Hierarchical Multi-Agent Reinforcement Learning Framework for Cellular Mobility Load Management

Aamen Elgharably[1], Mariam Aboelwafa[2], Karim Banawan[1,3], and Karim G. Seddik[1]

[1]The American University in Cairo, Egypt
[2]NewGiza University, Egypt
[3]Alexandria University, Egypt

Abstract—**The increasing complexity and density of modern networks necessitate advanced, AI-driven solutions to manage traffic efficiently and maintain high-quality service. In this paper, we present a novel reinforcement learning (RL) framework designed to optimize handover parameters for load balancing in cellular networks. Our framework adopts a *hierarchical multi-agent* RL approach. Closely adjacent cells (a.k.a., cluster) are controlled by *cluster-level* agents, whereas inter-cluster parameters are controlled by a *network-level* agent. By intricate design of state spaces and agent communication, both cluster-level and network-level agents work collaboratively to enhance network performance in terms of throughput and coverage. This method reduces the action and state spaces for each agent, facilitating faster learning, scalable network-wide control, and more efficient decision-making. Our simulation results demonstrate significant improvements in downlink throughput with respect to fully decentralized agents. Our approach incurs negligible throughput loss when compared to a fully centralized agent with full knowledge of the entire network. Our approach not only achieves scalable load balancing with minimal overhead but also allows for customizable reward functions tailored to different network needs.**

Index Terms—**Reinforcement Learning, Cellular Networks, Self-Optimized Networks, Mobility Load Balancing**

I. Introduction

In the continuous evolving of cellular communications networks, enhancing the efficiency and scalability of network management has become of paramount importance. With the advancement of 5G networks and the anticipation of 6G and beyond future networks, the complexity and density of eNodeBs (eNBs) have significantly increased. Efficiently managing these elements, particularly in terms of mobility load management, is critical for maintaining quality of service and optimizing network performance.

From another perspective, the future of cellular networks is envisioned to be AI-native, that is integrating advanced artificial intelligence and machine learning techniques at their core. As network complexity continues to escalate with the rollout of 5G and the approaching 6G era, traditional manual network management methods are becoming increasingly inefficient. AI-native networks promise to revolutionize the way cellular systems are managed by enabling real-time, autonomous decision-making processes that can adapt to dynamic network conditions. These networks adopt AI-based models to optimize resource allocation and mitigate potential issues before they impact service quality. This shift towards AI-driven management is essential to meet the growing demands for higher data rates, lower latencies, and enhanced user experiences.

In this work, we consider the problem of mobility load management in cellular networks. Specifically, we aim to enhance the downlink throughput, optimize resource block utilization, and minimize the number of uncovered users (i.e., enhance network coverage). Our previous works in [1]–[5] have addressed the load balancing challenge in cellular networks using various techniques, however, the proposed approaches had inherent limitations in scaling efficiently. This is due to the nature of the centralized agent that controls all cells concurrently. This motivates our efforts to investigate a *scalable* approach that 1) has a manageable dimensionality of state and/or action spaces, 2) Performs mobility load management with negligible performance loss compared to a centralized agent that may be challenging to implement, and 3) has a limited communication overhead between agents.

To that end, we propose a novel *hierarchical multi-agent reinforcement learning* (RL) framework for mobility load management in cellular networks by dynamically adjusting the relative cell individual offset (CIOs). CIOs control the handover thresholds between eNBs, thus achieving a more adaptive and efficient handover process. In this framework, the network is *clustered* such that each cluster comprises multiple adjacent eNBs. The network is divided into a number of clusters to balance the load on central agents and minimize action latency. There exist two levels (i.e., hierarchical) of RL agents, namely, the cluster-level agents, and the network-level agent. The proposed framework introduces a multi-agent RL approach, where both cluster-level and centralized agents collaborate to optimize the network's performance. We design the state and action spaces of all agents in addition to identifying a flexible reward function that fits various network needs. Moreover, we pinpoint the intra- and inter-cluster CIOs as sufficient information that can be communicated between agents to boost their performance.

Our proposed hierarchical multi-agent architecture significantly reduces the action space and state space for each agent, facilitating faster and more efficient learning and decision-making processes. This in turn addresses the aforementioned scalability issues in [1]–[5], offering a robust solution capable of managing the increasing complexity and density of modern cellular networks.

The results demonstrate the efficacy of the RL-based approach in improving network performance compared to fully decentralized methods as in [6]. Our approach incurs negligible loss compared to traditional centralized agents as in [1], which may be infeasible to implement in hyper-dense networks.

A. Related Work

Load-balancing techniques have been extensively explored in the literature. In works such as [7] and [1], the authors designed an RL framework to optimize cell parameters for balancing traffic loads across cells. These studies focused on controlling the CIOs of neighboring cells, prompting cell-edge users to hand over from congested cells to those with lighter loads. Additionally, [2] and [3] introduced an RL agent that manages both eNB transmission power and CIOs to achieve traffic load balancing, aiming to maximize downlink sum throughput while minimizing the number of uncovered users.

From another perspective, [8] explores a deep RL approach for coordinating inter-cell interference, specifically controlling power in cellular networks to maximize the network's sum rate. Furthermore, [9] presents an RL-based downlink scheduling approach that autonomously manages active traffic flows.

Works like [1]–[4], [7] have utilized RL to achieve load balancing in networks, ultimately aiming to maximize sum throughput. [10] provides a comprehensive survey on the applications of deep RL in cellular networks.

Proposing multi-agent models also exists in literature. For instance, in [6], the authors propose an approach such that each cell has a decentralized agent to learn handover parameters and antenna tilt angle. For a more comprehensive view, the study in [11] provides a detailed exploration of multi-agent RL and its related fields and the study in [12] has a narrower focus on surveying multi-agent RL for communication networks. Another survey is presented in [13] that reviews basic methods and use cases for multi-agent RL and outlines the trending research areas and their limitations.

B. Paper Contribution

The contributions of this paper can be summarized as follows:

- An RL-based framework that optimizes handover parameters to achieve load balancing across cellular networks. The framework efficiently manages the distribution of users among eNBs to prevent congestion and underutilization.
- A hierarchical multi-agent RL approach where both cluster-level and centralized agents collaborate. This approach reduces the action space and state space for each agent, facilitating faster learning and decision-making processes with negligible throughput loss compared to a fully centralized agent, which may be challenging to implement in realistic networks.
- The proposed framework shows scalability with minimal overhead by sharing limited information between agents. Specifically, the proposed scheme only shares intra- and inter-cluster CIOs between the cluster-level and network-level agents. It also allows for the customization of different rewards for each agent, enabling tailored optimization goals for various operational scenarios.
- Employing a realistic simulation environment using the NS3 simulator and the SUMO mobility model. This setup ensures that the proposed framework is evaluated under practical conditions, reflecting real-world network scenarios.

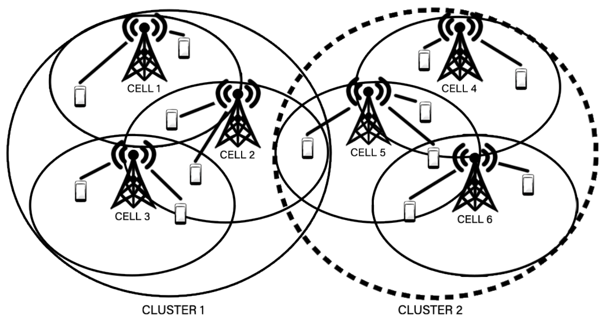

Fig. 1. cellular system clusters

II. System Model

We consider a cellular system (see Fig.1) that serves U user equipment (UE) in total. The cellular system consists of N base stations (a.k.a., eNBs in 4G or gNBs in 5G networks). In this work, we assume that the cellular network is divided into M clusters to facilitate network-control scalability. In the sequel, we present the system components in detail.

A. Base Stations (eNBs)

Each eNB sends its downlink transmission with power P_t dBm. The nth eNB is allocated a total bandwidth of B_n Hz. We assume that our cellular system operates with an OFDM-based air interface (which is the case in 4G and 5G, and envisioned for 6G) [14], [15]. Specifically, the eNB allocates transmission resources in the form of physical resource blocks (PRBs), each having a bandwidth of B_{PRB} Hz (e.g., $B_{\text{PRB}} = 180$ KHz in LTE [16].

The nth eNB serves U_n UEs. The ith UE belonging to the nth eNB is allocated $K_{i,n}$ PRBs. Consequently, the eNB utilization ρ_n, i.e., the ratio of the total PRBs required to serve the attached users to the total PRBs that the eNB can offer, can be written as:

$$\rho_n = \frac{\sum_{i=1}^{U_n} K_{i,n}}{B_n/B_{PRB}} \tag{1}$$

The utilization ρ_n measures the congestion level of the cell. Thus, $\rho_n \ll 1$ represents an under-utilized cell, while $\rho_n > 1$ is over-utilized (i.e., infeasible resource allocation).

Moreover, the cells are endowed with settable cell individual offsets (CIOs). A CIO is an offset power level that is used to control the users' handover procedure. Specifically, a positive

CIO $\theta_{i \to j} \in [\theta_{\min}, \theta_{\max}]$ dB implies that the RSRP from ith cell appears stronger than the RSRP from the jth cell by $\theta_{i \to j}$ dB. This discourages users in the ith cell from executing the handover procedure to the jth cell, and vice versa [17].

B. User Equipment (UEs)

The kth user moves with a velocity v_k m/s. Periodically each UE measures the signal-to-interference-plus-noise-ratio (SINR) of nearby eNB and sends an attachment request to eNB with the highest measured SINR.

The kth UE regularly searches for a better cell by measuring the reference signal received power (RSRP). Specifically, the RSRP measurements initiate a *handover* procedure for the kth user from the mth cell from the nth cell if the following criterion is true [17]:

$$\text{RSRP}_m + \theta_{m \to n} > \text{Hys} + \theta_{n \to m} + \text{RSRP}_n \quad (2)$$

where RSRP_i is the measured RSRP from the ith cell, Hys is the hysteresis power level used to reduce the ping-pong handover effects, and $\theta_{i \to j}$ corresponds to the relative CIO between the mth and the nth cells.

Moreover, the kth UE reports the channel quality indicator (CQI), denoted by ϕ_k. CQI is a discrete measure of the channel quality such that $\phi_k \in \{0, 1, \cdots, 15\}$. Specifically, $\phi_k = 0$ means that the kth UE is out of coverage, while higher CQI indicates a more favorable channel quality [18].

C. Hierarchical Control Agents and Clustering

In this work, we assume that the N cells of the cellular network are divided into M clusters. The mobility load management is performed using a *hierarchical* control agents. Specifically, each cluster is managed by a *cluster-level control agent*. The cluster-level agent has access to all key performance indicators (KPIs) of all cells belonging to such a cluster. In addition, the cluster-level agent manages the control knobs within the cluster (intra-cluster controls). The cluster-level agent cannot influence controls beyond the cluster or inter-cluster controls.

On the other hand, the cellular network is equipped with a *network-level control agent* that has access to *aggregate* KPIs from all clusters. The network-level agent manages inter-cluster controls only. Moreover, we assume that there exists a *backhaul* network connecting the cluster-level agents with the network-level agent that can convey *low-rate* inter-cluster information.

In this work, we assume that the way the network is clustered is chosen based on their physical location where nearby base stations are clustered together[1].

III. Problem Formulation

In this paper, we aim to design both cluster-level agents and network-level agent for mobility load management of the cellular network such that they simultaneously 1) maximize the long-term average sum throughput of the cellular network, and 2) minimize the probability of having out-of-coverage users.

Denote the instantaneous throughput at time $t = 0, 1, 2, \cdots$ of the nth cell by $R_n(t)$. Consequently, our objective is to maximize the following scalarized multi-objective function:

$$\lim_{L \to \infty} \mathbb{E} \left[\sum_{t=1}^{L} \eta^t \left(\sum_{n=1}^{N} R_n(t) - \lambda \bar{R}(t) \sum_{u=1}^{U} \mathbf{1}(\phi_u(t) = 0) \right) \right] \quad (3)$$

where η is the discount factor, which signifies the weight of future decisions, λ is the penalty factor of the throughput that signifies how important the coverage requirement compared to the throughput metric, and $\bar{R}(t)$ is the average user throughput, i.e., $\bar{R}(t) = \frac{1}{U} \sum_{n=1}^{N} R_n(t)$. Motivated by previous works in [1]–[3], we employ the reinforcement learning (RL) technique for mobility load management.

The aforementioned objective is maximized subject to the following design criteria (in addition to the hierarchical clustering structure in Section II-C): 1) The inter-cluster information exchanged through the backhaul links is limited. This is motivated by the need to have *manageable state dimensionality* to facilitate scalable learning, and 2) The state dimensionality of the network-level agent is no more than that of its counterpart of the cluster-level agents. In the following section, we present our hierarchical Multi-Agent Reinforcement Learning Framework in detail.

IV. Hierarchical Multi-Agent Reinforcement Learning Framework

In this section, we start by describing our proposed hierarchical multi-agent RL framework. We give an overview of the approach followed by the details of the state space, action space, and reward function for all agents.

A. Overview of the Proposed Approach

The scalarized multi-objective optimization problem in Section III is recast as a multi-agent RL problem. Specifically, the CIOs belonging to one cluster (intra-cluster CIOs) are controlled by the cluster-level RL agent, while the CIOs within multiple clusters (inter-cluster CIOs) are controlled via the network-level RL agent. This reduces the action space dimensionality of the agents, which in turn enables scalable extension of [1] without inflating the convergence interval.

To reduce the state space dimensionality (to enhance convergence rate), the cluster-level agent observes *all KPIs* belonging to the cluster. In addition, the cluster-level agent observes the inter-cluster CIO from the network-level agent. The network-level agent, however, observes *aggregate KPIs* from all clusters, i.e., it observes a simplified state of the overall network. The network-level agent observes the intra-cluster CIO values from all clusters. Thus, we adopt the CIO values as *communication messages* between the agents (see Fig. 2).

In this work, we choose the *Twin Delayed Deep Deterministic Policy Gradient (TD3)* implementation for all RL agents (cluster-level and network-level). TD3 is an actor-critic RL agent, which is a direct successor of the DDPG

[1]Exploring other clustering approaches is an interesting future direction of this work that is outside the scope of this paper.

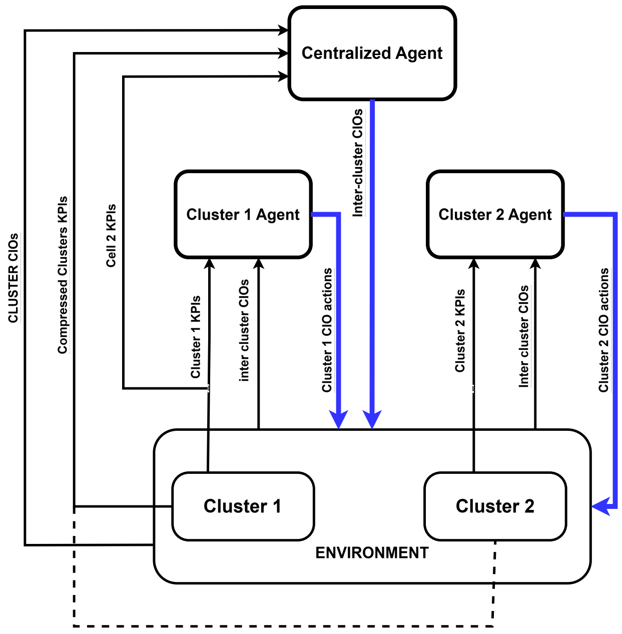

Fig. 2. RL Model

agent. TD3 works with continuous action spaces, reduces the over-estimation effects of the DDPG, and shows more stable convergence [19].

B. State Space

The state of the agent is a compact representation of the cellular network at a given instant. The agent chooses its action as a function of the observed state space. In this work, we use readily available KPIs, which are periodically reported in the cellular network as in [1]. This is in addition to the communicated CIO values.

1) Cluster-Level Agents: For the cluster-level agents, we choose the following KPIs for the state. These KPIs correspond to cells' information within the cluster only. The KPIs themselves remain exactly as in [1]. Specifically,

- Cell total downlink (DL) throughput: This is the sum of all users' downlink throughput in each cell. Denote $R_n^{[\ell]}(t)$ as the total downlink throughput of the nth cell belonging to the ℓth cluster at time t, then, $R_n^{[\ell]}(t) = \sum_{u_n=1}^{U_n} R_{u_n}(t)$. The following vector represents the total throughput from all cells of the ℓth cluster:
$$\mathbf{R}^{[\ell]}(t) = \begin{bmatrix} R_1^{[\ell]}(t) & R_2^{[\ell]}(t) & \cdots & R_{N_\ell}^{[\ell]}(t) \end{bmatrix} \quad (4)$$
where N_ℓ represents the total number of the cells in the ℓth cluster.
- Resource block utilization (RBU): It is a vector consisting of the PRB utilization $\rho_n^{[\ell]}$ at time t (see equation (1)) of the nth cell belonging to ℓth cluster, i.e.,
$$\boldsymbol{\rho}^{[\ell]}(t) = \begin{bmatrix} \rho_1^{[\ell]}(t) & \rho_2^{[\ell]}(t) & \cdots & \rho_{N_\ell}^{[\ell]}(t) \end{bmatrix} \quad (5)$$
The RBU measures the level of congestion in the cells belonging to a certain cluster.
- Number of connected users: It is a vector corresponding to the number of UEs connected to eNBs (i.e have been allocated a DL traffic channel) within the ℓth cluster at time t, which is denoted by $\mathbf{U}^{[\ell]}(t)$,
$$\mathbf{U}^{[\ell]}(t) = \begin{bmatrix} U_1^{[\ell]}(t) & U_2^{[\ell]}(t) & \cdots & U_{N_\ell}^{[\ell]}(t) \end{bmatrix} \quad (6)$$
$\mathbf{U}^{[\ell]}(t)$ and $\mathbf{R}^{[\ell]}(t)$ vectors signify the average throughput per UE in each cell.
- Modulation and coding scheme (MCS) penetration: It is a matrix $\mathbf{M}^{[\ell]}(t)$, whose rows correspond to the lower μ MCS schemes, and columns denote the eNB within the cluster, i.e., the element $M_{j,n}^{[\ell]}(t)$ denotes the fraction of UEs employing the jth MCS in the jth cell belonging to the ℓth cluster at time t. MCS penetration reflects the average channel quality of the UEs.
- Signaling from the network-level agent: In addition, the cluster-level agent receives the inter-cluster CIO values from the network-level agent, i.e., it receives a vector $\boldsymbol{\Theta}(t) = [\theta_{i\to j} : i \in C_{\ell_1}, j \in C_{\ell_2}, \ell_1 \neq \ell_2]$, where C_ℓ is the set of cells belonging to the ℓth cluster.

Consequently, the state of the cluster-level agent is a concatenation of all these vectors, i.e.,
$$S^{[\ell]}(t) = \left[\mathbf{R}^{[\ell]}(t) \;\; \boldsymbol{\rho}^{[\ell]}(t) \;\; \mathbf{U}^{[\ell]}(t) \;\; \text{vec}\left(\mathbf{M}^{[\ell]}(t)\right) \;\; \boldsymbol{\Theta}(t)\right] \quad (7)$$
where vec$(\cdot)$ denotes vectorization of matrix operation.

2) Network-Level Agent: The network-level agent bases its action on a state similar to the cluster-level agents with two major differences. First, the network-level agent receives the *aggregate* KPIs of the cluster and not the individual cells' KPIs. Specifically, each cluster passes the *average* KPIs across all eNBs belonging to the cluster. Consequently, the average DL throughput of the cluster vector $\mathbf{R}(t)$ is given by:
$$\mathbf{R}(t) = \left[\frac{\sum_{n_i=1}^{N_i} R_{n_i}^{[i]}(t)}{N_i} : i = 1, 2, \cdots, M\right] \quad (8)$$
i.e., the ℓth cluster passes the average value of $\mathbf{R}^{[\ell]}(t)$ to the network-level agent. Similarly, the the RBU of the cluster vector $\boldsymbol{\rho}(t)$, and the number of connected users of the cluster are given by:
$$\boldsymbol{\rho}(t) = \left[\frac{\sum_{n_i=1}^{N_i} \rho_{n_i}^{[i]}(t)}{N_i} : i = 1, 2, \cdots, M\right] \quad (9)$$
$$\mathbf{U}(t) = \left[\frac{\sum_{n_i=1}^{N_i} U_{n_i}^{[i]}(t)}{N_i} : i = 1, 2, \cdots, M\right] \quad (10)$$
For the MCS penetration, we average the columns of the matrix, i.e., the element $M_{j,\ell}(t)$ of the matrix $\mathbf{M}(t) = [M_{j,\ell}(t)]$ where $j = 1, \cdots, \mu$ and $\ell = 1, \cdots, M$ is given by:
$$M_{j,\ell}(t) = \frac{\sum_{n_\ell=1}^{N_\ell} M_{j,n_\ell}(t)}{N_\ell} \quad (11)$$
Therfore, the network-level agent is essentially dealing with the cluster's KPIs in the same way the cluster-level agent is dealing with cell's KPIs. This implies that the state space

dimensionality of the network-level agent is of a similar dimensionality of the cluster-level agent, which in turn facilitates scalable learning.

The network-level agent receives the intra-cluster CIO values from all clusters as side information to the KPIs, i.e., the ℓth cluster-level agent sends the vector $\mathbf{\Theta}^{[\ell]} = [\theta_{i\to j} : i,j \in C_\ell]$. Thus, the network-level agent adds the following vector to its state representation

$$\tilde{\mathbf{\Theta}}(t) = \Big[\mathbf{\Theta}^{[1]}(t) \quad \mathbf{\Theta}^{[2]}(t) \quad \cdots \quad \mathbf{\Theta}^{[M]}(t)\Big] \tag{12}$$

Therfore, the state of the network-level agent $S(t)$ is as follows:

$$S(t) = \Big[\mathbf{R}(t) \;\; \boldsymbol{\rho}(t) \;\; \mathbf{U}(t) \;\; \mathrm{vec}\left(\mathbf{M}(t)\right) \;\; \tilde{\mathbf{\Theta}}(t)\Big] \tag{13}$$

C. Action Space

The available actions of all agents are the relative CIOs ($\theta_{i\to j} : i,j = 1,2,\cdots,N, i \neq j$). The CIO values are picked from a continuous range $[\theta_{\min}, \theta_{\max}]$ dB. The CIO value controls the handover condition between adjacent cells.

1) Cluster-Level Agents: The ℓth cluster-level agent controls the relative CIOs between two cells (intra-cluster CIOs), which both belong to the cluster, hence, the action of the ℓth agent is given by:

$$A^{[\ell]}(t) = [\theta_{i\to j} : i \neq j,\;\; i,j \in C_\ell] \tag{14}$$

2) Network-Level Agent: The network-level agent controls the relative CIOs between cells belonging to different clusters (inter-cluster CIOs). Specifically, the action of the network-level agent $A(t)$:

$$A(t) = [\theta_{i\to j} : i \neq j,\;\; i \in C_{\ell_1}, j \in C_{\ell_2}, \ell_1 \neq \ell_2] \tag{15}$$

D. Reward Function

The reward function $\mathcal{R}(t)$ of an RL agent assesses the quality of the agent's applied action. In this work, we adopt the *penalized throughput* as a reward function that reflects the throughput and coverage in a unified metric as in [3]. The main difference from [20] is how this reward function is defined for the cluster-level and the network-level agents as we show next.

1) Cluster-Level Agents: The cluster-level agent calculates its reward function based only on the penalized throughput of the cells belonging to the cluster. Hence, the reward function for the ℓth agent $\mathcal{R}^{[\ell]}(t)$ is given by,

$$\mathcal{R}^{[\ell]}(t) = \sum_{n_i=1}^{N_\ell} R_{n_i}^{[\ell]}(t) - \lambda \bar{R}^{[\ell]}(t) \sum_{n_i=1}^{N_\ell} \sum_{j=1}^{U_{n_i}^{[\ell]}} \mathbf{1}(\phi_j = 0) \tag{16}$$

where $\bar{R}^{[\ell]}(t) = \frac{\sum_{n_i=1}^{N_\ell} R_{n_i}^{[\ell]}(t)}{\sum_{n_i=1}^{N_\ell} U_{n_i}^{[\ell]}(t)}$ is the average user's throughput in the ℓth cluster, i.e., the cluster-level agent only needs *local reward information* to adapt its RL parameters. This in turn promotes the scalability of our scheme.

2) Network-Level Agent: The reward function for the network-level agent is calculated with respect to the entire network, i.e.,

$$\mathcal{R}(t) = \sum_{\ell=1}^{M} \sum_{n_i=1}^{N_\ell} R_{n_i}^{[\ell]}(t) - \lambda \bar{R}(t) \sum_{u=1}^{U} \mathbf{1}(\phi_j = 0) \tag{17}$$

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Bandwidth(B_n)	5 MHz
eNB Transmission power (P_t)	30 dBm
eNB antenna height	30 m
eNB antenna pattern	omni
UE antenna height	1.5 - 2 m
Path loss model	Cost Hata
Uncovered Users penalty	2
Training Steps	100,000
Steps per Episodes	250
Step Time	0.2 seconds
MCS μ	{0-10}

V. SIMULATION AND NUMERICAL RESULTS

In this section, we present the simulation parameters used, the evaluation scenario, and the impact of the proposed approach on the network performance compared to the centralized/decentralized approaches.

A. Simulation Setup and Agent Implementation

In this work, we realize the network simulation using the NS3 simulator. NS3 is an open-source discrete-event network simulator. The NS3 includes a fully operational LTE module, which is a software library, that allows the simulation of LTE networks. We have extended the LTE module so that it supports controlling the relative CIO values between different cells. To simulate realistic users' mobility, we use Simulation of Urban Mobility (SUMO). SUMO imports accurately emulated environments from factual maps such as the Open Street Map (OSM). This imported environment considers the existing road structure, number of lanes, traffic light rules, buildings, $\cdots$, etc [21].

We implement the RL agents (specifically, the TD3 agents) using readily available Python implementations of stable-baselines 3 libraries [22]. The communication interface between the Python agents and the NS3 network simulator is realized using NS3gym [23]. Specifically, agents send their selected actions to the NS3 simulator and receive the calculated reward from the NS3 simulator using NS3gym. For implementation details and source code, see [24].

As for the simulation scenario, we choose an urban area of 900m $\times$ 1800m from the Fifth Settlement neighborhood in Egypt. The aforementioned area is covered by 6 eNBs. The exact positions of the eNBs were provided by one of the major 4G network operators in Egypt. The mobile UEs in this scenario are either vehicles or pedestrians. The pedestrians walk at a speed range between $0 - 3$ m/s. The vehicles'

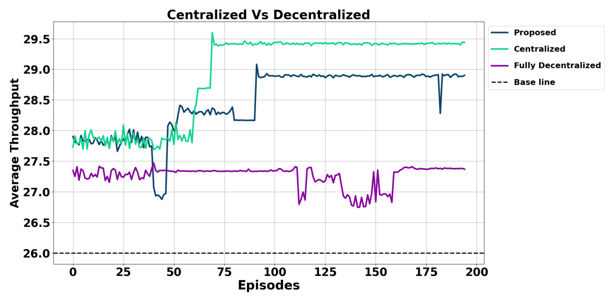

Fig. 3. Throughput performance of our proposed approach compared with benchmarks.

mobility characteristics, i.e., acceleration, deceleration, speed factor, and speed deviation, are taken to emulate the realistic behavior of the vehicles. We randomly distribute those UEs on the available streets and pedestrian lanes at the beginning of the simulation. Afterward, each UE has a random trip from a source to a destination street during the simulation time. UEs are assumed to have a full buffer traffic model, i.e., the users are always active. The unlimited demand of the users facilitates investigating network congestion with a lower number of UEs. The simulation parameters corresponding to our scenario are summarized in Table I.

B. Numerical Results

In this section, we present our numerical results concerning the performance of our proposed hierarchical multi-agent framework when applied to the aforementioned scenario. Our approach is compared against three benchmarks, namely: 1) Fully centralized agent that has access to the entire network KPIs and controls all relative CIO values of all eNBs [1], 2) Fully decentralized agents, each only deals with the KPIs and CIO values of only one cell with no communication across agents [6], and 3) Zero-CIO baseline, where the CIO values are all set to zero with no control from any of the agents. Furthermore, we highlight the importance of sharing CIO information between agents and the effect of the throughput penalty on the number of uncovered users within a cluster and the overall network.

1) Throughput Performance: First, we compare the throughput performance of our proposed approach compared to the aforementioned benchmarks. Fig. 3 shows that our proposed approach significantly outperforms the zero-CIO benchmark by 10.3% and the fully decentralized benchmark by 6%. Nevertheless, the fully centralized agent slightly outperforms our proposal by a mere 1.72%. This implies that there is a *minor reduction* in throughput due to having limited information for the hierarchical agents. The loss is expected as the cluster-level agents solve local optimization problems as opposed to the joint optimization problem that the centralized agent is solving. Nevertheless, the advantage of such an approach lies in its scalability compared to the centralized agent which becomes increasingly challenging to implement with increasing the number of eNBs.

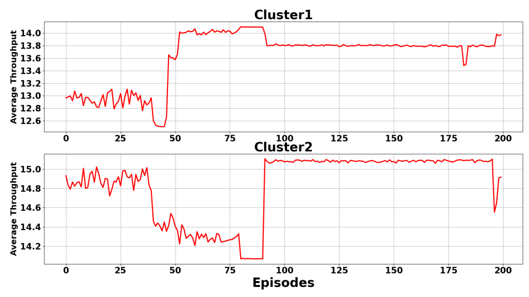

Fig. 4. Throughput per cluster

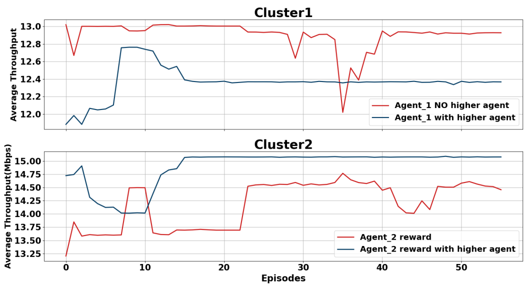

Fig. 5. Higher Agent Effect (Clusters)

Fig. 4 shows the individual throughput of each cluster during the learning phase. Interestingly, Fig. 4 shows that *both clusters' throughput* has increased. Specifically, the throughput enhancement cannot be derived from one successful agent, i.e., all agents work in unison to maximize the total throughput of the entire network.

2) Effect of the Presence of Network-Level Agent: We investigate the effects of including a network-level agent as opposed to using only cluster-level agents. We fix the number of training episodes to 50 and observe the agents' convergence behavior. Fig. 5 and Fig. 6 show that the inclusion of a network-level agent results in agents' converging in 15 episodes, while the throughput corresponding to the decentralized agents fluctuates even at the 50 episode training span. This maybe due to the fact that the network-level agent is compensating for the greedy actions of the cluster-level agents to reach a stable learning. On the other hand, compared to the fully-centralized agent in Fig. 6, the proposed hierarchical approach exhibits faster convergence as well. This is due to the fact that the state and action spaces of the agents in our approach are much smaller, which facilitates faster convergence.

3) Effect of Throughput Penalty: Next, we investigate the effect of the penalty factor λ on the throughput of the network. Fig. 7 shows that as λ increases, the overall throughput of the network decreases. This is because, with higher values for λ, the system tends to favor maximizing coverage at the cost of increasing user throughput to avoid incurring extreme losses due to the presence of uncovered users. This indicates an interesting tradeoff between throughput and coverage. The tradeoff can be controlled using the hyper-parameter λ. The network operator can set λ according to the current network

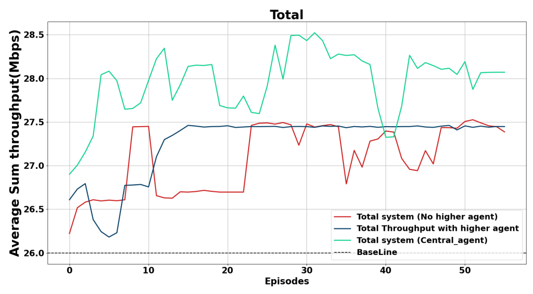

Fig. 6. Higher Agent Effect (Total)

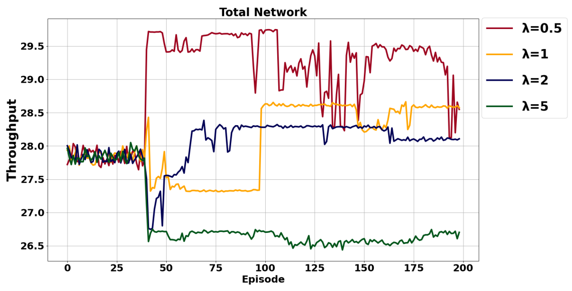

Fig. 7. Total Throughput different λ

needs.

VI. Conclusion

The primary motive for this study was to introduce a scalable approach for cellular network management using RL framework. We aim to enhance the throughput and coverage of the cellular network by optimizing handover parameters (a.k.a., CIO values) in a scalable fashion with minimal communication overhead. We proposed a hierarchical multi-agent approach, where cluster-level agents control intra-cluster CIOs, and network-level agent controls inter-cluster CIOs. We deduct that communicating CIO values is a sufficient communication overhead between agents. Our results show the following: 1) Our proposed approach outperforms fully decentralized agents with no network-level agent and no agents' communication, and 2) Our scheme is extremely scalable with respect to a centralized agent that observes all KPIs of the network and jointly controls all CIOs. This can be achieved with a negligible throughput reduction with respect to the fully centralized agent. Moreover, this approach enables customization for different rewards for each agent allowing different optimization goals on the cluster level for different operation scenarios.

References

[1] G. Alsuhli, K. Banawan, K. Attiah, A. Elezabi, K. G. Seddik, A. Gaber, M. Zaki, and Y. Gadallah, "Mobility load management in cellular networks: A deep reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1581–1598, 2021.

[2] G. Alsuhli, H. A. Ismail, K. Alansary, M. Rumman, M. Mohamed, and K. G. Seddik, "Deep reinforcement learning-based cio and energy control for lte mobility load balancing," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–6.

[3] G. Alsuhli, K. Banawan, K. Seddik, and A. Elezabi, "Optimized power and cell individual offset for cellular load balancing via reinforcement learning," in *IEEE WCNC*, 2021, pp. 1–7.

[4] M. Aboelwafa, G. Alsuhli, K. Banawan, and K. G. Seddik, "Self-optimization of cellular networks using deep reinforcement learning with hybrid action space," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 223–229.

[5] B. Salama, A. Elgharably, M. Aboelwafa, G. Alsuhli, K. Banawan, and K. G. Seddik, "Self-optimized agent for load balancing and energy efficiency: A reinforcement learning framework with hybrid action space," *Accepted for publication at IEEE Open Journal of the Communications Society*, 2024.

[6] H.-H. Chang, H. Chen, J. Zhang, and L. Liu, "Decentralized deep reinforcement learning meets mobility load balancing," *IEEE/ACM Transactions on Networking*, vol. 31, no. 2, pp. 473–484, 2023.

[7] K. Attiah, K. Banawan, A. Gaber, A. Elezabi, K. Seddik, Y. Gadallah, and K. Abdullah, "Load balancing in cellular networks: A reinforcement learning approach," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–6.

[8] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6255–6267, 2020.

[9] Y. Hao, F. Li, C. Zhao, and S. Yang, "Delay-oriented scheduling in 5g downlink wireless networks based on reinforcement learning with partial observations," *IEEE/ACM Transactions on Networking*, 2022.

[10] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[11] D. Huh and P. Mohapatra, "Multi-agent reinforcement learning: A comprehensive survey," *arXiv preprint arXiv:2312.10256*, 2023.

[12] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent reinforcement learning with communication," *arXiv preprint arXiv:2203.08975*, 2022.

[13] Z. Zhou, G. Liu, and Y. Tang, "Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges," *arXiv preprint arXiv:2305.10091*, 2023.

[14] ETSI, "5G NR, radio resource control (RRC), protocol specification, 3GPP TS 38.331 version 15.3.0 release 15," 2018.

[15] B. A. Adoum, K. Zoukalne, M. S. Idriss, A. M. Ali, A. Moungache, and M. Y. Khayal, "A comprehensive survey of candidate waveforms for 5g, beyond 5g and 6g wireless communication systems," *Open Journal of Applied Sciences*, vol. 13, no. 1, pp. 136–161, 2023.

[16] 3GPP, *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 14.2.0 Release 14)*, 2017.

[17] *LTE Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (3GPP TS 36.331 version 15.3.0 Release 15)*. 3GPP, 2018.

[18] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 15.2.0 Release 15)*. 3GPP, 2018.

[19] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[20] G. Alsuhli, K. Banawan, K. Seddik, and A. Elezabi, "Optimized power and cell individual offset for cellular load balancing via reinforcement learning."

[21] D. Krajzewicz and C. Rossel, "Simulation of urban mobility (sumo)," *Centre for Applied Informatics (ZAIK) and the Institute of Transport Research at the German Aerospace Centre*, 2007.

[22] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[23] P. Gawłowicz and A. Zubow, "ns3-gym: Extending openai gym for networking research," *arXiv preprint arXiv:1810.03943*, 2018.

[24] A. Elgharably, M. Aboelwafa, K. Banawan, and K. Seddik, "Paper source code," https://github.com/AamenElgharably/Hierarchical-Multi-Agent....