Reinforcement Learning Strategies for Self-Organized Coverage and Capacity Optimization

Muhammad Naseer ul Islam, Andreas Mitschele-Thiel
Ilmenau University of Technology
Ilmenau, Germany
{muhammad.naseer, mitsch}@tu-ilmenau.de

***Abstract*—Traditional manual procedures for Coverage and Capacity Optimization are complex and time consuming due to the increasing complexity of cellular networks. This paper presents reinforcement learning strategies for self-organized coverage and capacity optimization through antenna downtilt adaptation. We analyze different learning strategies for a Fuzzy Q-Learning based solution in order to have a fully autonomous optimization process. The learning behavior of these strategies is presented in terms of their learning speed and convergence to the optimal settings. Simultaneous actions by different cells of the network have a great impact on this learning behavior. Therefore, we study a stable strategy where only one cell can take an action per network snapshot as well as a more dynamic strategy where all the cells take simultaneous actions in every snapshot. We also propose a cluster based strategy that tries to combine the benefits of both. The performance is evaluated in all three different network states, i.e. deployment, normal operation and cell outage. The simulation results show that the proposed cluster based strategy is much faster to learn the optimal configuration than one-cell-per-snapshot and can also perform better than the all-cells-per-snapshot strategy due to better convergence capabilities.**

Keywords-antenna downtilt; self-organization; fuzzy logic; re-inforcement learning; fuzzy q-learning; LTE

I. INTRODUCTION

In order to simplify cellular network design and optimization processes, a strong research interest is developing to integrate machine intelligence into cellular networks. In this regard, the principles of *Self-Organizing Network (SON)* have got the attention of industry and academia alike. Main drivers for *Self-Organization (SO)* in cellular networks include the reduction of operational expenditure (OPEX) by minimizing the manual human effort in all phases of network lifetime, i.e. deployment, optimization and operation. Additionally, SONs also promise capital expenditure (CAPEX) reduction by optimizing the use of network nodes and thus making it possible to delaythe network capacity upgrades [1].The Next Generation Mobile Networks (NGMN) consortium of cellular network operators has formalized the requirements of SO in the form of several use cases for self-configuration, self-optimization and self-healing of cellular networks [2]. For these reasons, SO is now a part of a new cellular system standardization known as Long Term Evolution (LTE) [3]. And 3GPP the standardization body of LTE is working to define the concepts, requirements and solutions of different SO use cases for LTE.

One of these SO use cases is Coverage and Capacity Optimization (CCO) that aims to provide the required capacity in the targeted coverage areas, minimize the interference and maintain an acceptable quality of service in an autonomous way. To achieve these targets, antenna configuration especially the antenna downtilt plays a critical role. As antenna downtilt affects the direction of the antenna radiation pattern, it can be used to improve the received signal strength in the own cell as well as to reduce the interference to neighboring cells. This is especially important for re-use one based cellular networks like LTE that uses the same frequency band in all the cells. Antenna downtilt can be adapted both mechanically and electrically [5]. Electrically it can be done by shifting the signal phases of different antenna elements and is more suitable for frequent adaptations.

Antenna downtilt is widely studied because of its effectiveness in CCO. In [4] the benefits of dynamic downtilt adaptation compared to fixed network wide configuration were presented. To realize this dynamic adaptation, some approaches with heuristic algorithms were also presented in [5]-[7]. This can also be referred to an automation process of network planning tools, which tests different network configurations in an offline manner and then adapts the network according to the best solution. On the other hand, some distributed approaches [8] and [9] try to simplify the optimization problem by performing only local optimization within a cluster of cells instead of the whole network at once. Some studies [10] and [11] also propose simple rule based schemes for online optimization of downtilt in an operational network. They adapt the downtilt according to the network state based on some pre-defined rules. This on one hand requires human expertise to design these rules and on the other hand makes the optimization process relatively inflexible. The design process also becomes quite complex as realistic assumptions about the network and its environment have to be taken into account. Finally, in [12] a *Fuzzy Q-learning (FQL)* approach is presented for downtilt optimization. It combines fuzzy logic with q-learning to learn the control structure during its operation. However, they restrict to a stable strategy of only one cell update per snapshot. This would significantly slow down the learning process as the network size grows.

In this paper, we present the potential of different learning strategies in an *FQL* based solution for downtilt optimization. Instead of allowing only one cell per network snapshot to update its downtilt and learn from the environment [12], we allow simultaneous updates of all cells. This helps to speed-up

This research is supported by the German Research Foundation (DFG) and Alcatel-Lucent Bell Labs, Germany.

the learning process but makes it more complicated as the impact of each cell's action on the environment also depends upon the actions of other cells. This environmental impact is crucial to define the feedback for the cell in order to identify whether, the action was good or bad. Therefore, in the presence of combined actions the characterization of individual feedback becomes complex. We study the speed and convergence properties of both of these strategies and also propose a clustering mechanism which combines the benefits of both. It is faster as the simultaneous update of all cells and convergent as the one cell per snapshot approach.

The rest of the paper is organized as follows: Section II describes the downtilt optimization as a Multi-Agent Reinforcement Learning problem and its solution using Fuzzy Q-Learning (FQL). Section III then describes the different learning strategies. The simulation environment used for performance evaluation is explained in Section IV. The performance results are presented in Section V. Finally, Section VI concludes the paper.

II. FUZZY Q-LEARNING APPROACH

Reinforcement learning (RL) is a type of machine learning technique that exploits the learning agent's experience in order to learn the optimal behavior in an environment [13]. Through its interaction with the environment, the agent tries to learn actions for particular states of the system so that the long-term rewards are maximized. The learning problems are usually modeled as Markov Decision Processes (MDP) and solved using Dynamic Programming techniques. These techniques require very accurate system models in terms of state transition probabilities and their corresponding rewards. Q-Learning (QL) is a special type of RL that can solve problems where this system model is not available. Instead, it relies on the Temporal Difference (TD) method to incrementally solve the learning problem. QL achieves this goal by estimating a value function of each state-action pair known as the *Q-Value*. This value function estimates the expected reward of taking an action $a \in A$ out of all possible actions when starting from state s and following a fixed policy π afterwards. Each action transits the agent from state s_t to s_{t+1} and receives a reward r_{t+1} for that, as shown in Fig. 1. The target is to maximize the overall reward. The value function is defined as

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r(s_t,a_t) \mid s_0 = s,\ a_0 = a\right] \quad \textbf{(1)}$$

and can be estimated using the TD update method in an iterative way [13]:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \beta(s_t,a_t)\left[r_{t+1} + \gamma \underset{a'}{Max} Q_t(s_{t+1},a') - Q_t(s_t,a_t)\right] \quad \textbf{(2)}$$

β is the learning rate within the interval [0,1] and defines the influence of new information on the previous knowledge. A value of 0 means no learning and 1 means only the latest information is considered. γ is the discount factor within the range [0,1). The smaller the value of γ the more importance is given to the present reward as compared to the long-term reward.

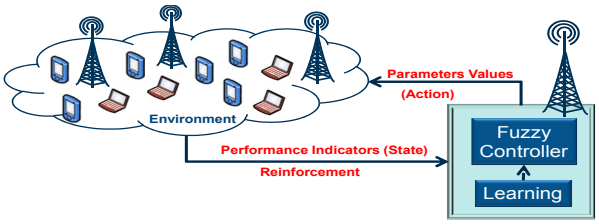


Figure1: Fuzzy Q-Learning Controller (FQLC)

However, as QL has to maintain a Q-Value for each state action pair, therefore it becomes quite complex if the state or action space is continuous. Here Fuzzy Logic (FL) can be used to discretize these continuous variables. Fuzzy Q-Learning (FQL) is such a technique that combines fuzzy logic with Q-learning in order to overcome the shortcomings of each [14] .In FQL, system states and actions are defined using fuzzy membership functions. The continuous domain state variables are first transformed into a finite number of fuzzy variable membership functions. In fuzzy logic terminology this process is known as *Fuzzification*. From these fuzzy variables the corresponding output is calculated based on the *Fuzzy Inference System (FIS)*. Finally the fuzzy output of the FIS is mapped back to the continuous domain output variable through the process of *Defuzzification*.

A. Fuzzy Q-Learning Controller (FQLC) Components

The basic components of our FQLC for coverage and capacity optimization (CCO) are described below

1) State and Action Fuzzy Variables

Cell states can be measured in terms of several KPI statistics, which gives an indication of how the system is performing currently. For CCO, we use the spectral efficiency as the performance metric as it directly reflects how efficiently the available spectrum is utilized. Therefore, the input state vector is defined as:

$$s = \begin{bmatrix} DT & SE^{center} & SE^{edge} \end{bmatrix} \quad \textbf{(3)}$$

where, DT is the current downtilt value of the cell. SE^{center} is the mean spectral efficiency measured for the cell. And SE^{edge} is the edge spectral efficiency measured from the lower 5-percentile of the mobile SINR reports in the cell. This helps to ensure that the performance not only improves on average but also at the cell borders.

As we focus on antenna downtilt adaptation for CCO, the possible actions for a cell are the changes that can be applied to the current tilt. For fuzzification we define a finite number of fuzzy labels over the domain of each input and output variable as shown in Fig. 2. The downtilt and output have five labels, whereas, the SE^{center} and SE^{edge} have three labels each. Finally,each label is assigned a membership function that maps the degree of membership of a particular fuzzy variable to each label within the real interval [0,1]. We use strict triangular membership functions so that the sum of membership values of all the functions equals 1 at any point over the domain of a specific variable.

2) Rule Based Fuzzy Inference System

Fuzzy Inference System (FIS) defines the mapping from the input states to the output actions.Typically FIS consists of rules defined by different combinations of the fuzzy variable membership functions. A typical FIS rule in Fuzzy Q-Learning is initialized as [14]:

$$IF\ s^1\ is\ L_i^1\ and\ s^2\ is\ L_i^2\ and \ldots and\ s^n\ is\ L_i^n$$
$$THEN\ y = o_i^1\ \ with\ q(L_i, o_i^1) = 0$$
$$or\ y = o_i^2\ with\ q(L_i, o_i^2) = 0$$
$$or \ldots$$
$$or\ y = o_i^k\ with\ q(L_i, o_i^k) = 0$$

where, L_i^j is a fuzzy label for a distinct fuzzy set defined in the domain of the *n*th component of s^n of the state vector $s = [s^1, s^2, \ldots, sn]$ and o_i^k is the *k*th output action for rule *i, O* being the set of *K* possible actions. The vector $L_i = \left[L_i^1, L_i^2, \ldots, L_i^n\right]$ is known as the modal vector of rule *i* and represents one state of the controller. $q(L_i, o_i^k)$ is the q-value for the fuzzy state L_i and action o_i^k and is initialized to zero. The total number of rules depends on the number of variables in the state vector as well as the number of membership functions defined for each of these. As we have five membership functions for current downtilt and three each for SE^{center} and SE^{edge} variables, so, we have a total of 45 rules/states in our FIS.

The target of FQLC is to find the best consequences for each rule that maximizes the overall reward. Therefore, apart from being able to handle continuous variable problems, FQLC can also overcome the human expertise deficiency in the FIS design phase. If a rule is precisely known than only one output action is enough to define that rule. In cases, where only partial or imprecise knowledge is available, a subset of possible actions can be integrated in the FIS. Finally, in the worst case, if no a-priori knowledge is available than all the possible actions are included in the consequent part of that rule. In this paper we also assume that no a-priori information is available for FQLC design and all action are equally probable in all states at the start of the learning process. Each of the 45 FIS states have five possible actions except that the states with downtilt in very low state can only increase the downtilt or keep it constant and the states with very high downtilt can only decrease the downtilt or keep it at the same level.

3) Instantaneous Reward

In reinforcement learning rewards are used to provide the controller with feedback about its previous action.We define a term *State Quality* (SQ), which characterizes how good a particular state is and the reward is calculated by the difference of two consecutive states. The SQ and the reward function are defined as follows:

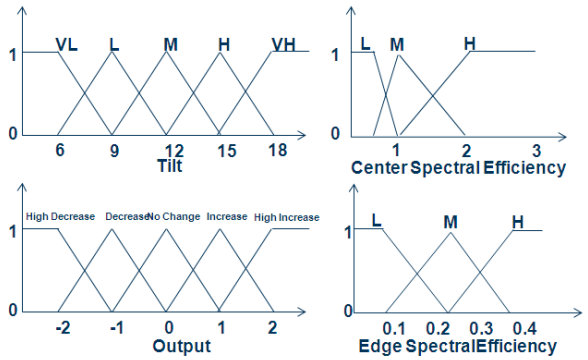


Figure 2: Fuzzy Membership Functions

$$SQ_c = SE_c^{center} + w\,SE_c^{edge} \quad \textbf{(4)}$$

$$r_{c,t+1} = SQ_{c,t+1} - SQ_{c,t} \quad \textbf{(5)}$$

SE^{center} and SE^{edge} are the mean and 5-percentile spectral efficiencies respectively of the cell '*c*' and *w* is a unitless factor to make the edge value comparable to the center value. Here, we use a value of 2 for our studies. As spectral efficiency represents how much data can be transmitted per unit bandwidth therefore mean value corresponds to the average data rate in cell center and the lower 5-percentile value corresponds to the data rates achievable at cell edges. r_{t+1} is the instantaneous reward when the controller transits from state s_t to s_{t+1} with state qualities SQ_t and SQ_{t+1} respectively. Thus for actions that lead to better state quality a positive reward is given and actions leading to poorer state quality are punished with negative rewards.

B. FQLC Algorithm

The FQLC algorithm starts by identifying the current state of the learning agent. This is done by calculating the degree of truth of each FIS rule *i* which is the product of membership values of each input state label for that specific rule *i*:

$$\alpha_i(s) = \prod_{n=1}^{N} \mu_{L_i^n}(s^n) \quad \textbf{(6)}$$

Let *P* be the set of activated rules with a non-zero degree of truth then an output action is selected based on the Exploration/Exploitation policy for each activated rule:

$$o_p = \arg\max_{k \in K} q(L_p, o_p^k) \qquad with\ prob.\ \varepsilon$$
$$o_p = \underset{k \in K}{random}(o_p^k) \qquad with\ prob\ 1-\varepsilon \quad \textbf{(7)}$$

where, ε defines the tradeoff between exploration and exploitation in the algorithm. A value of 1 means no exploration only the action with maximum q-values is chosen. The action to be applied by FQLC is then calculated from all the chosen actions of activated rules as:

$$a(s) = \sum_{p \in P_s} \alpha_p(s) o_p \quad \textbf{(8)}$$

The q-value for the input state vector is then calculated as an interpolation of the current q-values of the activated rules and their degree of truth:

$$Q(s,a(s)) = \sum_{p \in P_s} \alpha_p(s).q(L_p, o_p) \quad (9)$$

The applied action of the FQLC transits the agent to a new state s_{t+1} and receives a reinforcement r_{t+1}. FQLC then calculates the value of new state as:

$$V(s_{t+1}) = \sum_{p \in P_{s_{t+1}}} \alpha_p(s_{t+1}) \max_{k \in K} q(L_p, o_p^k) \quad (10)$$

In order to update the q-values, this value of the new state is used to find the change in the value of new and old state:

$$\Delta Q = r_{t+1} + \gamma V_t(s_{t+1}) - Q(s_t, a(s_t)) \quad (11)$$

And finally using (2) the q-values of the FIS rules are updated as

$$q_{t+1}(L_p, o_p) = q_t(L_p, o_p) + \beta \alpha_p(s_t) \Delta Q \quad (12)$$

As the final action was a combination of multiple actions of different rules so the degree of truth $\alpha_p(s_t)$ of each activated rule is included in order to highlight the contribution of each rule.

III. Learning Strategies

In this paper, we present three different learning strategies and analyze their learning speed and convergence properties.

A. One Cell per Network Snapshot

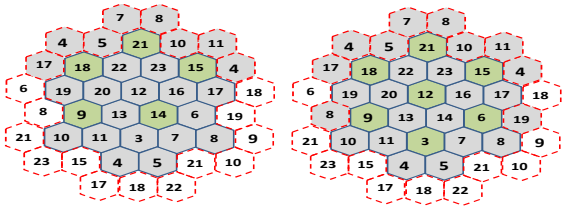
In this strategy we only allow one cell (learning agent) per network snapshot to execute the FQLC algorithm and thus update its downtilt. This simplifies the learning problem as the environment is only affected by one agent's action and therefore it is easy to identify whether that action has led the agent to a better state or not. But this slows-down the learning process as the network size increases. The selection of the agent follows a uniform random distribution.

B. All Cells per Network Snapshot

In order to speed-up the learning process we allow all the cells to update their tilts in every snapshot. However, the feedback from the environment becomes more complicated as the change in the environment is now a result of actions taken by different agents simultaneously.

C. Cluster of Cells per Network Snapshot

In order to combine the benefits of the two strategies we propose to allow a cluster of agents to update their downtilts per network snapshot. The cluster should be formed such that no two direct neighbors are part of it. This ensures that a relatively large number of agents can update their tilts per snapshot and also the change in the environment can be easily characterized. For this a simple clustering algorithm is used that starts by selecting one of the cells as the center cell. The selection follows a uniform random distribution. This center cell is than inserted into a list *To_Update_Green*. All the first tier neighbors of that center cell are than inserted into a second list *To_Ignore_Gray*. After that, starting from the center cell, we scan through all cells in ascending order of their IDs. If the cell is not in *To_Ignore_Gray* list it is inserted in the *To_Update_Green* list and all of its first tier neighbors are inserted in the *To_Ignore_Gray* list. Once we reach the cell with highest ID, we again start with the center cell and now scan the network in descending order of cell IDs and repeat the same procedure to update the *To_Update_Green* list. After we reach the cell with lowest ID we have all the cells in the *To_Update_Green* list that can perform the FQLC algorithm simultaneously. Fig. 3(a) shows examples of such a clustering when we start with cell 14 as the center cell. Cells with solid boundaries form our simulated network, whereas, the cells with dashed lines are the wraparound cells.



(a) (b)

Figure 3: Two Sample Clusters (a) Starting with cell 14 (b) Starting with cell 3

IV. Simulation Environment

The performance of the proposed solutions is evaluated using a system level simulator for LTE networks. The network simulator is based on software libraries provided by Alcatel-Lucent Bell Labs Germany and the Institute of Communication Networks and Computer Engineering, University of Stuttgart, Germany [15].

The simulator basically implements a LTE cellular network for downlink transmission as described in [16] and [17], yet w/o consideration of small-scale fading. The network scenario consists of 7 eNBs with 3 radio cells each as shown in Fig. 3. The term radio cell refers to the coverage area of a transceiver (TRx). The antennas are modeled according to [16] and include both horizontal and vertical pattern. A wrap-around is also implemented for reliable interference calculation. The parameters and placement of TRxs is according to "case 1" described in [16] and [17]. No shadow fading is considered for the simulated scenarios. More specific parameter details are given in Table 1.

Table I: Simulation Parameters

Parameter	Value
System bandwidth	10 MHz
Simulation time step	200ms
Inter-site distance	500m
Pathloss	$128.1 + 37.6 \log_{10}(\max(d_{km}, 0.035))$
eNB Tx power	46dBm
Number of eNB Tx antennas	1 per sector
Number of UE receive antennas	1 Omni
Height of eNB Tx antennas	32m
Height of UE receive antennas	1.5m
eNB max. antenna gain	15dBi
UE max. antenna gain	2dBi
UE speed	10kmph (random walk)
Traffic model	Full Buffer

For comparison purposes we also define a reference system where all cells have a fixed antenna downtilt of 15 degrees. As the considered scenario is a homogenous hexagonal cell structure with equal inter-site distance of 500 meters and uniform traffic distribution, the best solution would be to have the same antenna downtilt at all cells. To find the best value of this fixed downtilt, simulations for downtilts within the range of 6 and 16 were performed with a step of 1 degree and 15 was found to be best.

V. RESULTS AND DISCUSSION

This section presents the performance results for different learning strategies. First a comparison of all three learning strategies is presented in Fig. 4. The x-axis represents the snapshots passed for the complete network, where each snapshot is taken after 200 seconds of simulation time. Each mobile reports back its received SINR to its serving cell every 200 msec and the cells use this statistic of 200 seconds to define the KPIs of SE^{center} and SE^{edge}. The y-axis represents the average of state quality (*SQ)* as defined in (4) for all cells. The *'Reference'* curve represents the average *SQ* for the reference system with 15 degree tilts for all the antennas. The other three curves represent the results for our three learning strategies. For all three strategies the starting value of tilt is set to six degrees across all antennas so that it is far away from the optimal settings. The results show that all three strategies are able to learn the optimal antenna tilt settings with the passage of time. However, allowing only one sector to update its tilt at each snapshot significantly increases the convergence time and the graph shows that it requires more than 1500 snapshots before it gets close to the optimal performance. On the other hand allowing 'all sectors' to update their tilts at each snapshot is quite quick to get close to the optimal performance of the reference system. But the simultaneous actions of neighboring cells make the learning procedure quite complex. Therefore, its performance never matches the performance of the reference system exactly. The ideal solution in such an environment would be to learn the best joint action of all the cells. But the joint action space grows exponentially and makes it impossible to maintain a q-table for state action pairs. Therefore we have to rely on independent learning, where each cell maintains its own q-table of state action pairs regardless of the actions of the neighboring cells. For this reason the reinforcement signal after each action of the cell is not accurate as it also includes the effect of the actions of all the cells. Therefore, the performance of the simultaneous update of all sectors strategy gets affected and never matches the optimal reference system. The proposed clustering mechanism simplifies this situation by allowing only non adjacent cells to update their tilts at each snapshot. Therefore, its performance is better than other two strategies. It quickly gets close to the optimal level and also maintains it to the optimal level over long period of time.

In reinforcement learning problems an important aspect is to identify the best learning parameters. For this reason all three strategies were tested with different values of the learning variables. Table II shows the average state quality between snapshot 1 and 1000 to reflect the convergence speed of the different strategies. Whereas, Table III represents the average state quality between snapshot 1000 and 4000 to show the stable behavior under different learning parameter settings. In both cases the proposed clustering strategy performs better than the other two strategies. It quickly converges to the optimal performance and also maintains that level in the stable region. Moreover, there is not much difference in the performance of different parameter settings for each strategy because of the simple symmetric network scenario. There is only one optimal region around 15 degree tilt without any other local maxima points.

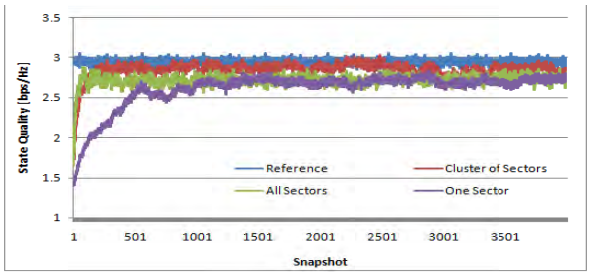


Figure 4: Average State Quality for Complete Network

Apart from this self-optimization capability of the three strategies, self-healing and self-configuration performance was also analyzed. For the self-healing study an outage is created by deactivating the center eNB with cells 12, 13 and 14 as shown in Fig. 5. This creates a coverage hole and the neighboring eNBs try to extend their coverage in order to compensate the coverage loss. For self-configuration analysis the center eNB is then reactivated but with a very high downtilt value of 22 degrees. This can be referred as a gradual deployment of eNB as the higher value of downtilt ensures that the initial coverage of the deployed eNB is very small and after that it tries to extend its coverage in order to balance the coverage and capacity of the whole neighborhood. The SINR distribution for different phases of this outage and deployment is presented in Fig. 5.

Table II: Average State Quality between Snapshot 1 and 1000

Learning Variables			Average State Quality [bps/Hz]			
Exploration Rate(ε)	Learning Rate (β)	Discount Factor (γ)	Reference	One Sector	Cluster of Sectors	All Sectors
6	0.1	0.0	2.95	2.36	2.73	2.71
		0.5	2.95	2.35	2.81	2.7
		0.7	2.95	2.31	2.77	2.73
	0.15	0.0	2.95	2.37	2.7	2.68
		0.5	2.95	2.39	2.78	2.65
		0.7	2.95	2.28	2.77	2.64

Table III: Average State Quality between Snapshot 1000 and 4000

Learning Variables			Average State Quality [bps/Hz]			
Exploration Rate(ε)	Learning Rate (β)	Discount Factor (γ)	Reference	One Sector	Cluster of Sectors	All Sectors
6	0.1	0.0	2.95	2.71	2.86	2.72
		0.5	2.95	2.7	2.86	2.73
		0.7	2.95	2.63	2.82	2.77
	0.15	0.0	2.95	2.65	2.84	2.73
		0.5	2.95	2.55	2.84	2.69
		0.7	2.95	2.64	2.86	2.72

The state quality (SQ) results for both the outage and deployment scenarios are presented in Fig. 6. The y-axis represents the average of SQs for all the 21 cells even when the center three cells are in outage. Therefore the values in outage

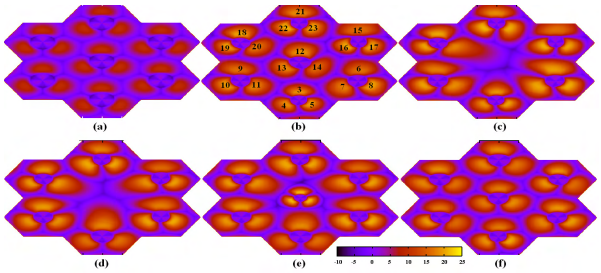


Figure 5: Simulated Network SINR for Cluster Update Strategy; (a) At start of simulation (b) Just before outage at snapshot number 2000 (c) Just after outage at 2001th snapshot (d) Outage compensation at 4000th snapshot (e) Start of recovery/deployment at 4001th snapshot (f) At End

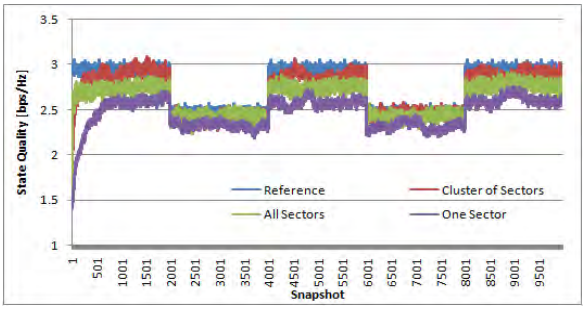


Figure 6: Average State Quality for Outage and Deployment Scenario

are always less than the normal operation. At snapshot 1, the system starts with 6 degree downtilt across all the cells. This means very large coverage areas for all the cells and higher interference to the neighboring cells. Therefore the SQ is quite low but the learning system quickly learns the optimal downtilt values and improves the overall SQ. However, the difference in all three learning strategies in terms of learning speed is clearly visible. At snapshot number 2000 the outage occurs and the system suffers from performance degradation. But the neighboring cells react by adjusting their downtilts to extend their coverage and compensate for this performance degradation. As a result the SQ improves again for all three learning strategies. Finally at snapshot number 4000, the center eNB is reactivated but with a higher downtilt of 22 degrees. This improves the SQ a little bit and the cells again adjust their downtilts using the proposed learning mechanisms. As a result, the gap between the reference system and the learning systems decreases after some snapshots.

VI. Conclusion

In this paper, we have presented different learning strategies for a Fuzzy Q-Learning based solution for the Coverage and Capacity Optimization problem. Simulation results show that the learning speed and convergence properties depend upon how many agents take actions simultaneously. Allowing only one agent to update at a time is highly stable and converges to the optimal settings but is quite slow especially as the network size grows. On the other hand if all the agents take actions simultaneously this speeds-up the process but at the cost of reduced performance gains. Therefore we propose a clustering mechanism that combines the benefits of both these strategies. Moreover, we also analyzed the performance of these strategies in three different network states i.e. deployment, normal operation and cell outages. The results show that the proposed clustering strategy can learn the optimal policy much quicker than the one agent per snapshot strategy and also gives performance better than the all agents per snapshot strategy.

Acknowledgment

We would like to thank Siegfried Klein from Alcatel-Lucent Bell Labs, Germany for his support with the simulator libraries.

References

[1] J.L. Van den Berg, et al., “Self-organisation in future mobile communication networks”, Proceedings of ICT Mobile Summit 2008. Sweden, 2008.

[2] NGMN, “Use Cases related to Self Organising Network. Overall Description”, version 2.02, Available at: http://www.ngmn.org

[3] Technical Specification Group Radio Access Network: Self-configuring and self-optimizing network (SON) use cases and solutions, Release 9, 3GPP TS 36.902 v9.1.0, Mar 2010, Available at: http://www.3gpp.org

[4] M. Garcia-Lozano and S. Ruiz, “Effects of downtilting parameters,” in Proc. 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC ’04), vol. 3, pp. 2166–2170, Barcelona, Spain, September 2004.

[5] J. Niemela, T. Isotalo, J. Lempiainen, "Optimum Antenna Downtilt Angles for Macrocellular WCDMA Network" EURASIP Journal on Wireless Communications and Networking, May 2005.

[6] I. Siomina, P. Värbrand, and D. Yuan, "Automated optimization of service coverage and base station antenna configuration in UMTS networks", IEEE Wireless Communications, vol. 13. pp. 16--25, 2006.

[7] E. Amaldi, A. Capone, F. Malucelli, “Radio planning and coverage optimization of 3G cellular networks", ACM Wireless Networks, 2008.

[8] M. N. ul Islam, et al., “Self-optimization of antenna tilt and pilot power for dedicated channels”, Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Jun. 2010.

[9] H.Eckhardt, S.Klein and M.Gruber “Vertical Antenna Tilt Optimization for LTE Base Stations” Proc. IEEE Veh. Technol. Conf. Workshops (VTC-Spring, WS), May 2011.

[10] A. Gerdenitsch, S. Jakl, Y. Y. Chong, M. Toeltsch “A Rule Base Algorithm for Common Pilot Channel and Antenna Tile Optimisation in UMTS FDD Networks”, ETRI Journal, Vol. 26, No. 5, Oct. 2004.

[11] M. Pettersen, L. E. Bråten, and A. G. Spilling, “Automatic antenna tilt control for capacity enhancement in UMTS FDD,” in Proc. IEEE 60th Vehicular Technology Conference (VTC ’04), vol. 1, pp. 280–284, Los Angeles, USA, September 2004.

[12] R. Razavi, S. Klein, and H. Claussen, “A Fuzzy Reinforcement Learning Approach for Self-Optimization of Coverage in LTE Networks”, Bell Labs Technical Journal, 15: 153–175, Dec 2010.

[13] R. Sutton “Reinforcement Learning: An Introduction” MIT Press, 1998.

[14] P. Y. Glorennec, ”Reinforcement Learning: an overview,” European Sym. on Intelligent Techniques, Aachen, Germany, 2000.

[15] IKR, “Institute of Communication Networks and Computer Engineering,University of Stuttgart”, Stuttgart, Germany, http://www.ikr.uni-stuttgart.de/, 2011.

[16] 3rd Generation Partnership Project, “Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA),” 3GPP TR25 814, Sept. 2006.

[17] 3GPP, “Further advancements for E-UTRA physical layer aspects (Release 9)”, TR 36.814, v9.0.0, March 2010.