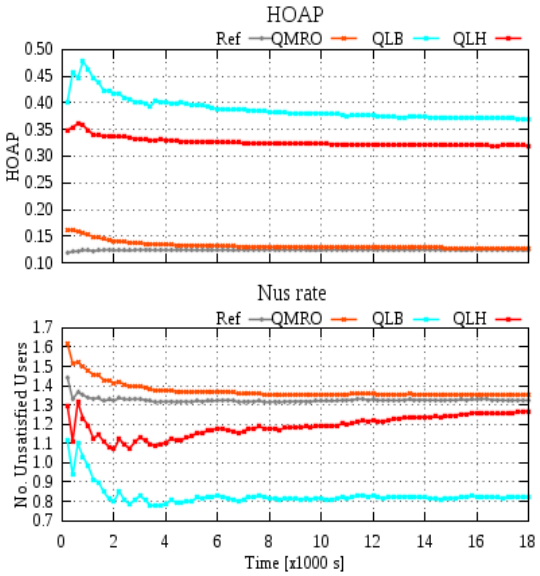


Fig. 6. Performance of the individual and uncoordinated QL SON Functions

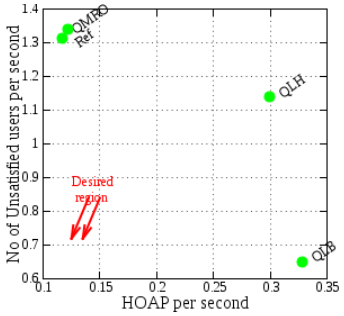


Fig. 7. Steady state performance of QL functions

each SF's action perceived as optimum for any state is actually worse than the one that SF would select when acting alone. The resulting worse performance justifies the need for coordination.

B. Coordinated operation of QLB and QMRO

In Figs. 10 and 11(a), we evaluate the performance for : 1) the independent SFs, QMRO and QLB, when acting alone; 2) the uncoordinated operation QLH ; and 3) the coordinated operation using each of the three proposed approaches.

1) Coordination with TSL: We observe that with TSL there is some improvement in LB performance compared to the case without coordination albeit with more degradation in HO performance. In general, as explained in Section III-A, concurrent action among neighbor cells implies that SFs do not learn independent effects of their actions but also the actions in neighbor cells. Although the agents may still perform well when only one SF is active, the performance then gets degraded when another agent is introduced in the network.

2) Coordination with CSS: By eliminating conflicts among neighbor cells, CSS improves performance for both SFs, even while both SFs run concurrently in each cell. This implies that the cross effects among neighbor cells are actually more important than those among SFs in a single cell. However, Concurrent action among SFs still imposes limitations to the performance gains, since SFs are not learning strictly independent behavior.

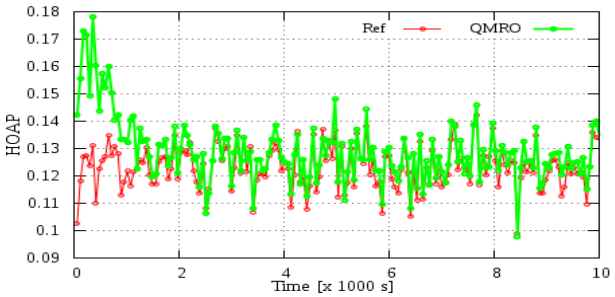


Fig. 8. Instantaneous average HOAP per cell for Ref and QMRO

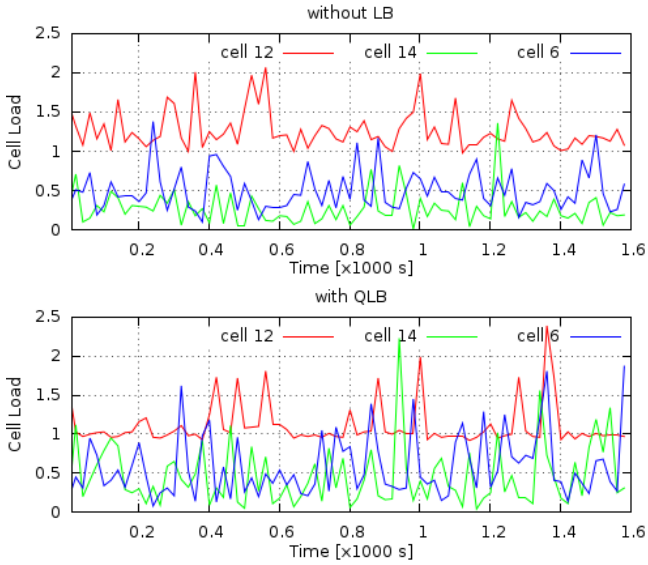


Fig. 9. QLB instantaneous performance: Load variation in 3 selected cells

3) Coordination with STS: By separating both in space and time, STS enables SFs to learn the strictly independent behavior that is free of peers' actions. This results in better performance compared to TSL and CSS. However, because during operation each SF actions are superimposed onto peers' actions, the performance gains will still not be equivalent to that when the SF acts independently in the network. In general

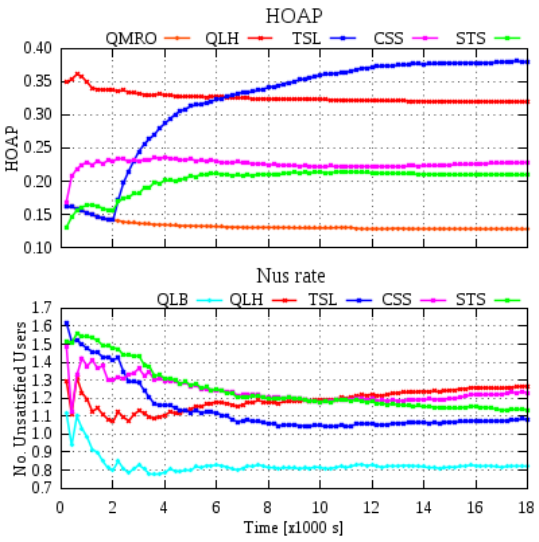
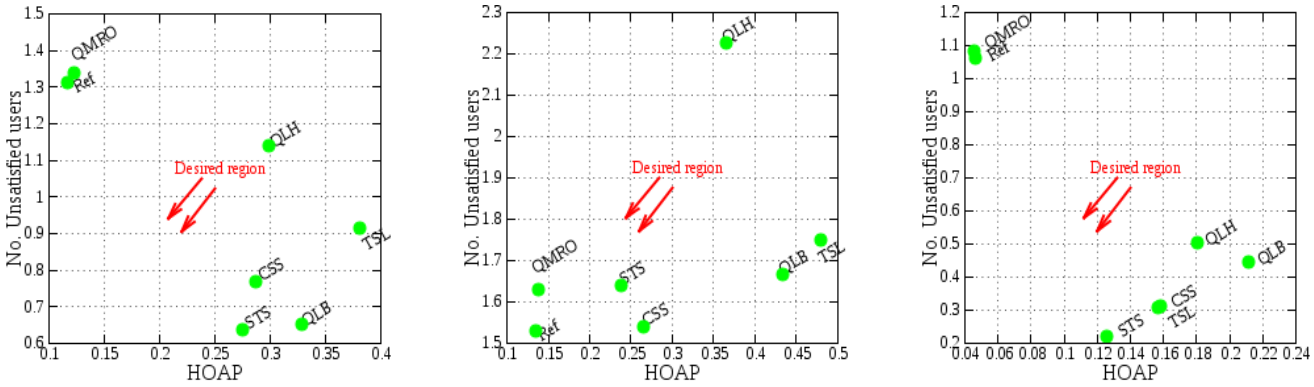


Fig. 10. Performance of the QL SFs without and with uncoordinated



a) $50m$ hotspot, $60kmph$ b) $20m$ hotspot, $60kmph$ c) $50m$ hotspot, $30kmph$

Fig. 11. SFs Performance results for hotspot sizes and mobile velocities respectively

however STS achieves the best performance where the multiple SFs all need to act concurrently in the network.

It is tempting to assume that some solutions achieve the same performance owing to the closeness of the lines in Fig. 10. It is evident however from the two dimensional plot in Fig. 11(a) that no two solutions can always achieve the same results in both dimensions. Besides the performance is also highly dependent on the operational scenario as described in the next section.

C. Discussion on performance limits

1) Scenario dependent performance: We note that the observed performance in not always fixed but will in general depend on the specific scenario in the network. For example in case of a small hotspot (like 20m radius in Fig. 11b), the QLB gain may be insignificant but owing to extreme degredation, coordination still achieves good results Similarly if the hotspot occurs in a lower velocity environment (say 30 kmph in Fig. 11c), coordination may achieve even more reduction in the number of unsatisfied users by optimally handing over users to cells where they have better SINR conditions.

2) Sizing the frames: Caution is required to ensure that SFs are allocated the appropriate lengths of time to execute appropriately. As such the frame periods should be designed from the inside outwards; first by establishing the SF requiring the longest interval so that the frame length is greater or equal to this interval. Shorter SFs would then either use part of the frame with some time remaining unused or would be cascaded one after another if their combined time can fit within the frame (e.g. cell k in multi frame n+1 in Fig. 5).

V. CONCLUSION AND OUTLOOK

In this work, we have presented our proposed Space-Time scheduling procedures aimed at minimizing negative cross effects among SON functions. Considering 2 Q-learning based SON functions, our results justified and quantified the degree of conflict that arises when SFs are not coordinated, and showed that spatial scheduling is able to achieve good compromise performance. We then showed that combined scheduling in space and time not only achieves the best performance but will in some cases achieve better results than those achieved when SFs act alone in the network. We concluded with a discussion highlighting the potential limits to the performance gains and the dependency of the absolute gains on the specific scenario in the network. Future research activities will focus on generalizing the proposed approaches for multiple SON functions. We intend to evaluate the best approach (STS) in an environment implementing more than 2 SFs to prove that it can easily be scaled to any number of SFs.

ACKNOWLEDGMENT

The authors wish to thank E. Kühn and S. Klein of Alcatel-Lucent Bell Labs, Germany for support with the simulator libraries. This work has been financed by DAAD and DFG.

REFERENCES

[1] NGNM, "Use cases related to self organising network, overall description," May 2007. [Online]. Available: http://www.ngmn.org/technology.html

[2] SOCRATES, "Self-optimisation and self-configuration in wireless networks." [Online]. Available: http://www.fp7-socrates.org/

[3] 3GPP, "Evolved universal terrestrial radio access network (e-utran); self-configuration and self-optimizing network use cases and solutions," TR 36.902 V0.0.1, Tech. Rep., December 2009. [Online]. Available: http://www.3gpp.org

[4] SOCRATES, "Deliverable d2.1: Use cases for self-organising networks, eu strep socrates," EU STREP SOCRATES (INFSO-ICT-216284), Tech. Rep., March 2008. [Online]. Available: http://www.fp7-socrates.org/?q=node/10

[5] T. Jansen, M. Amirijoo, U. Türke, L. Jorguseski, K. Zetterberg, R. Nascimento, L. C. Schmelz, J. Turk, and I. Balan, "Embedding multiple self-organisation functionalities in future radio access networks," in *proc. 69th VTC, Barcelona, Spain*, 2009.

[6] SOCRATES, "Deliverable d5.9: Final report on self-organisation and its implications in wireless access networks," EU STREP SOCRATES (INFSO-ICT-216284), Tech. Rep., December 2010.

[7] K. Tsagkaris, N. Koutsouris, P. Demestichas, R. Combes, and Z. Altman, "Son coordination in a unified management framework," in *proc. 77th VTC, Dresden, Germany*, 2013.

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

[9] S. Mwanje and A. Mitschele-Thiel, "Cooperative q-learning for lte self-organized handover optimization," in *IEEE 19th International Symposium on Computing and Communications (ISCC 2014)*, Madeira, Portugal, May 2014.

[10] 3GPP, "E-utra radio resource control (rrc) protocol specification (release 8)," 3GPP TS 36.331 V8.16.0, Tech. Rep., December 2011.

[11] Institute of Communication Networks and Computer Engineering (IKR), "Ikr simulation and emulation library." [Online]. Available: http://www.ikr.uni-stuttgart.de/en/Content/IKRSimLib/