Decentralized Deep Reinforcement Learning Meets Mobility Load Balancing

Hao-Hsuan Chang, Hao Chen, *Member, IEEE*, Jianzhong Zhang, *Fellow, IEEE*, and Lingjia Liu, *Senior Member, IEEE*

***Abstract*— Mobility load balancing (MLB) aims to solve the problem of uneven resource utilization in cellular networks. Since network dynamics are usually complicated and non-stationary, conventional model-based MLB methods fail to cover all scenarios of cellular networks. On the other hand, deep reinforcement learning (DRL) can provide a flexible framework to learn to distribute cell load evenly without explicit modeling of the underlying network dynamics. In this paper, we introduce a novel decentralized DRL-based MLB method where each cell has a DRL agent to learn its handover parameters and antenna tilt angle. As the number of cells increases, the decentralized framework is more computationally efficient than its centralized counterpart by dividing the action space. Furthermore, our designed decentralized DRL architecture only requires readily known information defined in existing cellular standards, and it can achieve a more balanced cell load distribution than the centralized DRL one by using individual reward functions. To provide realistic performance evaluation, a network simulator is introduced strictly following the Third Generation Partnership Project (3GPP) specifications. Furthermore, field data is used to construct the underlying cellular environment. Extensive evaluations have been conducted to demonstrate the fact that the introduced decentralized DRL-based MLB method can achieve a more balanced cell load distribution and a better performance of edge users than the state-of-the-art MLB methods.**

***Index Terms*— Mobility load balancing, deep reinforcement learning, self-organizing network, handover.**

I. Introduction

IT IS predicted that global IP traffic increases 3-fold from 2017 to 2022, reaching 396 exabytes per month by 2022 [1]. The fifth-generation (5G) mobile broadband networks is expected to meet the exponential increase of mobile data traffic due to the widespread use of mobile devices and new wireless services. In order to support the high data demand, small cells are densely deployed. This dense deployment reduces the distance between the mobile users and the base station (BS), which significantly increases the capacity and throughput of cellular networks. However, small cell networks are much more vulnerable to the mobility of users than large cell networks. Since the coverage areas of small cells usually overlap with each other, the dynamics of user clustering locations may cause an unbalanced cell load distribution within small cell networks. Such unbalanced cell load distribution over the network results in uneven resource utilization, i.e., many users compete for the resources in an overloaded cell while neighboring cells remain underloaded. Therefore, mobility load balancing (MLB), which aims to distribute cell load evenly by transferring part of the traffic from congested cells, is particularly important to small cell networks.

In many existing cellular networks, system parameters are manually adjusted to maintain a high level of system performance. However, manual parameter adjustments are becoming difficult due to the increasing complexity of cellular networks. To reduce the operational complexity of cellular networks, the concept of the self-organizing network (SON) has been introduced into the Third Generation Partnership Project (3GPP) as one of the key enabling technologies [2]. SON aims to achieve automation of planning, configuration, and optimization within mobile radio access networks. One main support function of SON is MLB [3], which is done by the self-optimization of mobility parameters or handover actions.

Conventional MLB strategies are model-based using pre-determined/simplified network models [4]. This is because the actual cell loads in real-world deployments depend on many factors such as coverage area, user distribution, and user traffics making it extremely challenging or even impossible to accurately analyze and determine a good network behavior model for these model-based MLB methods [5]. Accordingly, in modern cellular networks such as the 4G LTE-Advanced networks, fixed load balancing rules are adopted to conduct MLB strategies [6]. To be specific, based on the pre-determined load balancing rules, MLB strategies will tune the handover parameters, such as cell individual offset (CIO) [7], [8] and hysteresis margin [9] to conduct MLB and handover. However, these pre-determined rules can only cover limited network scenarios based on user mobility, traffic pattern, and network deployment scenarios. This is also the reason why the field trial results show that deploying model-based MLB strategies in real-world scenarios will lead to an 85% to 97% increase of connection dropping and handover failure ratios. Meanwhile, the high dynamics of user behaviors in practical networks can trigger frequent handovers

Manuscript received 7 March 2021; revised 6 December 2021 and 1 March 2022; accepted 28 April 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Ghaderi. Date of publication 31 August 2022; date of current version 18 April 2023. The work of Hao-Hsuan Chang and Lingjia Liu was supported in part by the U.S. National Science Foundation (NSF) under Grant CCF-1937487. *(Corresponding author: Lingjia Liu.)*

Hao-Hsuan Chang , Hao Chen, and Jianzhong Zhang are with the Standards and Mobility Innovation Laboratory, Samsung Research America, Plano, TX 75023 USA.

Lingjia Liu is with the Electrical and Computer Engineering Department, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: ljliu@ieee.org).

Digital Object Identifier 10.1109/TNET.2022.3176528

for model-based MLB strategies leading to extremely fluctuated cell load distribution [10]. As 5G and Beyond 5G networks become more and more dynamic and heterogeneous, it is clear that these model-based strategies will fall short even more significantly.

To cope with the increasingly complicated network operations encountered in 5G and Beyond, artificial intelligence, and machine learning techniques have been introduced [11]. In this paper, we utilize deep reinforcement learning (DRL) to design the MLB management policy, which has been successfully applied to many applications in wireless communications [12]–[16]. Reinforcement learning (RL) can adapt to unknown environments by learning from interactions with the environment, and DRL, which is a combination of deep learning and RL, aims to solve the large state space problem in RL by utilizing the generalization capability of deep neural networks. We will show that DRL is an appealing approach for MLB management since it is capable of learning to solve complex problems without explicit models to overcome the drawbacks of model-based MLB approaches and to enable a richer online feature presentation of the dynamic network behavior.

Existing DRL-based MLB methods use centralized learning algorithms [17], [18] to maintain a balanced load among cells. However, there are some significant issues for applying the centralized DRL-based MLB method when the number of cells in the network increases. First, the centralized DRL agent requires the network-wide information of all cells causing significant signaling overheads. Second, the centralized DRL method is computationally inefficient to the MLB problem because the underlying action space increases significantly. Third, it is inefficient to jointly learn the MLB control parameters of all cells because a single reward function is not informative to all cells in the network. Therefore, [17] first divides the cells into different groups, and then balances the intra-group load distribution using the centralized DRL-based algorithm in each group. In this paper, we develop a novel decentralized DRL-based MLB method that can be readily scaled to a large size network In our designed decentralized DRL-based MLB architecture, each cell has a DRL agent to learn its MLB control parameters from a more informative local reward without requiring any additional signaling compared to existing cellular standards.

To provide fair and relevant evaluations of different MLB methods, we develop a new network simulator to provide a near real-world cellular environment. This is achieved through the following: First, the simulator strictly follows 3GPP specifications to set up the mobility management in practical cellular networks, including the handover process, resource allocation, and user traffic models. Second, the underlying environment of the network simulator is constructed by measured channel gains obtained from field data. Compared to the model-based testing environments in existing MLB evaluations, the new network simulator provides an experimental platform that is very close to real cellular environments.

The main contributions of the paper are as follows:

1) To the best of our knowledge, our work is the first decentralized DRL architecture introduced for the MLB task in LTE/LTE-Advanced networks. As the number of cells increases, it becomes extremely challenging and computationally inefficient for the centralized DRL agent to perform the MLB function of all cells because the underlying action space also increases significantly. Therefore, the decentralized DRL-based MLB architecture utilizes multiple DRL agents to perform the MLB function of each cell, which largely decreases the computational complexity of the underlying DRL training.
2) Furthermore, we carefully design the decentralized DRL-based MLB architecture including the individual reward function as well as the state of each DRL agent so that the introduced decentralized DRL-based MLB only utilizes the exchanged control information from neighboring cells that are already defined in existing cellular standards such as 3GPP LTE-Advanced and 5G NR to ensure the scalability and the significance of the strategy. Note that in the load balancing problem a joint reward function is largely determined by the most crowded cell failing to provide enough information for all the cells. On the other hand, the state of each DRL agent needs to carefully designed so that the load distribution in every cell is balanced and it can be readily scaled to large-scale cellular networks.
3) Besides the methodology design, we also build a new network simulator to evaluate the performance of mobility management and load balancing strictly following 3GPP specifications based on real-world field measurements. Instead of doing the pure simulation-based performance characterization as shown in most of the existing works, we evaluate different MLB algorithms in a near real-world cellular environment using the introduced network simulator to enhance the practical relevance of the work.
4) Finally, we conduct extensive performance evaluations to demonstrate that the merits of the introduced decentralized DRL-based MLB method over the state-of-the-art MLB methods in terms of balanced cell load distribution and edge users' performance.

The remainder of this paper is organized as follows. Section II introduces the network model. Section III presents both the centralized and decentralized DRL solutions to address the MLB problem. Meanwhile, the issues of the centralized DRL-based MLB methods are presented to show the advantage of our introduced decentralized DRL architecture for the MLB task. The network simulator is introduced in Section IV, where field measurements are used to construct the underlying cellular environment. The simulator strictly follows 3GPP specifications making it 5G compliant. Section V shows the performance evaluations of different MLB algorithms. Lastly, we conclude the paper in Section VI.

II. Network Model

In this paper, we investigate the load balancing method for LTE downlink network. We consider small cell networks

where the number of cells is M and the number of mobile users is N. We let the set of cells be $\mathbb{M} = \{1, \ldots, M\}$, the set of users be $\mathbb{N} = \{1, \ldots, N\}$, and the set of neighboring cells of cell m be $\mathbb{U}_m$. The connections between users and cells are determined by both the locations of users and the coverage areas of cells. Each user is connected to one cell in $\mathbb{M}$, and we denote the set of users connected to cell m at time t as Φ_m^t. It is important to note that the MLB function in cellular networks aims to balance the load of cells in a relatively large time-scale since the exchange of the load information among neighboring cells is through the Radio Resource Control (RRC) layer on the order of seconds (1sec – 10sec) as defined in 3GPP TS 136.423. Even though the state-of-the-art cellular networks such as the 3GPP LTE-Advanced and 5G NR define many advanced transmission modes such as multi-user MIMO [19] and coordinated multi-point (CoMP) transmission [20]–[22], these transmission modes are configured by the network depending on the dynamic situation of the UE on a subframe (millisecond level) basis. Therefore, MLB and handover strategies do not consider actual transmission modes in modern cellular networks since the decisions on MLB and transmission modes are on completely different time scales.

Each cell has a scheduler that conducts resource allocation based on its scheduling algorithm. The granularity of a cell's resource allocation is called resource block group (RBG), a bundle of multiple resource blocks (RB). The available RBGs of each cell are finite tying into the system bandwidth. It is possible that some cells in the network are overloaded because many users connect to them, while some cells with few connected users are underloaded. Since each cell can only offer finite resources to its connected users, the uneven cell load distribution leads to unbalanced resource utilization in the network. Therefore, MLB is an important technique to maintain a more balanced resource utilization by transferring the load from overloaded cells to the neighboring cells with free resources.

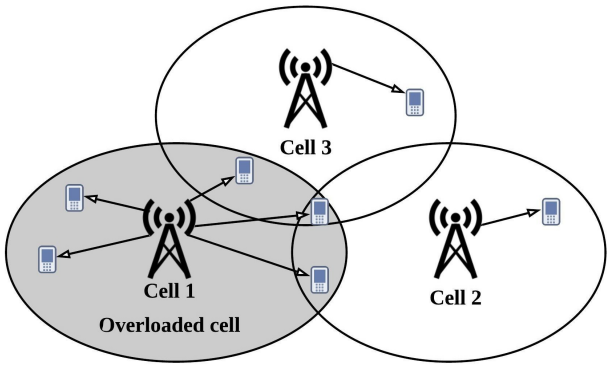
Handover procedure is a key function of LTE to transfer some users connected to a cell to neighboring cells in order to receive better signal quality or more available RBGs. By changing coverage areas or the handover criterion, the connections between users and cells can be adjusted for achieving a more even cell load distribution as shown in Fig. 1. Since users cluster at different locations in different time periods, it is important for the MLB algorithm to adjust the cell load distribution dynamically in order to adapt to the dynamic user distribution. In this paper, we consider how to balance the cell load distribution through adjusting handover parameters and antenna tilt angles of cells.

A. Cell Load

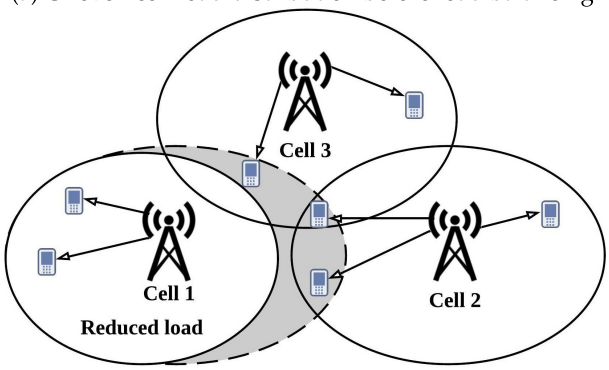
In each resource allocation period, each cell allocates RBGs to its connected users based on the scheduling algorithm. The scheduling algorithm used in this paper is detailed in Section IV-D. According to the scheduling result, the load of the cell is defined as the ratio of allocated RBGs versus the total number of RBGs. To be specific, the load of cell m at time t is

$$\rho_m^t = \frac{\sum_{n \in \Phi_m^t} W_{m,n}^t}{W_m}, \tag{1}$$

where Φ_m^t is the set of connected users of cell m at time t, $W_{m,n}^t$ is the number of RBGs that allocates to user n by cell m at time t, and W_m is the total number of RBGs of cell m.



(a) Uneven cell load distribution before load balancing.



(b) Even cell load distribution after load balancing.

Fig. 1. The scenario of load balancing.

B. Handover

We consider the handover procedure in LTE/LTE-Advanced networks [3]. To be specific, a cell first sends its connected user a measurement configuration message specifying the items that the user has to measure and the event that the user has to report, which is defined in the RRC protocol in 3GPP 136.331 [23]. Once the cell receives the event reported by the user, it makes the handover decision. Note that the RRC message for handover is only exchanged between the user and the cell in a relatively large time period (in the order of seconds) to reduce signaling overheads. We assume that the handover of user n from the serving cell m to the neighbor cell m' occurs at time t when user n reports the A3 event, which is defined as

$$F_{m',n}^t - F_{m,n}^t \geq O_{m,m'} + H_m, \tag{2}$$

where $F_{m,n}^t$ and $F_{m',n}^t$ represent user n's reference signal received power (RSRP) at time t from the serving cell m and the neighboring cell m', respectively, H_m is the hysteresis margin to prevent frequent handovers, and $O_{m,m'}$ is the CIO

between cell m and cell m'. Note that a user's RSRP from a cell is affected by the antenna tilt angle of the cell, where the antenna tilt angle is the angle between the main beam of the antenna and the horizontal plane. To be specific, the antenna tilt angle of a cell determines the direction of the antenna beam. If a user's location is on the line of sight direction of the antenna beam, then this user receives higher signal power from that cell and measures a larger RSRP value. In this paper, the MLB algorithm controls CIO values and antenna tilt angles to trigger user handovers to balance the cell load distribution.

III. DRL for Mobility Load Balancing

A. Background on RL

RL is one type of machine learning method that provides a flexible solution for many real-world problems because it does not need to model complex systems or to label data for training. In the RL setting, an agent interacts with an environment in discrete steps and learns how to select actions to maximize the cumulative reward in a stochastic environment. The dynamics of the environment is usually modeled as a Markov decision process (MDP), which is characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state transition probability, R is the reward function, and $\gamma \in [0, 1]$ is a discount factor for calculating the cumulative reward. To be specific, at step k, the state is $s[k] \in \mathcal{S}$, and the agent selects an action $a[k] \in \mathcal{A}$ based on a policy: $\pi(a, s) = \Pr(a[k] = a \mid s[k] = s)$. After taking the action $a[k]$, the agent receives a reward $r[k] = R(s[k+1], s[k], a[k])$, and then the system shifts to the next state $s[k+1]$ based on the state transition probability: $\mathcal{P}(s', s, a) = \Pr(s[k+1] = s' \mid s[k] = s, a[k] = a)$. The goal of the RL agent is to find the optimal policy π that maximizes the expected cumulative reward, $\mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^{k-1} r[k]\right]$.

B. Problem Formulation for Centralized and Decentralized Deep Reinforcement Learning

In this section, we formulate the MLB problem under the DRL framework, which can be done in a centralized or decentralized fashion. To achieve a balanced cell load distribution, we consider that the DRL agent controls the CIO values between cells and the antenna tilt angle of each cell. The CIO value between any two cells represents the decision thresholds of handovers, which is defined in (2). The antenna tilt angle is defined as the angle between the main beam of the antenna and the horizontal plane. Note that different antenna tilt angles result in different received power strengths over the area [24], [25], so the inter-cell interference experienced by an edge user can be mitigated through adjusting the tilt angles of cells. To be specific, if a cell has many edge users suffering from strong inter-cell interference, then it has to allocate many RBGs to these edge users according to the adopted proportional fair scheduling algorithm, which makes the load of the cell increase significantly. Therefore, the DRL-based MLB method can be aware of the strong inter-cell interference within a cell by detecting the high cell load, and then implicitly mitigates the co-channel inter-cell interference by changing the tilt angles to balance the cell load distribution.

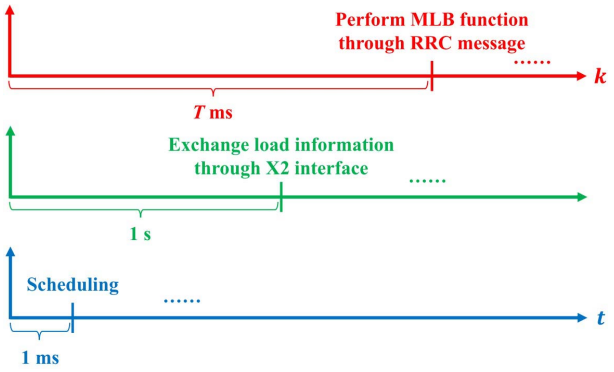


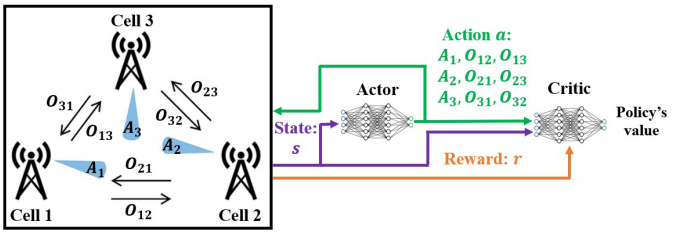
Fig. 2. The time structure of resource scheduling, exchanging load information, and execution of MLB function.

Furthermore, we design multiple reward functions to represent the load balancing performance in the network, where the reward functions scale with better load balancing performance. The goal of the DRL agent is to learn how to adjust CIO values and antenna tilt angles to maximize the cumulative reward.

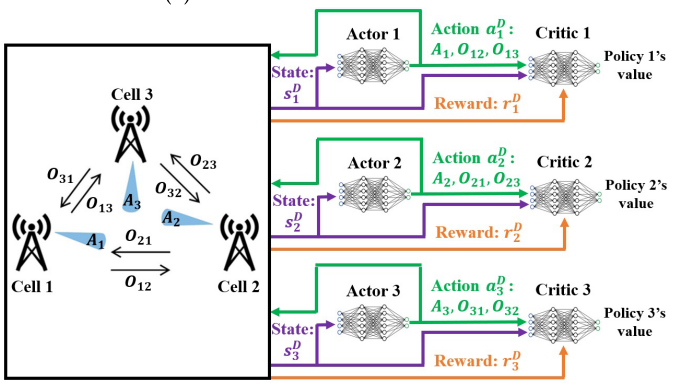
In modern cellular networks such as the 3GPP LTE-Advanced and 5G NR, the MLB function is supported by the load reporting mechanism activated between neighboring cells over the X2 interface defined in 3GPP TS 136.423 to exchange their load information [26]. Specifically, a cell sends a resource status request message to the target cell, then the target cell responds with a resource status response message and reports its load information by sending a resource status update message. The time period of exchanging the load information is in the order of seconds as defined in 3GPP TS 136.423. Meanwhile, the RRC message for handover between the UE and the cell is also exchanged in the order of seconds as defined in 3GPP 136.331 [23]. Compared to the dynamic transmission mode adaptation in modern cellular networks that usually works on a millisecond basis, the MLB function aims to balance the load of cells in a relatively large time-scale. According to the time-scales of different network operations, we assume that the resource scheduling is done every 1 millisecond, the load information of different cells is exchanged every 1 sec ($= 1000$ milliseconds), and the DRL agent performs its MLB function every T milliseconds. The time structure is shown in Fig. 2.

The DRL-based MLB framework can be centralized or decentralized, as shown in Fig. 3. The centralized DRL-based MLB method only has one DRL agent to determine CIO values and antenna tilt angles for all cells, while the decentralized DRL-based MLB method has a DRL agent for each cell to determine each cell's CIO value and antenna tilt angle. Since the DRL agent in the centralized DRL-based MLB method has to perform the MLB function for all cells in the network, it requires the load information of all cells. On the other hand, in our introduced decentralized DRL-based MLB method, each cell has a DRL agent to perform the MLB function locally, so each DRL agent only requires the local information from neighboring cells.

1) State: In this paper, we use the cell load and the user throughput as the input state to the DRL agent. To be specific, at time kT, the state of the centralized DRL-based MLB



(a) Centralized DRL framework.



(b) Decentralized DRL framework.

Fig. 3. DDPG algorithm schematic in load balancing.

method is defined as

$$s^{\mathrm{C}}[k] = \left\{ \left(\overline{\rho}_m[k-1], \overline{R}_m[k-1] \right) \mid \forall m \in \mathbb{M} \right\}, \tag{3}$$

where $\overline{\rho}_m[k-1]$ and $\overline{R}_m[k-1]$ are the average load and the average user throughput of cell m over time $(k-1)T+1$ to kT, respectively. On the other hand, for the decentralized DRL-based MLB method, the state of cell m's DRL agent at time kT is defined as

$$s_m^{\mathrm{D}}[k] = \left\{ \left(\overline{\rho}_{m'}[k-1], \overline{\rho}_m[k-1], \overline{R}_m[k-1] \right) \mid \forall m' \in \mathbb{U}_m \right\}, \tag{4}$$

The average load $\overline{\rho}_m[k-1]$ and the average user throughput $\overline{R}_m[k-1]$ are calculated by

$$\overline{\rho}_m[k-1] = \frac{\sum_{t=(k-1)T+1}^{kT} \rho_m^t}{T} \tag{5}$$

and

$$\overline{R}_m[k-1] = \frac{\sum_{t=(k-1)T+1}^{kT} \left(\sum_{n \in \Phi_m^t} R_n^t / |\Phi_m^t| \right)}{T}, \tag{6}$$

respectively, where ρ_m^t is the load of cell m at time t defined in (1), Φ_m^t is the set of connected users of cell m at time t, and R_n^t is the instantaneous throughput of user n at time t.

2) Action: The action of the centralized DRL-based MLB method can be written as

$$a^{\mathrm{C}}[k] = \{ (O_{m,m'}[k], A_m[k]) \mid \forall m \in \mathbb{M}, \forall m' \in \mathbb{U}_m \}, \tag{7}$$

where $O_{m,m'}[k]$ is the CIO between cell m and its neighboring cell m' from time $kT+1$ to $(k+1)T$, and $A_m[k]$ is the tilt angle of cell m from time $kT+1$ to $(k+1)T$. On the other hand, the action of cell m's DRL agent in the decentralized DRL-based MLB method is written as

$$a_m^{\mathrm{D}}[k] = \{ (O_{m,m'}[k], A_m[k]) \mid \forall m' \in \mathbb{U}_m \}, \tag{8}$$

In the experiment, the CIO value is continuous in the range of $[-O_{\max}, O_{\max}]$ and the tilt angle is discrete in the range of $\{0^\circ, 1^\circ, \ldots, 12^\circ\}$. To enable discrete tilt angle selection using the continuous action space of the DDPG algorithm, the continuous action space of antenna tilt angle is divided into discrete intervals to determine the tilt angle.

3) Reward: The reward function should scale with better load balancing performance, so the DRL agent can learn to achieve a more balanced cell load distribution by maximizing the cumulative reward. For the centralized DRL-based MLB method, we use two reward functions as examples. The first reward function is the reciprocal of the maximum cell load, which can be written as

$$r^{\mathrm{C1}}[k] = \frac{1}{\max_{m \in \mathbb{M}} \overline{\rho}_m[k+1]}. \tag{9}$$

The second reward function is the negative of the standard deviation of all cells' load, which is written as

$$r^{\mathrm{C2}}[k] = -\sigma_{\overline{\rho}}[k+1], \tag{10}$$

where $\sigma_{\overline{\rho}}[k+1]$ stands for the standard deviation of $\{\overline{\rho}_m[k+1] \mid m \in \mathbb{M}\}$. For the decentralized DRL-based MLB method, the reward function of cell m is the negative of the maximum load difference between it and other neighboring cells, which is written as

$$r_m^{\mathrm{D}}[k] = -\max_{m' \in \mathbb{U}_m} |\overline{\rho}_{m'}[k+1] - \overline{\rho}_m[k+1]|. \tag{11}$$

C. Issues of Centralized DRL-Based MLB Method

Existing DRL-based MLB methods use centralized learning algorithms [17], [18]. However, there are some significant issues for applying the centralized DRL-based MLB method to a large-scale cellular network. First, the centralized DRL-based MLB method requires additional signaling overheads for obtaining network-wide information of all cells to construct the state of the underlying DRL agent. Therefore, the centralized DRL-based MLB method is difficult to be applied to a large-scale cellular network because obtaining the information of all cells causes significant signaling overheads in a large-scale cellular network. On the other hand, our introduced decentralized DRL-based MLB method constructs the state based on the exchange information from neighboring cells that are already defined in existing cellular standards. This is clear evidence showing our approach can be readily scaled to large size cellular networks. Second, the centralized DRL training becomes computationally inefficient for the MLB problem in a large-scale cellular network because the action space scales significantly with the number of cells. This is because the DRL agent has to explore the large action space to learn how to achieve a balanced cell load distribution. Third, it is inefficient to jointly learn the MLB control parameters of all cells using the centralized DRL framework because a single reward function mainly reflects the cell load distribution of the most crowded area in the network. For example, [17] uses (9)

as the reward function, which represents the load balancing performance of the most overloaded cell. Accordingly, this reward function is only informative to the most overloaded cell, but fails to provide insights and information to other cells that are not operating at the maximum load. In this case, it is challenging to efficiently learn strategies to achieve balanced cell loads. In a large-scale network with a huge number of cells, it is difficult or even impossible to use a single reward that can accurately reflect the load balancing performance in different parts of the network. Therefore, our decentralized DRL-based MLB method utilizes multiple rewards in (11) to represent the load balancing performance in different local parts of the network. In this way, the DRL agent of each cell can learn from a more informative reward that represents the local cell load distribution, which largely improves the learning efficiency of the DRL agent.

D. Training Algorithm for DRL-Based MLB Method

A model-free RL algorithm can learn a policy without requiring the knowledge of the state transition probability and the reward function, which is useful for the MLB problem since the underlying wireless system is complicated and difficult to model. Meanwhile, data collection is usually expensive in wireless communications, so an off-policy DRL method is needed for sample efficiency. Therefore, we adopt a model-free and off-policy training algorithm that can be applied to continuous action space, which is called Deep Deterministic Policy Gradient (DDPG) [27]. DDPG has an actor-critic architecture [28], which comprises an actor network and a critic network. The actor network with parameters θ_A represents a deterministic policy: $\mu(s|\theta_A)$, which maps a given state s to a specific action. The critic network $Q\left(s,a|\theta_C\right)$ with parameters θ^C represents the estimated Q-function of the actor's policy $\mu(s|\theta_A)$. During training, the actor network improves its policy $\mu(s|\theta_A)$ with the estimated Q-function $Q\left(s,a|\theta_C\right)$ based on the policy gradient method, where the policy gradient with respect to θ_A is given as

$$\nabla_a Q\left(s,a|\theta_C\right)|_{s=s[k],a=\mu(s[k]|\theta_A)} \nabla_{\theta_A}\mu(s|\theta_A)|_{s=s[k]} \tag{12}$$

On the other hand, the critic network improves its estimated Q-function $Q\left(s,a|\theta_C\right)$ with the policy $\mu(s|\theta_A)$ based on the DQN training algorithm. To improve the training stability, the DDPG training algorithm adopts the target network idea in Deep Q-Network [29]. Specifically, DDPG creates a copy of the actor and critic networks, $\mu(s|\theta_A^-)$ and $Q\left(s,a|\theta_C^-\right)$, respectively. These two target networks are utilized for calculating the target Q-values as follows:

$$y[k] = r[k] + \gamma Q\left(s[k+1], \mu\left(s[k+1]|\theta_A^-\right)|\theta_C^-\right), \tag{13}$$

and the loss function for updating θ_C is written as

$$\left(y[k] - Q\left(s[k],a[k]|\theta_C\right)\right)^2. \tag{14}$$

The weights of target networks θ_A^- and θ_C^- are updated using soft updates: $\theta_A^- \leftarrow \tau\theta_A + (1-\tau)\theta_A^-, \theta_C^- \leftarrow \tau\theta_C + (1-\tau)\theta_C^-$, where $\tau \in (0,1]$ is a small positive number for a soft target network update. The critic network and the actor network take turns to improve the estimated Q-function $Q\left(s,a|\theta_C\right)$ and the policy $\mu(s|\theta_A)$ until convergence.

Since each DRL agent in our decentralized MLB method may have different input and output dimensions, it is difficult to train all DRL agents with various input and output dimensions in an offline manner. Therefore, we adopt online learning to train the underlying DRL agents. To be specific, each DRL agent takes actions based on the DDPG policies, and then utilizes the collected training data to train itself on the fly. Since DDPG policies are deterministic policies that directly output the actions, we add zero-mean Gaussian noise to the deterministic actions in order to better explore the environment. Note that the variance of the Gaussian noise is set to be a relatively large value at the beginning of the online training to explore the environment with a wide variety of actions. Then, the variance of the Gaussian noise will gradually decrease when the training starts to converge. This online training scheme has the advantage of updating the DRL-based MLB policies continuously to adapt to the dynamic wireless environment over time. The decentralized DDPG-based training algorithm for the MLB problem is provided in Algorithm 1.

Algorithm 1 Decentralized DDPG-Based Training Algorithm for the MLB Problem

Initialize the cellular network with M cells.
Each cell m initializes a DDPG-based DRL agent with actor $\mu(s|\theta_{A,m})$ and critic $\mu(s|\theta_{A,m})$.
Each cell m initializes the target networks $\mu(s|\theta_{A,m}^-)$ and $Q\left(s,a|\theta_{C,m}^-\right)$ with $\theta_{A,m}^- \leftarrow \theta_{A,m}, \theta_{C,m}^- \leftarrow \theta_{C,m}$.
Set the action noise $\mathcal{W}$ as uncorrelated and zero-mean Gaussian random variables with variance σ_W.
Each cell m receives $s_m^{\mathrm{D}}[1]$.
for $k = 1, 2, \ldots$ **do**
  Each cell m executes $a_m^{\mathrm{D}}[k] = \mu\left(s_m^{\mathrm{D}}[k]|\theta_{A,m}\right) + \mathcal{W}$.
  Each cell m receives $r_m^{\mathrm{D}}[k]$ and $s_m^{\mathrm{D}}[k+1]$.
  Each cell m stores $\left(s_m^{\mathrm{D}}[k], a_m^{\mathrm{D}}[k], r_m^{\mathrm{D}}[k], s_m^{\mathrm{D}}[k+1]\right)$ in its replay buffer B_m.
  Each cell m randomly samples from B_m to form a training batch.
  Each cell m trains $\theta_{C,m}$ by minimizing (12).
  Each cell m trains $\theta_{A,m}$ using (14).
  Each cell m updates their target networks:

$$\theta_{A,m}^- \leftarrow \tau\theta_{A,m} + (1-\tau)\theta_{A,m}^-$$
$$\theta_{C,m}^- \leftarrow \tau\theta_{C,m} + (1-\tau)\theta_{C,m}^-$$

  Decrease σ_W.
end for

It is important to note that our designed decentralized DRL-based MLB architecture can largely reduce the complexity of the underlying DDPG training. To be specific, the introduced decentralized learning architecture can largely decrease the state space and the action space of each DRL agent, and thus each DRL agent can search in a much smaller state-action space to reduce the computational complexity

of the underlying DDPG training. Furthermore, we carefully designed the individual reward function to improve the learning efficiency of each DRL agent. Note that in the load balancing problem a joint reward function is largely determined by the most crowded area in the network, but it fails to provide insights and information to other cells that are not operating at the maximum load. Therefore, the learning efficiency of the decentralized DRL framework is substantially improved by utilizing individual rewards in the MLB problem.

The computational complexity of DDPG can be characterized by the required computations for training the underlying neural network of DDPG, which has an actor network and a critic network. Let a neural network has L layers and the number of nodes $\left(d^1, \cdots, d^L\right)$, where d^l $(1 \leq l \leq L)$ is the number of nodes in the l^{th} layer. We train this neural network using the backpropagation algorithm with a batch size of b. Since the computational complexity of multiplying an $i \times j$ matrix by a $j \times k$ matrix is $O(ijk)$, we can know that the computational complexity of applying the backpropagation algorithm to this neural network is $O\left(b \times \left(\sum_{l=1}^{L-1} d^l d^{l+1}\right)\right)$. Accordingly, we can characterize the computational complexity for training the actor network and the critic network in DDPG if the neural network architecture is known. For the actor network, the node number in the input layer is the dimension of the state space, and the node number in the output layer is the dimension of the action space. For the critic network, the node number in the input layer is the dimension of the sum of state space and action space, and the node number in the output layer is 1. It is important to note that both the state space and the action space of the centralized DRL-based MLB method are much larger than that of our decentralized DRL-based MLB method. To be specific, the dimension of the state space in the centralized and the decentralized DRL-based MLB method are $2M$ and $2+|\mathbb{U}_m|$, respectively. Meanwhile, the dimension of the action space in the centralized and the decentralized DRL-based MLB method are $M+\sum_{m=1}^{M}|\mathbb{U}_m|$ and $1+|\mathbb{U}_m|$, respectively. Therefore, we can show that the computational complexity of the centralized DRL-based MLB method is much larger than that of the decentralized DRL-based MLB method.

IV. Network Simulator Based on 3GPP Specifications and Field Measurements

The relationship between the cell load distribution and the control parameters of MLB algorithms is affected by many factors in cellular networks. To evaluate different MLB algorithms, it is important to build a network simulator that is close to real cellular environments. The network simulator strictly follows 3GPP specifications with all the important mobility management procedures, including the handover process, resource allocation, and user traffic models. Furthermore, the underlying cellular environment of the network simulator is constructed by channel gains obtained from field measurements. Since most mobile devices are carried by people, we adopted a human mobility model to generate user mobility traces. At any given user location, the received signal-to-interference-plus-noise ratio (SINR) is calculated by using the channel gains from field measurements. Finally, we combine the network simulator and user mobility traces to generate a near real-world cellular environment. The flowchart of the network simulator is shown in Fig. 4.

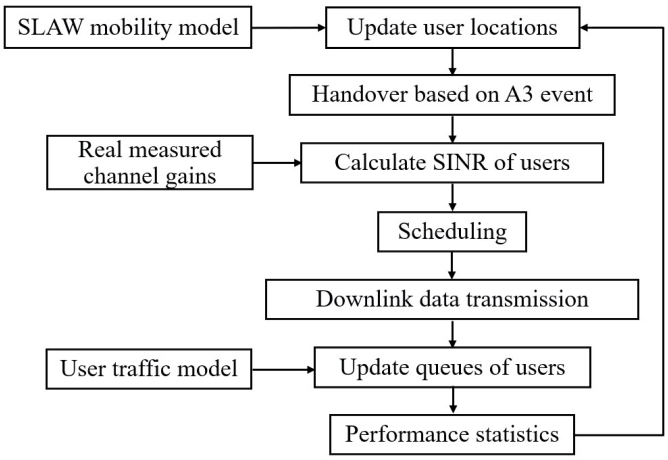


Fig. 4. Flowchart of the network simulator.

A. User Mobility Model

Since wireless devices are often carried by people, we generate the user trajectories using a human mobility model called Self-similar Least-Action Walk (SLAW) mobility model [30]. The SLAW mobility model captures several important human mobility patterns. First, the destinations of people are dispersed in a self-similar manner because people cluster around similar places like offices, restaurants, or homes. Second, people have the least-action planning because they are more likely to first go to the destination that is closest to their current locations when visiting multiple destinations in succession. The SLAW mobility model creates realistic human mobility trajectories, which is important for the performance evaluation of mobile networks.

B. Obtain Channel Gains From Field Data

We assume that all cells are operating on the same bandwidth for data transmission, thereby the transmit signals from different cells cause interference to one another. Given that user n is connected to cell m, the received SINR of user n at time t is calculated by

$$\text{SINR}_{m,n}^{t} = \frac{P_m H_{m,n}^{t}}{N_0 + \sum_{j \in \mathbb{M}, m' \neq m} P_{m'} H_{m',n}^{t}}, \tag{15}$$

where $\mathbb{M}$ is the set of cells, P_m and $P_{m'}$ are the transmit powers of cell m and cell m', respectively, $H_{m,n}^{t}$ is the channel gains between cell m and user n at time t, $H_{m',n}^{t}$ is the channel gains between cell m' and user n at time t, and N_0 is the noise power. Note that $P_m H_{m,n}^{t}$ is the desired signal power from cell m at time t, while $P_{m'} H_{m',n}^{t}$ is the interference power caused by cell m' at time t.

We obtained the channel gains from field measurements to reflect the real-world cellular environment. To be specific, we measure the channel gains from each cell for tilt angles

TABLE I
THE MAPPING FROM THE SINR TO THE MCS

SINR (dB) (dB)	CQI	Modulation order (total bits/symbol)	Code rate (useful bits/total bits)
< -6.94	0	Raio Link Failure	
≥ -6.94	1	2	78/1024
≥ -5.15	2	2	120/1024
≥ -3.18	3	2	193/1024
≥ -1.25	4	2	308/1024
≥ 0.76	5	2	449/1024
≥ 2.70	6	2	602/1024
≥ 4.69	7	4	378/1024
≥ 6.53	8	4	490/1024
≥ 8.57	9	4	616/1024
≥ 10.37	10	6	466/1024
≥ 12.29	11	6	567/1024
≥ 14.17	12	6	666/1024
≥ 15.89	13	6	772/1024
≥ 17.81	14	6	873/1024
≥ 19.83	15	6	948/1024

$0°, 1°, \ldots, 12°$ at several specified locations. The channel gains are measured at multiple locations with a discrete grid structure that has a separation distance of 8 meter. The parameters of the grid structure and the tilt angle range are chosen jointly considering the accuracy of the measurement as well as the difficulty of conducting the field measurement. In our performance evaluation, interpolation is used to obtain channel gains between the specified locations.

C. LTE Downlink Transmission Scheme

We calculate the user throughput based on the underlying modulation and coding scheme (MCS) adopted in the 3GPP LTE/LTE-Advanced standard. To be specific, a user measures the SINR as the quality of the wireless link and sends the Channel Quality Indicator (CQI) to its connected cell. Since there is no explicit regulation of how a user calculates CQI in 3GPP specifications, we adopt a mapping table to map SINR to CQI in a similar way of [31]. After receiving the CQI from the user, the cell determines the MCS for data transmission based on the CQI table in 3GPP TS 136.213 [32]. The MCS defines the numbers of useful bits which can be carried by one Orthogonal Frequency Division Multiplexing (OFDM) symbol. There are two components in the MCS: the modulation order and the code rate. The modulation order is the number of bits carried by an OFDM symbol, and the code is the ratio of useful bits to total bits. Table I is the mapping table from the SINR value to the MCS. Note that if the received SINR of a user is under a threshold, the radio link of this user becomes disconnected, which is called Radio Link Failure (RLF). Given that user n is connected to cell m, we let $\mathrm{MO}^t_{m,n}$ and $\mathrm{CR}^t_{m,n}$ be the modulation order and the code rate of user n provided by cell m at time t, respectively. We can calculate the instantaneous throughput of user n provided by one RBG of cell m at time t as follows:

$$r^t_{m,n} = \frac{\mathrm{MO}^t_{m,n}\mathrm{CR}^t_{m,n} N_{\mathrm{RB}} N_{\mathrm{symbol}}}{T_{\mathrm{RBG}}}, \tag{16}$$

where N_{RB} is the number of RBs in one RBG, N_{symbol} is the number of OFDM symbols for data transmission in one RB, and T_{RBG} is the time duration of one RBG. Since $\mathrm{MO}^t_{m,n}\mathrm{CR}^t_{m,n} N_{\mathrm{RB}} N_{\mathrm{symbol}}$ represents the useful bits carried by one RBG of cell m at time t, we can divide it by T_{RBG} to calculate the user throughput provided by one RBG.

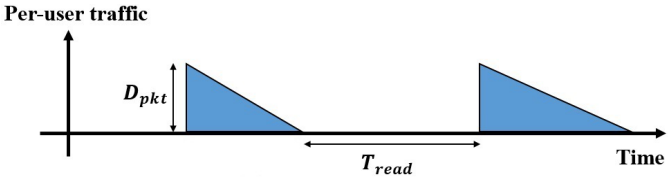


Fig. 5. Traffic generation per user.

D. Scheduling and User Throughput

Without loss of generality, we adopt Resource Allocation Type 0 in 3GPP TS 136.213 [32]. Resource Allocation Type 0 divides RBs into multiple RBGs and the resource allocation is done at the level of RBG. The number of RBs in each RBG varies depending on the system bandwidth. The scheduler allocates RBGs based on the proportional fair scheduling algorithm [33], [34], which is designed to maintain a balance between two competing interests: maximizing network throughput and achieving fairness among users. Specifically, at time t, the scheduler of cell m allocates each RBG to one user n according to the following priority function:

$$n = \underset{n' \in \Psi^t_m}{\operatorname{argmax}} \frac{r^t_{m,n'}}{\overline{R}^t_{n'}}, \tag{17}$$

where Ψ^t_m is cell m's connected users that have unsent downlink data at time t, $r^t_{m,n'}$ is defined in (16), and $\overline{R}^t_{n'}$ is the average throughput of user n' until time t. Note that $\overline{R}^t_n$ is updated during scheduling to achieve the fairness between connected users, i.e., if user n gets one RBG, then $\overline{R}^t_n$ is increased when deciding the next RBG allocation. Therefore, a balance between the maximum throughput and the average throughput can be achieved. Note that the time-scale of scheduling in LTE systems is 1 ms.

According to the scheduling result, let $W^t_{m,n}$ be the number of RBGs that allocate to user n by cell m at time t. The instantaneous throughput of user n at time t is calculated by

$$R^t_n = W^t_{m,n} r^t_{m,n}. \tag{18}$$

E. User Traffic Model

In real-world scenarios, the data transmission required by a user usually consists of several sequences of packets with periods of inactivity in between [35]. Therefore, we adopt the traffic model 2 defined in 3GPP [36], which is a non-full buffer model for generating the per-user traffic. Specifically, each user has bursty traffic as shown in Fig. 5, where D_{pkt} is the packet size, the reading time T_{read} is the time duration between the end of download of the previous packet and the request for the next packet. The reading time T_{read} follows an exponential distribution as follows:

$$f_{T_{\mathrm{read}}}(t_{\mathrm{read}}) = \frac{1}{\lambda} e^{-\frac{t_{\mathrm{read}}}{\lambda}}, t_{\mathrm{read}} \geq 0 \tag{19}$$

where T_{read} is the random variable of the reading time, t_{read} is the possible realization of T_{read}, $f_{T_{\text{read}}}(t_{\text{read}})$ is the probability density function of the reading time, and λ represents the mean of the reading time. D_{pkt} and λ can be set to be different values to capture a range of real-world user traffic types.

We assume that there is a queue for each user to save the unsent data. The queue is updated in each scheduling cycle, and the speed of emptying the queue scales with the user throughput. Once a user's queue is empty, the user will download another packet with size D_{pkt} after T_{read}.

V. Performance Evaluations

In this section, we evaluate the performance of the introduced decentralized DRL-based MLB algorithm (DecenDRL). We compare with four methods: 1) The centralized DRL-based MLB method with the objective of minimizing the maximum load [17] (CenDRL-Max). 2) The centralized DRL-based MLB method with the objective of minimizing the standard deviation of load (CenDRL-SD). 3) The rule-based CIO tuning method [7] that uses a fixed step size to adjust CIO values. 4) The baseline method that does not adjust the CIO values and the antenna tilt angles

The introduced DecenDRL uses the decentralized action in (8) and individual reward function in (11). Both CenDRL-Max and CenDRL-SD use the centralized action in (7), but CenDRL-Max uses the reward function in (9) and CenDRL-SD uses the reward function in (10). The baseline method randomly initializes antenna tilt angles and sets all CIO values to be zeros. The rule-based CIO tuning method compares the load between any two cells m and m', $\overline{\rho}_m[k]$ and $\overline{\rho}_{m'}[k]$, and then adjusts the CIO values as follows:

$$\begin{cases} \text{If } \overline{\rho}_{m'}[k] < \overline{\rho}_m[k] - \epsilon, & O_{m,m'}[k+1] = O_{m,m'}[k] - \Delta O. \\ \text{If } \overline{\rho}_{m'}[k] > \overline{\rho}_m[k] + \epsilon, & O_{m,m'}[k+1] = O_{m,m'}[k] + \Delta O. \\ \text{Otherwise,} & O_{m,m'}[k+1] = O_{m,m'}[k] \end{cases} \tag{20}$$

The rule-based CIO tuning method adjusts the CIO value by a fixed step size ΔO if the difference between two cell loads is larger than the threshold ϵ. In the experiment, we set ϵ and ΔO to be 0.05 dB and 0.5 dB, respectively.

We evaluate different MLB methods on the scenario of 6 small cells and 60 mobile users in 400 m $\times$ 400 m area. The user mobility trajectories are generated by the SLAW mobility model. We initialize the user-cell connection by connecting each user to the cell that has the highest RSRP. The scheduling period is 1 ms in LTE systems, and the period of tuning CIO and tilt angles is set to be 30 seconds. For the A3 event, the range of CIO value is set to be [-5 dB, 5 dB] and the hysteresis is set to be 1 dB. We assume that the system bandwidth is 20 MHz. According to the relationship between RBG size and the system bandwidth in 3GPP TS 136.213 [32], 20 MHz system bandwidth corresponds to 25 RBGs in each cell, where one RBG is formed by 4 RBs. Meanwhile, the number of OFDM symbols of one RB for data transmission is 132. For the user traffic model, the packet size is set to be 8 Mb and the mean reading time is set to be 2 seconds. The transmit powers of all cells are set to be 500 mW. The noise power is set to be 3.8×10^{-16}. We list all parameters in the cellular environment in Table II.

TABLE II
The Parameter Values in the Cellular Environment

Parameter	Value
Number of cells (M)	6
Number of users (N)	60
Simulation area	400 m $\times$ 400 m
Evaluation time	24 hr
User mobility model	SLAW mobility model
CIO range ($[-O_{\max}, O_{\max}]$)	[-5 dB, 5 dB]
Handover hysteresis value of cell m (H_m)	1 dB
CIO/Tilt tuning period (T)	30×10^3 ms
Scheduling period	1 ms
System bandwidth	20 MHz
Number of RBGs of cell m (W_m)	25
Number of RBs per RBG (N_{RB})	4
Number of data symbols per RB (N_{symbol})	132
Time duration of one RBG (T_{RBG})	0.5 ms
Packet size (D_{pkt})	8 Mb
Mean reading time (λ)	5 s
Transmit power of cell m (P_m)	500 mW
Noise power (N_0)	3.8×10^{-16} mW

Each cell has a DRL agent for the DecenDRL method, while there is only one central DRL agent to control all cells in the network for both the CenDRL-Max method and the CenDRL-SD method. For a fair comparison, we set the same DDPG architecture for the underlying DRL agent of DecenDRL, CenDRL-Max, and CenDRL-SD. To be specific, both the actor network and the critic network of each DDPG-based DRL agent are fully-connected networks with two hidden layers, and the number of neurons is set to be 50 for each hidden layer. All the hidden layers are followed by rectified linear units (ReLU). The output layer of the actor network uses the hyperbolic tangent (tanh) activation function. Compared to a large neural network with tens of thousands of neurons in most DRL applications, we use a relatively small neural network of 50 neurons in each layer to prevent the issue of overfitting for a cellular network with conventional hexagonal structures. Since the state/action space of the MLB problem is linearly dependent on the number of neighboring cells which is 6 for the hexagonal cellular structure, 50-dimensional vector space should be sufficiently large for the MLB task. The learning rates for actor and critic networks are set to 0.001 and 0.01, respectively. The τ value for the soft target network update is set to be 0.01. The training batch size is set to be 64. The discount factor γ is set to be 0.99. The exploration noise $\mathcal{W}$ is a zero-mean Gaussian noise, where the initial variance of $\mathcal{W}$ is 0.1, and it will decrease exponentially to 0.001 at the end of training.

We evaluate different MLB methods with three performance metrics: 1) the standard deviation of cell load; 2) the maximum of cell load; 3) the RLF probability of users. The standard deviation of cell load and the maximum of cell load represent the load balancing performance after tuning CIO values and antenna tilt angles. The RLF probability of users is calculated by the ratio of disconnected users versus the total number of users. The unprocessed results are noisy due to the dynamic nature of networks. Therefore, the results are moving averaged to eliminate short-term fluctuations and to highlight long-term

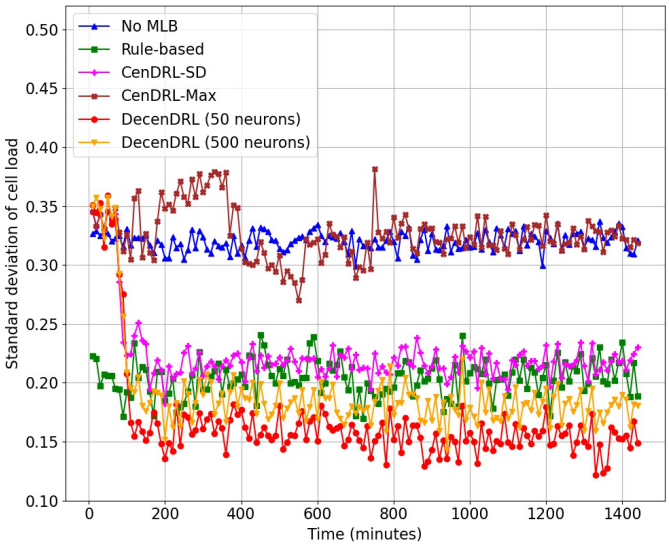


Fig. 6. Standard deviation of cell load (6 cells, 60 users).

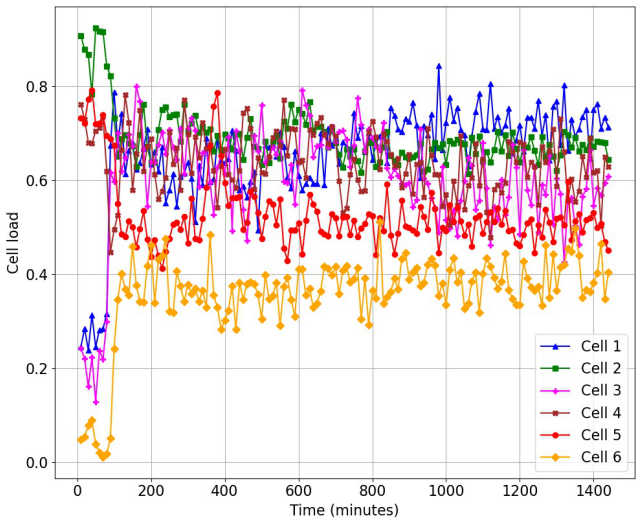


Fig. 7. Individual cell load of DecenDRL (6 cells, 60 users).

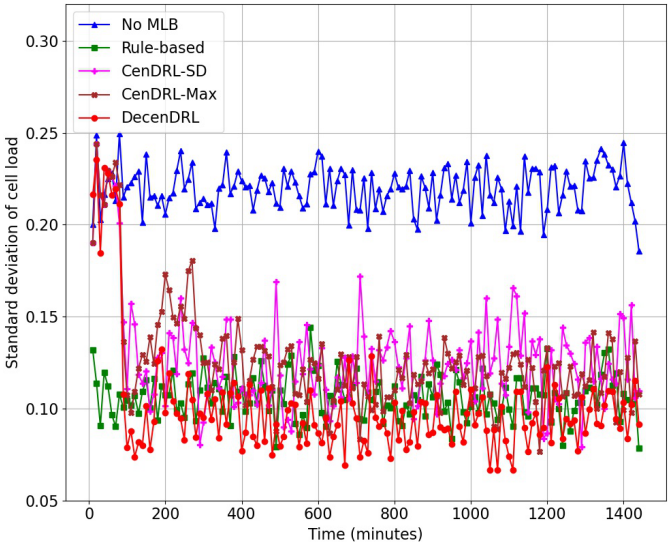


Fig. 8. Standard deviation of cell load (3 cells, 30 users).

trends. Specifically, the results in this section are averaged over the previous 10 minutes.

Fig. 6 shows the standard deviation of cell load over time of different MLB methods. The introduced DecenDRL has the lowest cell load standard deviation, which shows that the decentralized DRL-based method can achieve a more balanced cell load distribution compared to other methods. Meanwhile, we evaluate the impacts of the number of neurons for each hidden layer on the system performance. To be specific, we let the number of neurons for each hidden layer in the DecenDRL method to be 50 and 500 respectively and compare the corresponding network performance. From Fig. 6, we can observe that using more neurons does not necessarily help balance the network load due to the fact that large neural networks usually suffer from overfitting issues. On the other hand, CenDRL-Max cannot decrease the standard deviation of cell load effectively because the reward is only meaningful to the most overloaded cell. Although CenDRL-SD is better, it is still worse than the rule-based method and DecenDRL. These results verify that decentralized learning is more effective to the load balancing problem than centralized learning. The rule-based method is also a decentralized approach that adjusting the CIO values locally in different areas of the network. However, the load balancing performance of the rule-based method is worse than DecenDRL because a fixed step size of CIO adjustment cannot cover all different scenarios in the dynamic cellular network. Furthermore, it is difficult to determine the optimal step size of CIO adjustment beforehand. Fig. 7 shows the result of each cell's load in DecenDRL. We can observe that the cell load of DecenDRL converges at around 150 minutes, which corresponds to collecting 300 training samples from the environment.

Fig. 8 shows the standard deviation of cell load over time of different MLB methods in a small-scale network that only has 3 cells and 30 users. When the number of cells is relatively small, centralized and decentralized learning have similar load balancing performance. The reason is that a single reward function can represent the load balancing performance in a small-scale network better.

The results of the maximum cell load over time of different MLB methods are shown in Fig. 9. We can observe that DeCenDRL has the lowest maximum cell load even if its objective is not to minimize the maximum cell load. The reason is that the maximum cell load is closely related to all cells' control parameters. In other words, the maximum cell load can be decreased most effectively if all cells collaborate on achieving a more balanced cell load distribution in every part of the network. Although CenDRL-Max's objective is to minimize the maximum cell load, it only learns how to adjust control parameters of the most overloaded cell from its reward function. Therefore, the traffic of the most overloaded cell cannot be transferred to other cells effectively.

An edge user is the user that receives a weak signal from its connected cell. The worst case for an edge user is that the RLF happens and its radio link is disconnected. Maintaining the performance of edge users is important. If many users are

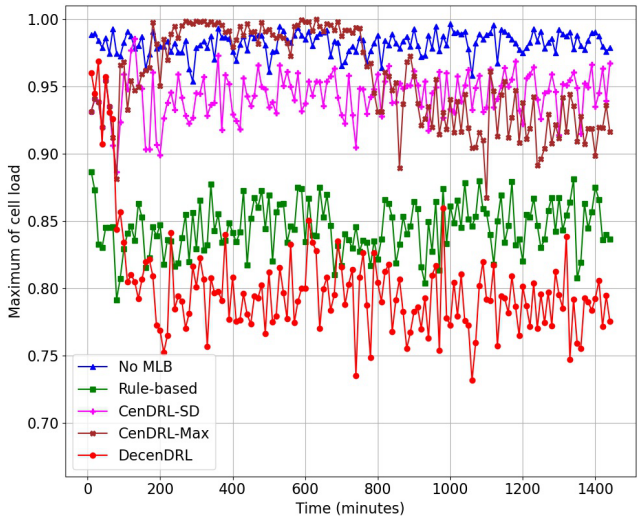


Fig. 9. Maximum cell load (6 cells, 60 users).

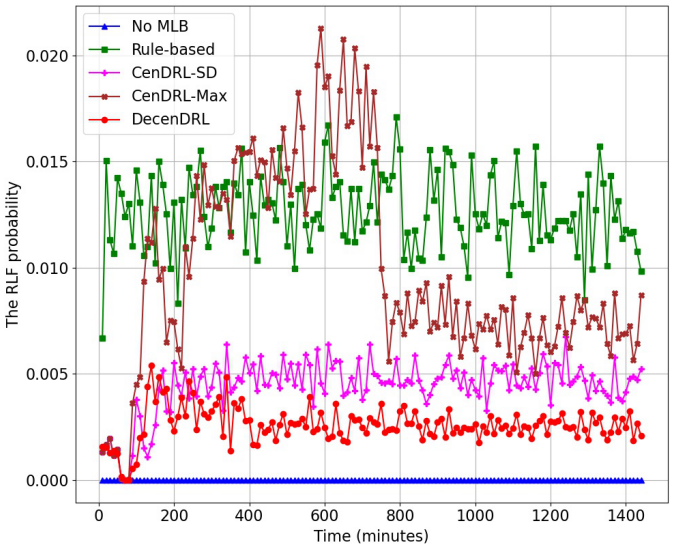


Fig. 10. The RLF probability of users (6 cells, 60 users).

disconnected from their serving cells, then the cell load can be decreased because fewer users compete for resources, which is not a desirable outcome that we want to see. Therefore, we calculate the RLF probability of each method, and the results are shown in Fig. 10. The baseline method without changing any control parameters has the lowest RLF probability because all CIO values are set to be zeros, which means that every user connects to the cell has the strongest signal strength. The rule-based method has the highest RLF probability because it cannot adjust CIO values quickly to adapt to the dynamic environment. When users change their clustered locations, the delayed CIO adjustments already result in RLFs of many users. CenDRL-Max has the second-highest RLF probability because the maximum load can be decreased if many RLFs happen to connected users of the most overloaded cell. Furthermore, the standard deviation of cell load can be decreased if many RLFs happen in the area crowded with users, so CenDRL-SD also has a higher RLF probability than DecenDRL. On the other hand, the objective of DecenDRL is to minimize the load difference with other cells, which cannot be directly achieved by making more RLFs happen.

It is important to note that since our evaluation is based on available field measurements in an area within a city, the number of cells is limited to 6. For this relatively small number of cells, we can already see the performance benefits of our introduced DecenDRL over the centralized one. In fact, as the number of cells increase, we are expecting that the performance benefits will be even more significant. This is because the DRL agents of both CenDRL-Max and CenDRL-SD will have to control an overwhelmingly large number of MLB parameters. On the other hand, each DRL agent in our introduced DecenDRL only takes neighboring cells' exchanged information as the input state and controls the MLB parameters with neighboring cells. Therefore, the load balancing performance of the decentralized method will be scaled and the size of the network will not have a significant impact on the performance.

VI. Conclusion

In this paper, we introduce a novel decentralized DRL-based MLB method, where the DRL agent of each cell controls its CIO values and antenna tilt angle to balance the cell load distribution. When the number of cells is large, the decentralized DRL-based MLB method is more computationally efficient for learning MLB control parameters of each cell compared to existing centralized DRL-based MLB methods. Our designed decentralized DRL-based MLB architecture utilizes individual rewards that reflect the local cell load distribution in different local parts of the network to achieve better load balancing performance than centralized DRL-based MLB methods. Furthermore, our designed decentralized DRL-based MLB architecture can be readily scaled to a large size cellular network because it does not require additional signaling overheads compared to existing cellular standards. To evaluate MLB algorithms, we build a near real-world cellular environment by developing a new network simulator, where the mobility management procedures strictly follow 3GPP specifications and the underlying cellular environment is obtained by field measurements. Extensive performance evaluations show that the decentralized DRL-based method can achieve the most balanced cell load distribution and the best performance of edge users compared to other state-of-the-art MLB methods.

References

[1] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*, VNI Global Fixed Mobile Internet Traffic Forecasts, Cisco, San Jose, CA, USA, Feb. 2019.

[2] *Self-Organizing Networks (SON); Concepts and Requirements*, 3GPP, document TS 32.500, Jun. 2018.

[3] *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall Description; Stage 2*, 3GPP, document TS 36.300, Jan. 2021.

[4] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5G: How to empower SON with big data for enabling 5G," *IEEE Netw.*, vol. 28, no. 6, pp. 27–33, Nov./Dec. 2014.

[5] J. Andrews, S. Singh, Q. Ye, X. Lin, and H. Dhillon, "An overview of load balancing in HetNets: Old myths and open problems," *IEEE Wireless. Commun.*, vol. 21, no. 2, pp. 18–25, Apr. 2014.

[6] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[7] R. Kwan, R. Arnott, R. Paterson, R. Trivisonno, and M. Kubota, "On mobility load balancing for LTE systems," in *Proc. IEEE 72nd Veh. Technol. Conf.-Fall*, Sep. 2010, pp. 1–5.
[8] M. M. Hasan, S. Kwon, and J.-H. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2205–2217, Apr. 2018.
[9] Z. Becvar and P. Mach, "Adaptive hysteresis margin for handover in femtocell networks," in *Proc. 6th Int. Conf. Wireless Mobile Commun.*, Sep. 2010, pp. 256–261.
[10] D. Lopez-Perez, I. Guvenc, and X. Chu, "Mobility management challenges in 3GPP heterogeneous networks," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 70–78, Dec. 2012.
[11] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. Zhang, "Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 212–217, Apr. 2019.
[12] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Sep. 2017.
[13] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.
[14] L. Xiao *et al.*, "Reinforcement learning-based downlink interference control for ultra-dense small cells," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 423–434, Jan. 2020.
[15] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, Apr. 2019.
[16] L. Li *et al.*, "Accelerating model-free reinforcement learning with imperfect model knowledge in dynamic spectrum access," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7517–7528, Aug. 2020.
[17] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load balancing for ultradense networks: A deep reinforcement learning-based approach," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9399–9412, Dec. 2019.
[18] G. Alsuhli *et al.*, "Mobility load management in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, early access, Aug. 30, 2021, doi: 10.1109/TMC.2021.3107458.
[19] L. Liu, R. Chen, S. Geirhofer, K. Sayana, Z. Shi, and Y. Zhou, "Downlink MIMO in LTE-advanced: SU-MIMO vs. MU-MIMO," *IEEE Commun. Mag.*, vol. 50, no. 2, pp. 140–147, Feb. 2012.
[20] A. Beylerian and T. Ohtsuki, "Multi-point fairness in resource allocation for C-RAN downlink CoMP transmission," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, pp. 1–10, Dec. 2016.
[21] L. Liu, J. Zhang, B. Clerckx, and J.-K. Han, "Coordinated multipoint (CoMP) joint transmission using channel information feedback and higher rank dedicated beam-forming," U.S. Patent 8 442 566, May 14, 2013.
[22] S. Mosleh, L. Liu, and J. Zhang, "Proportional-fair resource allocation for coordinated multi-point transmission in LTE-advanced," *IEEE Trans. Wireless Commun.*, vol. 15, no. 8, pp. 5355–5367, Aug. 2016.
[23] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol Specification*, 3GPP, document TS 36.331, Jan. 2021.
[24] B. Yu, L. Yang, H. Ishii, and X. Cheng, "Load balancing with antenna tilt control in enhanced local area architecture," in *Proc. IEEE 79th Veh. Technol. Conf. (VTC Spring)*, May 2014, pp. 1–6.
[25] A. J. Fehske, H. Klessig, J. Voigt, and G. P. Fettweis, "Concurrent load-aware adjustment of user association and antenna tilts in self-organizing radio networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 5, pp. 1974–1988, Jun. 2013.
[26] *LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 Application Protocol (X2AP)*, 3GPP, document TS 36.423, Nov. 2021.
[27] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Rep. (ICLR)*, 2016, pp. 1–14.
[28] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2000, pp. 1008–1014.
[29] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," in *Proc. NeurIPS Deep Learn. Workshop*, 2013, pp. 1–9.
[30] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: Self-similar least-action human walk," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 515–529, Apr. 2012.
[31] A. Chiumento, M. Bennis, C. Desset, L. V. der Perre, and S. Pollin, "Adaptive CSI and feedback estimation in LTE and beyond: A Gaussian process regression approach," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, no. 1, p. 168, Jun. 2015.
[32] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures*, 3GPP, document TS 36.213, Jan. 2021.
[33] L. Liu, Y.-H. Nam, and J. Zhang, "Proportional fair scheduling for multi-cell multi-user MIMO systems," in *Proc. 44th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2010, pp. 1–6.
[34] H. J. Kushner and P. A. Whiting, "Convergence of proportional-fair sharing algorithms under general conditions," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1250–1259, Jul. 2004.
[35] P. Ameigeiras, Y. Wang, J. Navarro-Ortiz, P. E. Mogensen, and J. M. Lopez-Soler, "Traffic models impact on OFDMA scheduling design," *EURASIP J. Wireless Commun. Netw.*, vol. 2012, no. 1, p. 61, Dec. 2012.
[36] *Further Advancements for E-UTRA Physical Layer Aspects (Release 9)*, 3GPP, document TR 36.814, Mar. 2010.

Hao-Hsuan Chang received the B.Sc. degree in electrical engineering and the M.S. degree in communication engineering from the National Taiwan University, Taiwan, in 2013 and 2015, respectively, and the Ph.D. degree in electrical and computer engineering from Virginia Tech, USA, in 2021. He is currently the Senior Research Engineer at Samsung Research America. His research interests include dynamic spectrum access, deep reinforcement learning, and machine learning for wireless communications.

Hao Chen (Member, IEEE) received the B.S. and M.S. degrees in information engineering from Xi'an Jiaotong University, Shaanxi, in 2010 and 2013, respectively, and the Ph.D. degree in electrical engineering from The University of Kansas, Lawrence, KS, USA, in 2017. Since 2016, he has been a Research Engineer with the Standards and Mobility Innovation Laboratory, Samsung Research America. His research interests include network optimization, machine learning, and 5G cellular systems.

Jianzhong (Charlie) Zhang (Fellow, IEEE) received the Ph.D. degree from the University of Wisconsin, Madison, USA. He is currently a SVP and the Head of the Standards and Mobility Innovation Laboratory, Samsung Research America, where he leads research, prototyping, and standards for 5G and future multimedia networks. From 2009 to 2013, he has worked as the Vice Chairperson of 3GPP RAN1 and led development of LTE and LTE-advanced technologies, such as 3D channel modeling, UL-MIMO, CoMP, and carrier aggregation for TD-LTE.

Lingjia Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering from Shanghai Jiao Tong University and the Ph.D. degree in electrical and computer engineering from Texas A&M University, USA. He is currently a Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, USA. He is also serving as the Associate Director for Wireless@Virginia Tech and an Executive Committee Member for the National Spectrum Consortium (NSC). His research interests include machine learning for wireless communications, enabling technologies for 6G, mobile edge computing, and the Internet of Things.