Load Balancing in Cellular Networks: A Reinforcement Learning Approach

Kareem Attiah[†], Karim Banawan[†], Ayman Gaber[‡], Ayman Elezabi[⋆], Karim Seddik[⋆],
Yasser Gadallah[⋆], Kareem Abdullah[⋆]
[†]*Alexandria University*, [‡]*Vodafone Egypt*, [⋆]*The American University in Cairo*

***Abstract*—Balancing traffic among network installed radio base stations is one of the main challenges facing mobile operators because of the unhomogeneous geographical distribution of mobile subscribers in addition to practical and environmental limitations preventing acquiring the best locations to build radio sites. This increases the challenge of satisfying the increasing data speed demand for smartphone users. In this paper, we present a reinforcement learning framework for optimizing neighbor cell relational parameters that can better balance the traffic between different cells within a defined geographical cluster. We present a comprehensive design of the learning framework that includes key system performance indicators and the design of a general reward function. System level simulations show that reinforcement learning based optimization for neighbor cell borders can significantly improve overall system performance; in particular, with a reward function defined as throughput, an improvement up to 50% is achieved.**

I. Introduction

Traffic growth in cellular networks is driven by both the rising number of smartphone subscriptions and an increasing average data volume per subscription, fueled primarily by video content. This trend is expected to continue, as video is increasingly embedded in all types of online content. In addition, emerging media formats and applications, such as streaming high-quality video and augmented/virtual reality, will continue to drive traffic growth while enhancing the user experience. Video traffic in mobile networks is forecast to grow by around 35% annually until 2024 to account for 74% of all mobile data traffic while traffic from social networking is also expected to rise annually by 24% over the next 5 years [1].

Total traffic carried by 4G networks is not uniformly distributed among different cells. As a rule of thumb, 15% of the network cells carry 50% of the generated traffic [2]. This increases the network resource utilization in certain spots (hot spots) compared to relaxed utilization in the rest of the network that could be first- or second-tier neighbors of the highly utilized cells. This increases the need for offloading the congested cells and employing load balancing techniques that can balance traffic among neighbor cells with minimum impact on radio channel conditions for the offloaded users.

Unlike earlier generations of cellular networks, LTE networks are designed to support some degree of self-optimization (SO) functionalities. Moreover, future cellular networks are likely to rely further on such SO mechanisms in order to dynamically cope with the exponential growth in traffic [3]. As a result, the past decade has witnessed a growing interest in realizing load balancing via self-tuning of handover (HO) parameters [4]–[7]. In [4], the authors present a simple but powerful algorithm to iteratively update the cell individual offset (CIO) of congested serving cells with respect to their underutilized neighbor cells. The CIO is an offset that can be applied to alter the handover decision, which in effect changes the effective radius of cell i with respect to cell j. The main goal here is to determine the CIO value that would offload the highest amount of user traffic while preventing the ping-pong effect. One possible drawback of this approach is that it does not allow the transferred load to propagate further through the entire network. Another approach was introduced in [6] where the authors propose a reactive load-balancing technique in which the CIOs of the serving and target cells are symmetrically updated with opposite signs by a specific value. This value is chosen from a predefined discrete subset by a reinforcement learning (RL) agent that interacts with the environment and learns the optimum choice with respect to some reward function. In the proposed model, the reward function is measured in terms of the negative of the number of unsatisfied users, i.e. the number of users whose data rate falls below a certain threshold. However, in this model the RL states are partially expressed in terms of the cell-edge user distribution, a piece of information which may not be readily known to a network operator in a practical scenario.

In addition to the techniques above, there have been a few attempts to automate load balancing in heterogeneous networks where femto-cells are deployed to improve capacity and coverage. In [7], a supervised learning strategy is introduced to solve the load balancing problem. Specifically, by dynamically expanding the range of femto-cells through adjustments of CIO values, macro cells are relieved from increased user traffic. The network estimates the correct CIO values through historical data that are generated using a network simulator. A similar approach is reported in [5], in which a combination of fuzzy logic control and reinforcement learning (RL) algorithms is employed to determine the required femto-cell range expansion, measured in both femto-cell transmit power and CIO. We refer the reader to [8], for a more comprehensive survey on SO techniques for load balancing.

Similar to [5]–[7], in this paper we present a machine learning approach to solving the load balancing problem for LTE cellular networks. Particularly, we employ RL to optimize the throughput and/or resource block utilization of the network. The RL technique is a dynamic learning framework, where an agent learns to self-optimize its action by interacting with

an environment. More specifically, in RL the agent seeks to specify the optimal policy to maximize the long term average of a certain reward function. RL can be modeled as Markov decision process (MDP), where the agent observes the current state of the environment and applies one of several allowable actions, which changes the state of the environment and earns a reward as a result of the agent's action [9].

In this work, we employ deep RL for load balancing. To that end, we use NS-3 [10] to implement an accurate model of an LTE cellular network. This model represents the environment, which the agent interacts with. We choose NS-3 as it supports the full protocol stack of the LTE system and can provide simulated, yet accurate, key performance indicators, which are readily available for cellular operators. We choose the state of the environment to be a subset of these KPIs. For the steering action, we use the CIO to trigger the HO procedure. This effectively forces some of the users to leave congested cells even if these cells may result in the highest user SNR. Additionally, we choose the reward function to be the total throughput of the network.

To solve the MDP corresponding to RL, we use Q-learning [9]. In Q-learning, the agent constructs a table of the Q-values corresponding to each (state, action) pair where the Q-value is a proxy for the quality of the decision at a certain state. These Q-values are updated based on the interaction with the environment. Furthermore, since the chosen states in our formulation belong to an infinite space, we use a deep neural network to approximate the Q-values and replace the Q-table with a Q-function.

The rest of the paper is organized as following: Section II presents the formal system model, Section III presents the proposed machine learning framework, including the used environment model and RL, Section IV provide some numerical results that can be obtained using our proposed technique and some discussions, and Section V concludes the paper.

II. System Model and Load Balancing Mechanism

Consider the downlink of an LTE cellular network with N base stations (eNodeBs or eNBs) and K active users. Initially, K_n users are associated with the nth base station, such that $\sum_{n=1}^{N} K_n = K$. The kth user is associated with the nth base station if it results in the best-received signal at the kth user. The kth user periodically reports the channel quality indicator (CQI) ϕ_k to the associated base station. The CQI is related to the SINR or BLER measured by the user. Furthermore, the kth user wishes to be served with a minimum data rate of R_k bits/second. Consequently, the associated base station assigns B_k physical resource blocks (PRBs) to the kth user, such that:

$$B_k = \left\lceil \frac{R_k}{g(\phi_k, M_{n,k})\Delta} \right\rceil \tag{1}$$

where $\Delta = 180$KHz in LTE, and $g(\phi_k, M_{n,k})$ is the achievable spectrum efficiency based on the reported CQI ϕ_k and the antenna configuration $M_{n,k}$ using the default LTE scheduler. The total required PRBs needed to serve all associated users in the nth base station is $T_n = \sum_{k=1}^{K_n} B_k$. Moreover, we denote the total offered PRBs by the nth base station by $\tilde{T}_n$.

Our main goal is to offload traffic from over-utilized cells to under-utilized cells so that the total network load can be shared more evenly among network cells. One possible method of performing this is by adjusting the cells' CIO values. According to the 3GPP LTE specifications [11], a user initially served by some cell i will commence a handover request to some neighbor cell j if the following condition holds

$$M_j + \theta_{j\to i} > \mathit{Hys} + M_i + \theta_{i\to j}, \tag{2}$$

where M_i and M_j are the measured values of reference signal received power (RSRP) from cells i and j, respectively, $\theta_{i\to j}$ is the CIO value of cell i with respect to cell j, $\theta_{j\to i}$ is the CIO value of cell j with respect to cell i, and *Hys* is a hysteresis value that minimizes the likelihood of ping-pong scenarios that arise due to fading fluctuations. One may interpret $\theta_{i\to j}$ as the offset value that makes the measured RSRP of cell i appear stronger (or weaker) when compared with the measured RSRP of cell j. Likewise, $\theta_{j\to i}$ is the added value that makes the measured RSRP of cell j appear stronger (or weaker) when compared with the measured RSRP of cell i.

Note that this definition allows the values of the CIOs to be different depending on the neighbor cells. However, for simplicity we use the same offset value at a given cell, irrespective of its neighbor cells. That is, we assume that $\theta_{i\to j} = \theta_i$, for all j's that are neighbors to cell i. In this case, one may readily see that a good strategy for attaining load balancing is by assigning high CIO values to under-utilized cells and low CIO values to over-utilized ones.

As we shall see later, we employ an RL framework to learn which cells are over-utilized (or under-utilized) and accordingly perform decisions on the CIO values so as to achieve network load balancing. To do so, we assume that the network is equipped with a central agent, that can monitor all key performance indicators (KPIs) at the network level and controls θ_n. More specifically, the central agent and network (environment) interact at discrete time instants $t = 0, 1, 2 \cdots$. We define the state of the network at time t, $S(t)$, to be a subset of the KPIs reported in the network at time t. We define the action of the central agent at time t, $A(t)$, to be the CIOs at time t, i.e., $A(t) = (\theta_n(t),\, n \in \{1, \cdots, N\})$. Due to applying the action, the state of the network changes to $S(t+1)$, and the central agent receives a reward $\mathcal{R}(t+1)$.

III. Proposed Load Balancing Scheme

Many possible reward functions may be considered in addition to different weighted sums of these. Possible reward functions include:

1) Instantaneous sum throughput,

$$\mathcal{R}(t) = \sum_{n=1}^{N} \sum_{k_n=1}^{K_n} \hat{R}_{k_n}(t) \tag{3}$$

where $\hat{R}_{k_n}(t)$ is the actual measured throughput of the k_nth user in the nth cell at time t.

2) Total number of blockage events,

$$\mathcal{R}(t) = -\sum_{n=1}^{N}\sum_{k_n=1}^{K_n} E_{k_n}(t) \tag{4}$$

where $E_{k_n}(t) = 1$ if the k_nth is not served with the minimum required data rate R_{k_n}.

3) Average deviation of the total number of offered PRBs,

$$\mathcal{R}(t) = -\sum_{n=1}^{N}\left|\tilde{T}_n(t) - \frac{1}{N}\sum_{n=1}^{N}\tilde{T}_n(t)\right| \tag{5}$$

At each time instant, the central agent implements a stochastic policy π_t, where $\pi_t(a|s)$ is the probability that the central agent performs action $A(t) = a$ given it was in state $S(t) = s$. The central agent aims at maximizing the long-term average reward function, i.e.,

$$\max_{\pi} \lim_{L\to\infty} \mathbb{E}\left[\frac{1}{L}\sum_{t=0}^{L}\mathcal{R}(t)\right] \tag{6}$$

where $\pi = (\pi_1, \pi_2, \cdots)$.

A. Reinforcement Learning Technique

In this section, we give an overview on RL. Specifically, for this problem, we use deep Q-learning technique [12], [13] to construct a self-optimizing agent. The agent's task is to maximize the long-term average of the reward functions presented in Section II. The idea is to learn an *approximate* version of the Q-table using a neural network.

We begin our discussion by defining the state $S(t)$. In this work, we propose three KPIs to define the network state [14]. The first is the resource block utilization (RBU) $\mathbf{U}(t) \in [0,1]^N$, which is an N-length vector. Each element in $\mathbf{U}(t)$ corresponds to the fraction of the utilized RB in the nth eNB at time t. This is an important feature as it reflects the congestion level of each cell. The second KPI is the total downlink (DL) throughput vector $\mathbf{R}(t) \in \mathbb{R}_+^N$. Each element in $\mathbf{R}(t) \in \mathbb{R}_+^N$ represents the total DL throughput in the nth cell at time t, which quantifies the overall performance of each cell. Finally, the third KPI is the modulation and coding scheme (MCS) utilization $\mathbf{M}(t) \in [0,1]^{N\times\mu}$, where μ is the total number of modulation and coding schemes defined in LTE, which are currently 29 active MCS in [11]. Each element in the matrix $\mathbf{M}(t)$ is the ratio of users that employs a specific MCS. This is a metric to assess the relative channel qualities of the users over the cell. Now, we are ready to write the state $S(t)$, which is a concatenation of all these KPIs, i.e.,

$$S(t) = [\mathbf{U}(t)^T \quad \mathbf{R}(t)^T \quad \text{vec}(\mathbf{M}(t))^T]^T \tag{7}$$

where $vec(\cdot)$ is vectorization function. Consequently, the input layer of the neural network is a vector of size $2N + N\mu$.

For the actions $A(t)$, the central agent can control the CIOs of all eNBs (or specific subset of them). In each cell, the central agent choose $\theta_n(t)$, where $\theta_n(t)$ is the CIO of the nth eNB at time t. The values of $\theta_n(t)$ are selected from the discrete subset of $[-\theta_{\max}, \theta_{\max}]$ dB of size L, where $\theta_{\max}$ is the maximum possible CIO in the system, i.e.,

$$A(t) = [\theta_1(t) \quad \theta_2(t) \quad \cdots \quad \theta_N(t)]^T \tag{8}$$

This creates an action space of L^N possible actions. Additionally, we use the neural network as a multi-class classifier. In this case, the neural network operates with the goal of identifying the optimal action given the state of the environment. Hence, the output layer of the neural networks consists of L^N output neurons, with softmax activation function, where the softmax function $\sigma(\cdot)$ is defined as:

$$\sigma(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{L^N}\exp(z_j)} \tag{9}$$

where z_i is the an input to the output layer of the neural network. To learn the non-linear dependencies of the actions $A(t)$ on $S(t)$, we further add N_h hidden layers. Each hidden layer has the size of $\max\{2N + N\mu_t, L^N\}$ neurons and with rectified linear (relu) activation function. To avoid overfitting, some of the weights of each hidden layer are randomly dropped with probability q. This is a known regularization technique in neural networks to minimize possible generalization errors.

Now, our neural network is constructed and ready to learn an approximate version of the Q-function (Q-table) by employing RL. To that end, the learning is done over N_e episodes. Each episode corresponds to a complete simulation of the environment (in this case the NS-3 simulator) over T_{sim} time period with a time step Δ. To balance the exploration and exploitation, we define $\epsilon(t)$ to be the probability of picking a random action in time t (exploration), where,

$$\epsilon(t) = (\epsilon_d)^{\ell(t)} \tag{10}$$

where $0 < \epsilon_d < 1$ is the decay factor of the exploration probability and $\ell(t)$ is the index of the time step corresponding to t.

At each time step, it is required to estimate the optimal value function (Q-function) $Q^*(S, A)$ by forming an estimate $Q_t(S(t), A(t))$. To that end, the central agent picks a random action with probability of $\epsilon(t)$ and exploit the action that maximizes the predicted Q-function with the probability $1 - \epsilon(t)$, i.e.,

$$A(t) = \begin{cases} \arg\max_A \quad Q_t(S(t), A) & \text{w.p.} \quad 1-\epsilon(t) \\ \mathcal{A} & \text{w.p.} \quad \epsilon(t) \end{cases} \tag{11}$$

where $\mathcal{A}$ denotes a random action that is drawn uniformly from the action space.

The action $A(t)$ is applied to the environment via the gym interface [15]. The reward $\mathcal{R}(t+1)$ and the state $S(t+1)$ are observed. The Q-function is then updated using the following equation:

$$Q_{t+1}(S(t), A(t)) = \mathcal{R}(t+1) + \lambda \max_A Q_t(S(t+1), A) \tag{12}$$

where λ is the discount, which signifies how much we weigh

Fig. 1: The simulate LTE Network Scenario, all UEs are initially served by the center cell.

future expected reward.

Now, we train the neural network with the results of this time step. In this case, the target function $y(t)$ consists of a vector of the previously learned $Q_t(S(t), A)$ for all A in the action except for $A(t)$, i.e., the action taken in the previous time step. Consequently, the neural network is trained by the new example $(S(t), y(t))$.

IV. Numerical Analysis

A. Case Study

We consider the LTE network scenario shown in Fig.1, consisting of 3 eNBs with intersite distance of 500 meters. Each eNB covers a hexagonal area using a single omnidirectional antenna. We refer to the middle cell as cell 2, whereas the left and right cells as cell 1 and 3, respectively. Within each cell coverage, the UEs assumed to be stationary and are deployed randomly according to a predefined UE density value: high (low) density values amount to dense (light) deployment. The stationarity in our model is to simplify the simulation and give better insight into the results, i.e. mobility would not change the method used. Using appropriate density values, we implement the the extreme case of having all UEs be initially served by cell 2, i.e., the center cell. The simulation is carried out using the NS-3 simulator [10]. Table I summarizes the parameters used in our simulations and their corresponding values. A parameter of particular interest is the environment step time, which is defined as the duration for which the LTE network is allowed to run before a new action is generated by the RL agent (i.e., the length of one RL time step in seconds). In our simulations, we set this value to 1s. The 1s time step is chosen such that the HO procedure has sufficient time to be triggered as a result of changing the CIO level. We will show in the next section that the proposed RL agent is able to achieve significant gains in the total DL throughput within a few iterations. Thus, in a real-time practical scenario, we expect our proposed algorithm to possess a relatively fast convergence (order of minutes).

B. Results and Discussions

In this section, we present the results of our case study. In Table II, we present the hyper-parameters of the used RL technique. We aim at optimizing the long term average of the total DL throughput in the network by controlling the CIO value of each cell. In all results, we compare with the base line system which does not employ CIO, i.e., when all cells have CIO of 0 dB.

TABLE I: LTE Environment Simulation Parameters

Parameter	Value
Intersite Distance	500m
Center Frequency	2GHz
System Bandwidth	10MHz
# of Antennas	Tx: 1, Rx: 1
Antenna Pattern	Omni
eNB antenna height	30m
eNB Tx Power	20dBm
Pathloss Model	Okurama-Hata
Shadowing	No
Traffic Direction	Downlink
Traffic Model	Fixed packet size at a fixed time interval Packet size = 12 KB Packet Interval = 1s.
Scheduler	Proportional Fair
UE Mobility Model	Static
UE Antenna Height	1.5m
Initial UE distribution	#UEs in cell 2: 22 #UEs in cells 1, 3: 0
Handover	A3-event based Time to trigger = 40ms Hysteresis = 3dB.
Environment step time	1s

TABLE II: Hyper-parameters of the machine learning technique

Hyper-parameter	Value
CIO set	$\{-6, -3, 0, 3, 6\}$dB
Reward function	Total DL throughput
Number of episodes	10
Number of iterations/episode	50
ϵ_d	0.995
Discount (λ)	1
Number of hidden layers (N_h)	3
Dropout (q)	0.5
Activation function	relu
Activation of the output layer	softmax
Loss function	categorical cross-entropy
Optimizer	Adam(0.001)

In Fig. 2, we plot the simulated DL throughput versus the episode index. We take the average of all DL throughput values in the time span of one episode, which consists of 50 iterations (time steps). Upon comparing with the base line system, we observe a enhancement in the total DL throughput due to optimizing the CIO of each cell. More specifically, we observe that the average throughput with RL outperforms the base line system at any episode. The throughput converges

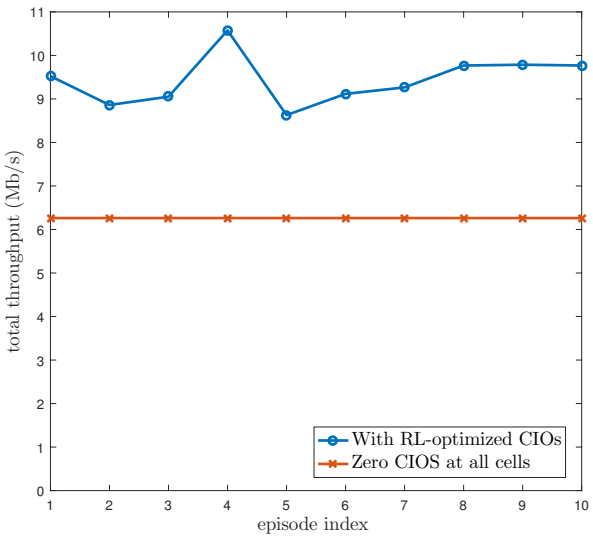


Fig. 2: Average throughput per episode versus the episode index. The RL-optimized CIOs result in 55.91% increase in total throughput over the base-line system that uses zero CIOs.

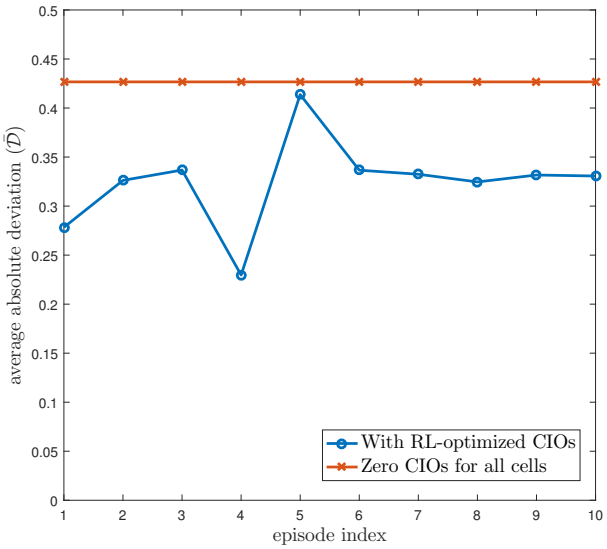


Fig. 3: Average absolute deviation from the mean RB utilization per episode versus the episode index. The deviation decreases by 19.71% below the base-line system.

at 9.76Mb/s starting from episode 8. The base line system achieves 6.26Mb/s at all episodes. This results in 55.91% increase in the total throughput of the system as a consequence of load balancing. Generally speaking, due to the exponential decay of ϵ, one expects to have a monotonic increase in the throughput over the episodes as the random exploration decreases with time and exploitation of the (supposedly) optimal decision dominates. Aside from the fluctuation at episode 4, this seems to be the case. A second observation one can quickly infer is that the RL model has converged to a strictly sub-optimal policy as opposed to an optimal one. We speculate that this may have occured due to the use of relatively fast exploration rate of decay, or a relatively simple neural network. Next, we investigate the RB utilization across the simulated network cells. We calculate the average absolute deviation from the mean as:

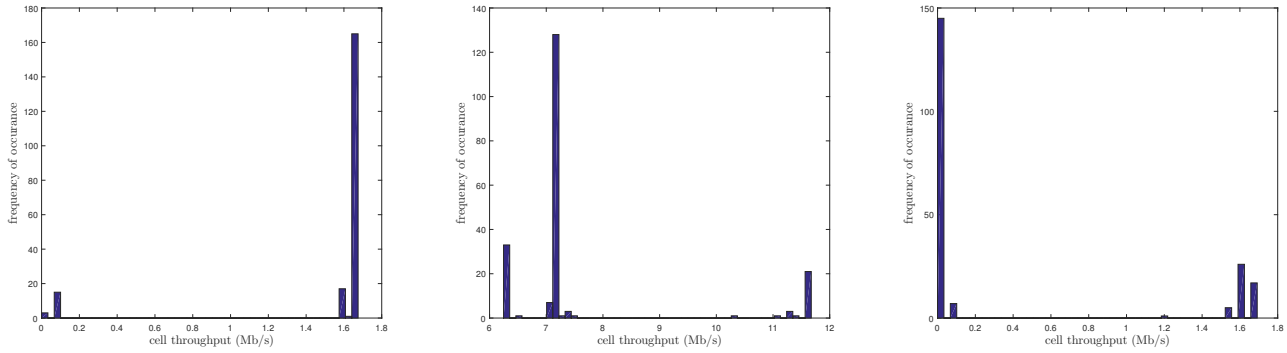
$$\mathcal{D}(t) = \frac{1}{N}\sum_{n=1}^{N}\left|\tilde{T}_n(t) - \frac{1}{N}\sum_{n=1}^{N}\tilde{T}_n(t)\right| \tag{13}$$

Again, we take the average of these absolute deviation over each episode to get $\bar{\mathcal{D}}$ and compare it with absolute deviation in the case of the base line system. In Fig. 3, we plot the average absolute deviation versus the episode index. The figure reveals that the average absolute deviation converges to 0.3371 with the RL-optimized CIOs in contrast to 0.42 with zero CIOs, i.e., the average absolute deviation has decrease by 19.71%, and thus achieving superior load balancing. It is crucial to remark that since the reward function is defined in terms of the total DL throughput, it may not be the case that the individual cell utilization converges to the average cell utilization. In fact, there have been reported cases of outstanding total DL throughput despite having significant disparity in individual cell utilization (66% in cell 2 and 90% in cells 1 and 3). This follows from the fact that once they have moved to cell on either side, cell-edge users enjoy the use of an excess number of unexploited physical RBs.

In Fig. 4 and Fig. 5, we plot the histogram of the throughput and the RB utilization of individual cells. We restrict our plot to the last 4 episodes (last 200 iterations) to minimize the effect of the random exploration in the first 6 episodes. This reflects an aggregate view of the cellular network KPIs after operating the agent at convergence. From Fig. 4, we can see that at steady state, cell 1 accommodates a maximum DL throughput of about 1.6 Mb/s for nearly most of the time as opposed to the inferior maximum DL throughput of 0.9 Mb/s for the zero CIO case. Likewise, we notice a similar increase in the maximum DL throughput of cell 2 from 6.26 Mb/s to 7 Mb/s. Additionally, we observe some rare events where the throughput increase significantly to 11.5Mb/s. For cell 3, the load is almost negligible in both cases. We conclude that the aforementioned suboptimal policy amounts to offloading user from cell 2 to cell 1 only, while completely disregarding cell 3. We speculate that this is directly connected to the edge user placement around cell 3.

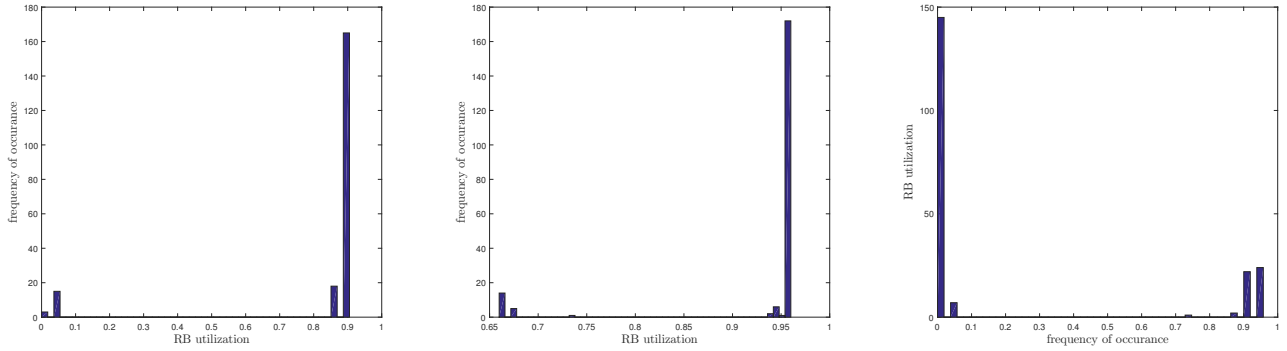
V. Discussion and Conclusions

In this paper, we presented a reinforcement learning startegy to achieving load balancing in LTE cellular networks. In order to simulate the LTE environment, we used NS-3, a high-level simulator that allows the extraction of practical key performance indicators. The states of the RL agent were described in terms of a subset of such indicators, whereas the RL reward function was defined in terms of the network DL throughput. Finally, by allowing the RL agent to perform actions via adjusting the cell offsets, we have shown clear



(a) cell 1 (b) cell 2 (c) cell 3

Fig. 4: Histogram of individual cell throughput for the last 4 episodes.



(a) cell 1 (b) cell 2 (c) cell 3

Fig. 5: Histogram of individual cell RB utilization for the last 4 episodes.

enhancement of the network DL throughput and absolute deviation of RB utilization.

Future extensions of this work include investigating other reward functions (e.g., PRB utilization, user blockage probability, etc). We expect that introducing mobility, shadowing, and increasing the LTE network size would reflect more practical scenarios although would not yield qualitatively different results.

REFERENCES

[1] Ericsson. Ericsson mobility report june 20019.

[2] Harri Holma and Antti Toskala. *LTE advanced: 3GPP solution for IMT-Advanced*. John Wiley & Sons, 2012.

[3] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang. What will 5g be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, June 2014.

[4] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan. Load balancing in downlink lte self-optimizing networks. In *2010 IEEE 71st Vehicular Technology Conference*, pages 1–5, May 2010.

[5] P. Muoz, R. Barco, J. M. Ruiz-Avils, I. de la Bandera, and A. Aguilar. Fuzzy rule-based reinforcement learning for load balancing techniques in enterprise lte femtocells. *IEEE Transactions on Vehicular Technology*, 62(5):1962–1973, Jun 2013.

[6] S. S. Mwanje and A. Mitschele-Thiel. A q-learning strategy for lte mobility load balancing. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2154–2158, Sep. 2013.

[7] C. A. S. Franco and J. R. B. de Marca. Load balancing in self-organized heterogeneous lte networks: A statistical learning approach. In *2015 7th IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–5, Nov 2015.

[8] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza. A survey of machine learning techniques applied to self-organizing cellular networks. *IEEE Communications Surveys Tutorials*, 19(4):2392–2431, Fourthquarter 2017.

[9] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[10] Nicola Baldo, Marco Miozzo, Manuel Requena-Esteso, and Jaume Nin-Guerrero. An open source product-oriented lte network simulator based on ns-3. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 293–298. ACM, 2011.

[11] 3GPP ETSI TS 136 213 V14.2.0. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. 2017.

[12] V. François-Lavet, P. Henderson, R. Islam, M. Bellemare, J. Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018.

[13] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[14] Kareem Abdullah, Noha Korany, Ayman Khalafallah, Ahmed Saeed, and Ayman Gaber. Characterizing the effects of rapid lte deployment: A data-driven analysis. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 97–104. IEEE, 2019.

[15] P. Gawlowicz and A. Zubow. ns3-gym: Extending OpenAI Gym for Networking Research. *CoRR*, 2018.