Deep Reinforcement Learning-based CIO and Energy Control for LTE Mobility Load Balancing

Ghada Alsuhli, Hassan A. Ismail, Kareem Alansary, Mahmoud Rumman, Mostafa Mohamed, Karim G. Seddik

Electronics and Communications Engineering Department, American University in Cairo, Cairo, Egypt 11835
Email: {ghadaalsuhli, hassan_ismail, auckareemalansary, mrumman, mustafamohammed_auc, kseddik}@aucegypt.edu

***Abstract*—cellular networks' congestion has been one of the most common problems in cellular networks due to the huge increase in network load resulted from enhancing communication quality as well as increasing the number of users. Since mobile users are not uniformly distributed in the network, the need for load balancing as a cellular networks' self-optimization technique has increased recently. Then, the congestion problem can be handled by evenly distributing the network load among the network resources. Lots of research has been dedicated to developing load balancing models for cellular networks. Most of these models rely on adjusting the Cell Individual Offset (CIO) parameters which are designed for self-optimization techniques in cellular networks. In this paper, a new deep reinforcement learning-based load balancing approach is proposed as a solution for the LTE Downlink congestion problem. This approach does not rely only on adapting the CIO parameters, but it rather has two degrees of control; the first one is adjusting the CIO parameters, and the second is adjusting the eNodeBs' transmission power. The proposed model uses Double Deep Q-Network (DDQN) to learn how to adjust these parameters so that a better load distribution in the overall network is achieved. Simulation results prove the effectiveness of the proposed approach by improving the network overall throughput by up to 21.4% and 6.5% compared to the base-line scheme and the scheme that only adapts CIOs, respectively.**

***Keywords*—** Load Balancing (LB), Reinforcement Learning (RL), Double Deep Q-network (DDQN), Cell Individual Offset (CIO)

I. Introduction

Throughout the last decade, there has been a considerable increase in the number of smart cell phone users. At the same time, there has been a huge technological breakthrough in the field of mobile communications and cellular networks to enhance the quality of the means of both voice and data communications. This increase in the quality accompanied by the increasing demand for cellular communications has created noticeable congestion in cellular networks. This increase in mobile traffic is expected to grow aggressively in the coming years with the introduction of future generations [1]. Increasing the networks' capacity is always costly and it is not always feasible, which makes it not an efficient solution for the congestion problem. Another more feasible solution, that has been recognized recently, to deal with this problem is load balancing.

Mobile users are not expected to be uniformly distributed in cellular networks [2]. This leads to inefficient use of the peak performance of cellular networks due to the bad utilization of the network resources. In other words, at any point in time, there are always free resources in cellular networks that are not utilized. Therefore, exploiting these existing free resources is a more reasonable solution for the congestion problem rather than introducing new resources into the networks.

Load balancing is a way to exploit the free resources that exist in cellular networks to overcome the congestion problem. It is based on the process of moving some of the loads from the over-loaded eNodeBs to the less loaded eNodeBs. By doing this, the loads can be better distributed in the network. Starting from LTE and continuing through future generations of cellular networks, networks are designed in a way that allows for self-optimization [3]. This has enabled several techniques to realize load balancing through self-adjustment of the network's self optimizable parameters.

Two load balancing algorithms were proposed in [4]. The first algorithm was a reactive one. The algorithm achieves load balancing by first checking the eNodeB utilization. If the utilization of some eNodeB exceeds a certain threshold, it decrements the CIOs of this eNodeB relative to its neighboring eNodeBs with some constant, determined prior to network operation, and increments the CIOs of the neighboring eNodeBs relative to this eNodeB by a constant step. This process keeps repeating until the over-utilized eNodeB is no longer congested, namely, its utilization is less than the threshold. The second algorithm is a Q-learning-based load balancing algorithm. It is basically the same as the reactive algorithm except for the constant step used to increment or decrement the CIOs being no longer a constant but rather a parameter that gets learned by a Q-learning algorithm during the operation of the network. The benefit of this algorithm is that after the learning process of this parameter, the process of handing over the extra loads from the congested eNodeB to the neighboring eNodeBs will be much faster. This reduces the probability of having some calls dropped by the congested eNodeB before moving to the less congested eNodeBs. Despite the effectiveness of these two algorithms, there is a drawback of both. They ignore checking the utilization of the neighboring eNodeBs before moving the load to them. This can lead to a ping-pong effect which is handing over the load back and forth between the over-utilized eNodeB and the neighbor eNodeB if the neighbor eNodeB is over-utilized as well. The author of [4] tried to solve this problem by setting a counter to avoid returning the load instantaneously to the source eNodeB after handing it over to the target eNodeB. However, this is not the ultimate solution to the problem.

In [5], another load balancing algorithm was presented. It has completely solved the ping-pong problem that appeared in [4]. This algorithm is also based on the reactive load balancing algorithm proposed in [4]. However, it considers the utilization of the neighboring eNodeBs as well in making the decision of adjusting the CIOs. Because of this consideration, the overall performance of the whole network is greatly improved. The authors of [6] proposed a more sophisticated approach for load balancing by considering two more factors besides the utilization of the eNodeBs. The first additional factor is the throughput of the eNodeBs, and the other one is the Modulation and Coding Scheme (MCS) of the eNodeBs. The authors, then, define a small set of eNodeB CIO values and use a reinforcement learning agent to learn the optimum values of the eNodeBs' CIOs at every state of the network. The reinforcement learning algorithm used in this proposed model is Q-learning. This model has an advantage over the previous ones in that it takes more network features into account. However, this model suffers from the problem of dealing with absolute CIOs rather than relative CIOs, which adds a restriction for the model to optimize the performance of the network.

The authors in [7] used the Q-learning technique integrated with a macro-femto eNodeBs' arrangement. Algorithms balance the load between the overlay macro eNodeB and the underlay femto eNodeBs according to several metrics. Two algorithms work together to update the Q-table containing the optimal transmission power of the femto eNodeBs to sustain signal to interference plus noise ratio (SINR) and throughput. The load balancing based on reinforcement learning of end-user SINR (LBRL-SINR) observe the SINR of the user equipment, the drop rate and block error rate of the macro eNodeB, and adjust the transmission power of the femto eNodeBs to achieve load balancing. The load balancing based on reinforcement learning of the macro eNodeB throughput (LBRL-T) algorithms observes the results of the actions of LBRL-SINR on the throughput, drop rate, and block error rate of the macro eNodeB and update the Q-table to achieve constant throughput. The authors used power adjustment of the femto eNodeBs to offload the congested macro eNodeB, which has a constant transmission power.

In [8], the authors proposed a deep reinforcement learning (DRL) based mobility load balancing (MLB) algorithm along with a two-layer architecture. It presents a two-layer architecture to alleviate the large-scale load balancing issue, by grouping small eNodeBs (eNodeBs), by historical loads, into clusters. To optimize the load balancing an intracluster MBL algorithm based on DRL is proposed. Furthermore, an offline safeguard mechanism is introduced to mitigate the risks of using non-optimal MLB policy which could potentially cause random handover events to trigger. The simulations proved that the proposed algorithm considerably outperforms existing algorithms. It was also proved that the clustered architecture outperforms the centralized one.

In this paper, a new load balancing model is proposed. This model attempts to overcome the problems accompanied with the previously discussed models. In our model, an RL agent is used to learn the optimum actions as a function of the state of the network. Double Deep Q-Networks (DDQNs) are used in the agent to achieve this goal. In addition, the proposed model considers three features of the network: the Resource Block (RB) utilization of the eNodeBs, the throughput of the eNodeBs, and the MCS of the eNodeBs' users, to be the state of the environment. Regarding the actions, our model considers applying two adjustments to the network so as to properly distribute the load and achieve higher network performance. The first adjustment is modifying the relative CIOs of the eNodeBs in the network. The second adjustment is tuning transmission power of the eNodeBs in the network. To the best of our knowledge, this paper is the only work that suggests merging the control of the power and CIOs to balance the load in LTE networks. Since maximizing the throughput of the network is an aim of the network provider, the total downlink throughput of the network is adopted as the reward. NS-3 network simulator is used to simulate the LTE network, which represents the environment that the RL agent interacts with to perform the learning process.

The rest of this paper is organized as follows. Section II discusses the system model for LTE and the load balancing mechanism. Section III proposes a new load balancing model and explains the network environment and the reinforcement learning algorithm used in the model. Section IV presents the numerical analysis and results of the proposed model. Section V provides the conclusion for the paper.

II. System Model and Background

A. System Dynamics

A downlink LTE system is considered in this paper. The system is composed of C eNodeBs and U users. Every user u measures the Signal-to-Interference-plus-Noise-Ratio (SINR) of nearby eNodeBs, and it connects to the eNodeB c that results in the highest SINR considering that:

$$SINR_{u,c} = \frac{P_c L_{u,c}}{N + \sum_{i=1,\ i\neq c}^{C} (P_i L_{u,i})} \tag{1}$$

$$c = \arg\max_{i\in[1,C]} SINR_{u,i}, \tag{2}$$

where P_c is the transmitted power of eNodeB c, N is the noise power at user u, and $L_{u,c}$ is the path loss that affects the signal transmitted by eNodeB c and received by user u. Additionally, every user u is served by its serving eNodeB c with a data rate of $B_{u,c}$ and $K_{u,c}$ physical resource blocks (PRBs) such that:

$$K_{u,c} = \frac{B_{u,c}}{\delta(SINR_{u,c}) B_{PRB}}, \tag{3}$$

where δ denotes to the achievable spectrum efficiency which is a function of the measured SINR, and B_{PRB} denotes the bandwidth of one PRB which is equal to 180 KHz in LTE.

B. eNodeB utilization and Overload

According to the discussed system model dynamics, every eNodeB serves a number of users $U_c \in [1, U]$ with a total number of physical resource blocks $\sum_{i=1}^{U_c} K_{i,c}$. The Utilization of the eNodeBs can then be calculated as:

$$\rho_c = \frac{\sum_{i=1}^{U_c} K_{i,c}}{B_c / B_{PRB}}, \tag{4}$$

where ρ_c is the utilization of eNodeB c, and B_c is the bandwidth of eNodeB c. The numerator of this equation represents the number of required PRBs from the eNodeB, and the denominator represents the maximum number of PRBs that can be offered by the eNodeB. So, as long as the total PRBs required to serve the users at their Guaranteed Bit Rates (GBRs) are less than the total PRBs that the eNodeB can provide, the eNodeB is considered under-utilized $\rho_c < 1$. And in this case, users are guaranteed to be served with data rates that are equal or higher than their GBRs. However, when the total PRBs required to serve the users at their GBRs are more than the total PRBs that the eNodeB can provide, the eNodeB is considered over-utilized $\rho_c > 1$. And in this case, some users are served with data rates that are less than their GBRs leading to their dissatisfaction. Hence, the maldistribution of loads can negatively affect the throughput of the LTE system leading to a large number of dissatisfied users despite the availability of unused PRB resources in the system.

C. Load Balancing

The main goal of this paper is to introduce a solution to the problem of the load maldistribution among base stations, or eNodeBs, in the network by controlling the handover. The handover of a user from an eNodeB to another in 4G LTE happens when the following condition holds:

$$M_j + \theta_{j-i} > Hys + M_i + \theta_{i-j}, \tag{5}$$

where eNodeB i is the eNodeB currently serving the user, and eNodeB j is the neighbor eNodeB. M_i and M_j refer to the measured values of the Reference Signal Received Power (RSRP) from eNodeBs i and j respectively. The CIO value of the eNodeB i with respect to the eNodeB j is θ_{i-j}, and the CIO of the eNodeB j

with respect to the eNodeB i is θ_{j-i}. A hysteresis value Hys is used to minimize the probability of occurrence of the ping-pong scenario, where a user keeps requesting handover due to small-scale fluctuations in the signal quality. The CIO values impact how the user perceives the RSRP of the serving eNodeB, or any neighboring eNodeB. Unlike the other load balancing techniques, we introduce more control over the handover parameters in equation 5. The actions taken by the agent affect θ_{i-j} and θ_{j-i}, as well as, M_i and M_j. Moreover, we introduced relative CIO values. The relative CIO value of the eNodeB i with respect to eNodeB j is the difference $\theta_{ij} = \theta_{i-j} - \theta_{j-i}$. Whereas, the relative CIO value of the eNodeB j with respect to eNodeB i is $\theta_{ji} = -\theta_{ij}$. As a result, each serving eNodeB has a single CIO value for each neighbor eNodeB, which is different from [6], where only one CIO was used for all neighboring eNodeBs. Another advantage is that the action space of the agent is expected to be relatively smaller than the case when non-relative CIOs are used.

As we will see in the next section, we assume the network has a central RL agent that takes actions according to a predefined KPIs and controls the relative CIO values and the transmission power of each eNodeB. The interaction between the agent and the environment, i.e., the network, happens at discrete time instances with a constant duration between two consecutive interactions, or time steps. We define the state of the network, $S(t)$, as a subset of the network KPIs at time t. In addition, the actions of the central agent, $A(t)$, are defined to be the CIO and transmission power values at time t. The state of the network changes after the actions of the central agent to be $S(t+1)$ and the central agent receives a reward $R(t+1)$.

III. Proposed Model

Two reward functions have been considered in this paper. Each one of them has been used by the RL agent to optimize for in order to know which one of these two reward functions is the best fit for our proposed load balancing model. The two considered reward functions are:

1) Instantaneous Sum Throughput,

$$R(t) = \sum_{c=1}^{C} \sum_{k_c=1}^{K_c} \hat{R}_{k_c}(t), \tag{6}$$

where $\hat{R}_{k_c}(t)$ is the measured value of the actual throughput, at time t, of the k_cth user in the nth eNodeB.

2) Average Deviation of RBs Utilization,

$$R(t) = -\sum_{c=1}^{C} \mid \rho_c(t) - \frac{1}{C}\sum_{c=1}^{C} \rho_c(t) \mid, \tag{7}$$

where ρ_c is the RB utilization of the eNodeB c, and C is the total number of eNodeBs.

It should be noted here that maximizing the sum throughput is a more reasonable reward function than just balancing the RB utilization across the cells in the network. The sum throughput reward function not only guarantees some load balancing but also allows for increasing the data rates of the users, which enhances the users' quality of experience (QoE). Using RB utilization as the reward function can cause users to be connected to less favorable cells although their best cells are not congested.

Every environment step, the central agent takes an action $A(t) = a$ after receiving state $S(t) = s$, according to a stochastic policy $\pi_t(a|s)$. The central agent's main goal is to maximize the long-term average reward function

$$\max_{\pi} \lim_{T\to\infty} \mathbf{E}\left[\frac{1}{T}\sum_{t=0}^{T} R(t)\right], \tag{8}$$

where π is the agent policy.

A. Reinforcement learning

In this section, we discuss the RL method used to tackle the problem of load balancing. This technique is Q-learning using a Double Deep Q-Network (DDQN) with an experience replay. The central agent is going to learn an approximate version of the Q-table using two neural networks. The purpose of the existence of the two neural networks is to avoid the action values overestimation that usually happens when the traditional DQN is used. This overestimation, results from using the same network to select the action and to estimate its value, which might lead to a low-quality policy in addition to unstable learning [9]. As a result, one network, the base network, is used to determine the action of the next state. While the other network, usually called the target network, is responsible for estimating the Q-value of the selected action. On the other hand, adding experience replay to the conventional Deep Q-Learning method allows to achieve better convergence properties [10].

We will start by defining the state $S(t)$ as the concatenation of the three key performance indicators (KPIs). The first KPI is the Resource Block Utilization (RBU), an C-length vector $U(t) \in [0,1]^C$. Each element in $U(t)$ represents the fraction of the utilized RB in the cth eNodeB at time t. The fraction of utilization is a measure of the congestion level in the eNodeB. The total downlink (DL) throughput in the cth eNodeB at time t is the second KPI, represented by an element in the vector $R(t) \in \mathbb{R}_+^C$, where $\mathbb{R}_+$ denotes the set of positive real numbers. The DL throughput is a measure of the overall performance of the eNodeB. The MCS utilization is the third KPI in the state function $S(t)$, for which there are 29 currently defined MCSs in LTE [3]. MCS utilization can be represented by a matrix $M(t) \in [0,1]^{C\times\mu}$, where μ is the 29 modulation and coding schemes. Each element in $M(t)$ represents the ratio of users with a particular MCS, providing a metric for the relative channel qualities of the users. After vectorizing $M(t)$ using the vectorization function $vec(.)$, we can define the state function as the concatenation of all three KPI vectors as

$$S(t) = [U(t)^T \quad R(t)^T \quad vec(M(t))^T]^T. \tag{9}$$

Therefore, the input layers of the two neural networks will be of size $2C + C \times \mu$.

The central agent takes an action $A(t)$ controlling the values of the relative CIOs and the transmission energy of each eNodeB. The actions are chosen from a predefined set of offsets that are then added to a default value. The relative CIO, $\theta_{cn}(t)$, which is equivalent to $-\theta_{nc}(t)$, can take discrete values from the set [-θ_{max}, θ_{max}] dB of size L, where $c \in C$ (the number of eNodeBs in the network), and $n \in N$ (the number of neighbor eNodeBs to the cth eNodeB). θ_{max} is the maximum value a relative CIO can take. Likewise, the central agent chooses the transmission energy E_c of the cth eNodeB from a set $[-E_{max}, E_{max}]$ dB of size L. The size of the action space is H, the total number of the relative CIOs according to the topology, plus the total number of the eNodeBs C, i.e.,

$$A(t) = [\theta_{12}, \cdots, \theta_{23}, \theta_{24}, \cdots, \theta_{CN}, E_1, E_2, \cdots, E_C]^T. \tag{10}$$

The total number of possible actions in the action space equals L^{C+H}. Both neural networks are used as a multi-class classifier; hence, the output layer of each of the neural networks consists of L^{C+H} output neurons with a linear activation function. Additionally, we add a hidden layer of size $2C+C\times\mu$ neurons. Both the input and hidden layers in each network have a rectified linear (relu) activation function.

The simulation runs for a number of N_e episodes. Each episode corresponds to a complete simulation of the NS-3 environment. The episode duration is divided into time steps Δ. The action selection is performed using an epsilon greedy policy, where a random action A is selected with probability $0 < \epsilon < 1$.

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

This probability, $\epsilon(t)$, is important to balance exploration-exploitation. At the beginning, $\epsilon(t) = 1$ and thus the agent begins to explore new random actions. The probability $\epsilon(t)$ is decaying with time, which indicates more exploitation and less exploration until it reaches a minimum value when the exploration is no longer needed.

As we mentioned before, experience replay is used in our model. The experience replay saves all the actions, states, and rewards in a replay memory of a limited length. When this memory is full, the old experience is overwritten. At each step, the central agent picks a number of random samples, a mini-batch, from this memory to train and update the weights of the base network. On the other hand, updating the weights of the target network is delayed for a specific period (an episode in our case). Because of the great correlation between the samples of consecutive time steps, using mini-batches is essential for our problem. Thus, within each batch, the samples can be considered to be Independent and Identically Distributed (IID) and the double estimator is unbiased [9].

The action values of the target network are used to compute the action values in the base network as

$$\begin{aligned} Q_{t+1}(S(t), A(t)) &= R(t+1) \\ &+ \gamma Q_{target}(S(t+1), \arg\max_A Q_t(S(t+1), A; \mathbf{w_t}); \mathbf{w'_t}) \end{aligned} \tag{12}$$

where γ is the future rewards' discount factor, and $\mathbf{w_t}$ and $\mathbf{w'_t}$ are the two sets of weights of the action selection (base) network and the evaluation (target) network, respectively, [10].

IV. Numerical Analysis

In this section, we present our simulation network setup, the RL model, and our simulation results and analysis.

A. Network setup

The LTE network scenario, shown in Fig. 1, consists of 3 eNodeBs with inter-site distance of 500 meters. Each eNodeB site covers a hexagonal area using a single omni-directional antenna. Respectively, the eNodeBs are labeled eNodeB 1, eNodeB 2 and eNodeB 3 from the left to the right. Within each eNodeB coverage, a number of UEs are assumed to be stationary and are deployed randomly with low density. Thus, all of those users are satisfied and the network does not suffer from any kind of congestion. To add a hotspot congestion to the network, a bus that contains a number of users are moving with constant speed starting at the left, eNodeB 1, and moving towards eNodeB 3 as the simulation progresses. The simulation is carried out using the NS-3 simulator [11]. Table I summarizes the parameters used in the simulations and their corresponding values. The environment step time is the time for which the network experiences updates until the RL agent generates new actions (i.e., the length of one RL time step in seconds). As the mobile UEs move from the vicinity of eNodeB 1 to eNodeB 2 and eNodeB 3, A3 event (handover event) triggers as the UEs satisfy the handover condition illustrated in equation 5.

B. RL model

After setting up the network on the NS-3 network simulator, we developed our RL agent that is supposed to use this network as its environment. The RL agent, which uses the DDQN and experience replay to learn the best control of this environment, is implemented using Python and Tensorflow [12]. After setting up both the network environment and the agent, we have used NS3-gym as an interface that enables the simulated network on NS-3 to be used as an environment for the RL agent. The complete model is shown in Fig. 2. The simulation runs for 200 episodes each consisting of 100 time-steps. To select the hyper-parameters that result in the best performance of the agent, the simulation is repeated several times. The final values of the hyper-parameters used in the RL agent are listed in Table II.

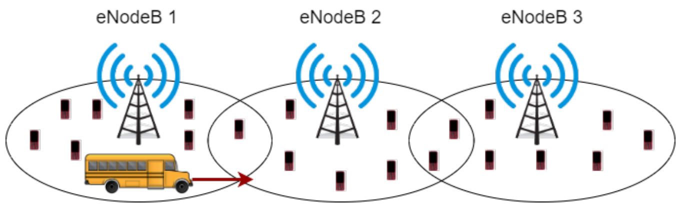

Fig. 1: Simulated LTE network topology

C. Results and discussion

In this sub-section, we present and analyze the numerical results of testing our proposed model on the small network discussed earlier in this section. The results of this model which employs both CIO and energy adjustments are, then, compared with the results of the model proposed in [6] which employs only CIO adjustments and the results of the base-line model which does not apply any adjustments. It should be mentioned that the presented curves are smoothed to reduce the effect of fluctuations and illustrate the trends.

Figure 3 shows the learning process of each of the three models. Each model aims at maximizing the sum throughput of the network in different ways. Our model maximizes the total throughput via controlling both the relative CIO values and the transmitted power of the different eNodeBs in the network. Any relative CIO can take a value of -3, 0, or 3, and any eNodeB's transmitting power can take a value of 40dbm, 43dbm, or 46dbm (which are typical transmission power values). The second model maximizes the total throughput by

Parameter	Value
Inter-site Distance	500m
Center Frequency	2GHz
# of eNodeBs	3
eNodeB Bandwidth	50MHz
# of Antennas per eNodeB	Tx: 1, Rx: 1
Antenna Pattern	Omni
eNodeB antenna height	30m
eNodeB Tx Power	43dBm
Shadowing	No
Traffic Direction	Downlink
Traffic Model	Full buffer
Scheduler	RrFfMacScheduler
UE Mobility Model	Mixed 20 stationary UEs 10 moving UEs (constant speed of 20 m/s)
UE Antenna Height	2m
Handover	A3-event based Time to trigger = 40ms Hysteresis = 3dB.
Environment step time	0.2s

TABLE I: Summary of network parameters

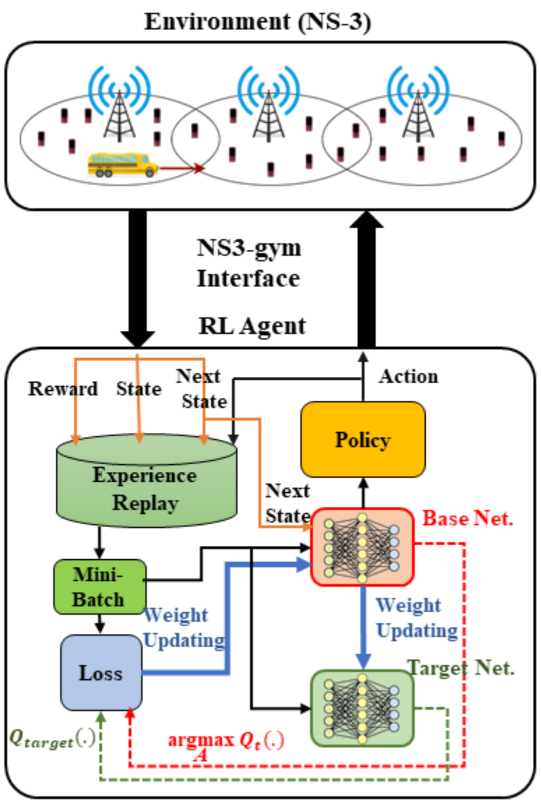

Fig. 2: The proposed model

Hyper-parameter	Value
CIO set	$\{-3, 0, 3\}$dB
Tx power set	$\{40, 43, 46\}$dBm
Reward function	Total DL throughput
Number of episodes	200
Number of iterations/episode	100
ϵ decay	0.999
gamma (discount factor)	0
Number of hidden layers (N_h)	1
Activation function(input/hidden layers)	relu
Activation function (output layer)	linear
Loss function	Huber loss
Optimizer	Adam(0.001)

TABLE II: Simulation parameters of the RL agent

controlling only the CIO values (-3, 0, or 3) of the eNodeBs in the network. And the base-line model uses fixed CIO values, set at zero, and fixed transmission power of 43dbm for all the eNodeBs in the network. The x-axis of the graph in Fig. 3 represents the number of episodes that the learning process has elapsed. Every episode is a complete network simulation of 20 seconds. The y-axis represents the total throughput of the three eNodeBs in the network averaged over all steps of the corresponding episode. As shown in Fig. 6, the average episode throughput of the base-line model is constant during the learning process, which is expected. The CIO model's throughput reaches 12.44 Mb/s. On the other hand, our proposed model takes longer to converge due to the extended action space after considering controlling the energy in addition to the CIOs. However, the proposed model records a much better throughput of 13.24 Mb/s. As a result, the CIO model achieves an increase of 14% in the instantaneous sum throughput of the network, while our proposed model achieves an increase of 21.4% in the instantaneous sum throughput compared to the base-line.

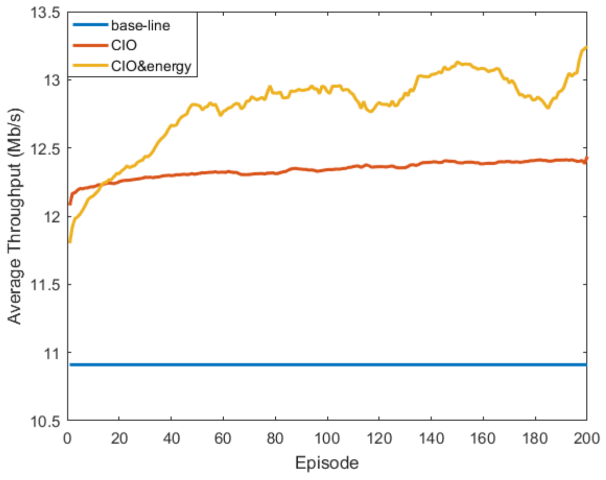

Fig. 3: Learning process of the different models

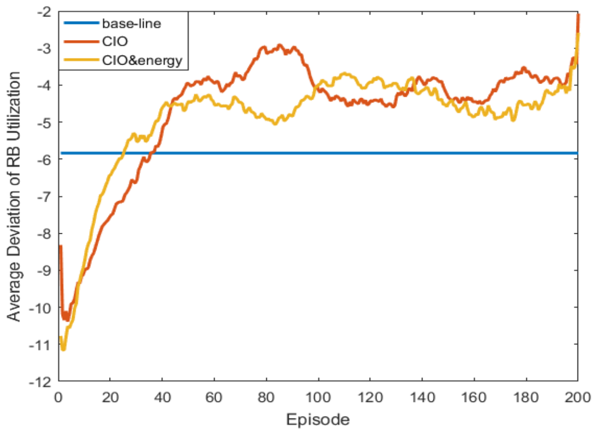

Fig. 4: Average RB utilization deviation under the different models

It should be noted that our model can be extended to account for other reward functions. Another reward has been tried by the RL model which is the average deviation of the network's RB utilization. Figure 4 shows the learning process of the RL agent to achieve the minimum deviation in order to guarantee a fair distribution of the users among all the eNodeBs of the network. As illustrated in the figure, both the RL agents of the proposed model and the CIO model perform very similarly using this reward. Both of them converge to a very similar value which is better than the value of the base-line model. This can be attributed to the fact that the RB utilization is mostly affected by the handovers of the users, which can be merely controlled by adjusting the CIOs. However, if the aim is to maximize the sum throughput, then the eNodeBs energies will play an important role; this is the reason why controlling both the CIOs and the energies resulted in an improved throughput performance as compared to the other two systems, where one only adjusts the CIOs and the other does not apply any adjustments (i.e., the base-line system)

After completing the learning process, five episodes (starting from different randomization seeds) have been used to evaluate the performance of the three models. Figure 5 shows the average total throughput achieved by each model during each episode. In this

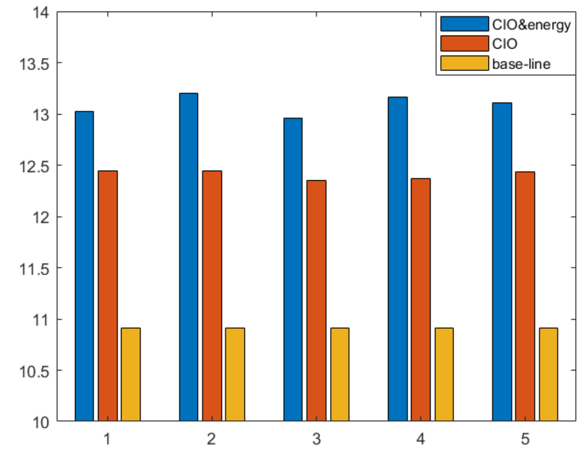

Fig. 5: Episode-Level comparison after convergence between the different models

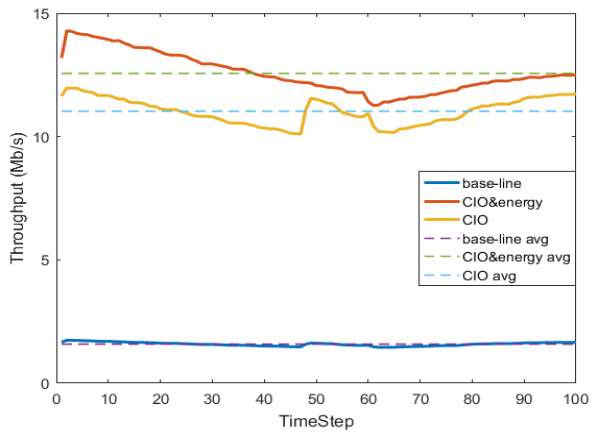

Fig. 6: Step-Level comparison after convergence between the different models

figure, it can be seen that the performance of our proposed load balancing model significantly outperforms both the base-line and the CIO based models. In addition, Figure 6 shows the performance of our proposed model throughout one complete testing episode. The graph in Fig. 6 shows that not only the average episode throughput of the proposed model exceeds the base-line average throughput, but also our model throughput exceeds the base-line throughput on the level of each step, i.e. for every possible state of the network. That being said, controlling the Tx power of every eNodeB in the network along with the CIOs adds a decent performance gain to the load balancing model. This is because it not only guarantees the balancing of the load, which is not an aim itself, but also allows increasing the data rates of the users. This is done through controlling the transmitted power of the eNodeBs and, thus, the interference among the different eNodeBs.

V. Conclusion

Mobile communication networks have ubiquitous existence in our daily modern lives. Congestion of users in mobile networks is a problem that is addressed in several research papers. This paper have introduced controlling both the CIO value and the transmission power of the eNodeBs to control the handover of users. DDQN with experience replay is used to find the values of the CIO and transmission power of different eNodeBs in the network to maximize some network reward function (e.g., sum throughput). Relative CIO values are used instead of absolute CIO to eliminate any restriction to achieve optimal performance. The results of the simulation shows that controlling relative CIO and transmission power of the eNodeBs to perform load-balancing significantly enhances the throughput of the network, compared to the base-line or when only the CIOs are adjusted.

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] TS ETSI. Lte: Evolved universal terrestrial radio access (e-utra), physical layer procedures-corresponding to 3gpp ts36 213. *3GPP TS*, 136(213):V10.

[4] 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, 2013.

[5] Andreas Lobinger, Szymon Stefanski, Thomas Jansen, and Irina Balan. Load balancing in downlink lte self-optimizing networks. In *2010 IEEE 71st Vehicular Technology Conference*, pages 1–5. IEEE, 2010.

[6] Kareem Attiah, Karim Banawan, Ayman Gaber, Ayman Elezabi, Karim Seddik, Yasser Gadallah, and Kareem Abdullah. Load balancing in cellular networks: A reinforcement learning approach. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2020.

[7] Sameh Musleh, Mahamod Ismail, and Rosdiadee Nordin. Load balancing models based on reinforcement learning for self-optimized macro-femto lte-advanced heterogeneous network. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(1):47–54, 2017.

[8] Yue Xu, Wenjun Xu, Zhi Wang, Jiaru Lin, and Shuguang Cui. Load balancing for ultradense networks: A deep reinforcement learning-based approach. *IEEE Internet of Things Journal*, 6(6):9399–9412, 2019.

[9] Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[10] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[11] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[12] Martín Abadi *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.