Energy-Efficient Deep Reinforcement Learning Assisted Resource Allocation for 5G-RAN Slicing

Yaser Azimi, Saleh Yousefi, Hashem Kalbkhani, and Thomas Kunz, *Senior Member, IEEE*

***Abstract*—One of the pillars of the 5G architecture is network slicing, in which hardware, radio, and power resources are virtualized as a logical network taking into account the requirements of diverse applications. While ensuring performance isolation among different slices, resource allocation in 5G Radio Access Networks (RANs) is associated with different challenges due to network dynamics and the different applications' requirements. In this paper, we have considered the allocation of power and radio resources to rate-based as well as resource-based users. We propose an energy-efficient deep reinforcement learning-assisted resource allocation (EE-DRL-RA) method for RAN slicing in 5G networks. The main idea of the proposed method is to exploit a collaborative learning framework that includes deep reinforcement learning (DRL) and deep learning (DL) to decide on resource allocation in the RAN. Specifically, we use DL for decision-making on resource allocation on a large time-scale and DRL for decision-making on resource allocation on a small time-scale. The asynchronous advantage actor-critic (A3C) and the stacked and bidirectional long-short-term-memory (SBiLSTM) network are used as DRL and supervised DL methods, respectively. Furthermore, we determine the optimal power and resource blocks (RBs) for rate-based users by formulating the energy-efficient power allocation (EE-PA) problem as a non-convex optimization problem and solve it by an efficient iterative algorithm. Our proposed approach is unique in that it simultaneously allocates power and RBs while ensuring slice isolation with low computational and time complexity. Simulation results show that EE-DRL-RA yields better performance compared to a state-of-the-art published method in terms of convergence speed, computational complexity, energy efficiency, and the number of accepted users as well as the degree of inter-slice isolation.**

***Index Terms*—Network slicing, 5G, reinforcement learning, deep learning, A3C, LSTM, power allocation, RAN slicing.**

I. Introduction

THE fifth generation of mobile technology (5G) is introduced to address the commercial requirements of infrastructure service providers and the demands of users in 2020 and beyond. As envisaged in [1], its economic value will grow to more than $12.3 trillion by 2035. 5G enables a new type of end-to-end network that can realize a fully mobile and connected society.

The 5G system supports diverse new use cases with different requirements. Typically, it can be divided into three types: enhanced mobile broadband connectivity (eMBB), massive machine-type communications (mMTC), and ultra-reliable low latency communications (uRLLC) [2]. mMTC and uRLLC have very different requirements compared to eMBB. The mMTC applications are characterized by a massive number of devices, low volume data transmissions, delay tolerance, and low power consumption [3]. Different from mMTC, uRLLC applications require higher throughput and low latency for real-time interaction. eMBB applications are characterized by higher capacity, high data rates, and higher user mobility across a wide coverage area. Nowadays, due to tremendous growth in the number of users, the density of traffic, various applications, and business models we need a network that provides better performance with larger connectivity density, much higher throughput, much lower latency, higher mobility range, and ultra-high reliability with regard to security, trust, and privacy compared to the current network [4].

Network slicing is introduced as one of the key technologies of 5G that takes advantage of enablers such as network virtualization, mobile edge computing, and software-defined networks, and provides various network services according to users' needs [5]. Each slice is able to independently adjust network functions and allocate the corresponding network resources with regard to the needs of commercial scenarios and traffic models, improving flexibility, reliability, and robustness in the whole network [4]. Network slicing can divide a shared physical network into multiple logical virtual networks, to optimize the allocation of various resources and to appropriately support different users of different services. It is considered as a logical end-to-end network that can flexibly provide one or more network services according to the slice requirements. Each network slice guarantees dedicated resources such as virtual computational resources, resource blocks (RBs), network bandwidth, and transmission power for each user in the slice. As the slices are isolated from each other, disruption of a slice does not affect the functionality of other slices [6].

A network slice includes a radio access network (RAN) and a core network (CN). So far, the slicing of 5G core networks has been extensively investigated, but relatively little emphasis has been placed on RAN slicing. Resource scheduling is one key issue in RAN slicing. The limited resources must be allocated to different users with various quality of service (QoS)

Manuscript received April 28, 2021; revised September 23, 2021; accepted November 8, 2021. Date of publication November 16, 2021; date of current version January 20, 2022. The review of this article was coordinated by Dr. Sudip Misra. *(Corresponding author: Saleh Yousefi.)*

Yaser Azimi and Saleh Yousefi are with the Department of Electrical and Computer Engineering, Urmia University, Urmia 57561-51818, Iran (e-mail: y.azimi@urmia.ac.ir; s.yousefi@urmia.ac.ir).

Hashem Kalbkhani is with the Faculty of Electrical Engineering, Urmia University of Technology, Urmia 57166-17165, Iran (e-mail: h.kalbkhani@uut.ac.ir).

Thomas Kunz is with the Department of Systems and Computer Engineering, Carleton University, Ottawa K1S 5B6, Canada (e-mail: tkunz@sce.carleton.ca).

Digital Object Identifier 10.1109/TVT.2021.3128513

requirements with regard to changes in the traffic and network state dynamics. Compared to core network slicing, resource scheduling in RAN slicing is much more challenging, considering the radio channel conditions and user mobility [7].

In this paper, we propose a distributed resource allocation algorithm to allocate power and RBs to each user in a real-time manner. Due to time-varying channel conditions, we allocate power and RBs by considering energy efficiency for each user in each slice. In the proposed energy-efficient deep reinforcement learning assisted resource allocation (EE-DRL-RA) method, both deep reinforcement learning (DRL) and deep learning (DL) work together in a collaborative manner, to address the resource allocation for both small and large timescales. In summary, we use the stacked and bidirectional long-short-term-memory (SBiLSTM) method as a supervised DL method to predict data traffic based on traffic patterns and allocate resources to RAN slices on a large time-scale. The accuracy of SBiLSTM depends on the amount of input data, so for dealing with unexpected changes in traffic and inaccurate predictions a distributed framework based on the asynchronous advantage actor-critic (A3C) method is used to allocate resources on a small time-scale. Compared to SBiLSTM, online DRL does not require as much data to make decisions and to respond to changes in the dynamic environment quickly. As more data becomes available over time, learning progresses and the algorithms (DRL and DL) will become more accurate. Furthermore, we propose the EE-PA method for determining the amount of required power and RBs to rate-based users, which reduces the power consumption, increases the number of accepted users and maximizes energy efficiency. The main contribution in this paper and some advantages of the proposed method are summarized as follows:

- The EE-DRL-RA method is a powerful and accurate resource allocation algorithm to allocate power and RBs using DRL and DL. It employs a distributed framework based on the A3C method [8], which is model-free and robust to unexpected changes in the environment.
- The EE-DRL-RA method employs the stacked and bidirectional LSTM (SBiLSTM), which has a higher accuracy than the traditional LSTM, to predict the required resources for each slice on a large time-scale. We also use one shared SBiLSTM between the identical slices in several A3C blocks, which improves the accuracy of SBiLSTM and achieves faster convergence by using more input data.
- The EE-DRL-RA method employs the EE-PA method, which is proposed for determining RBs and power to rate-based users. The problem of the resource determination in rate-based slices with considering energy efficiency, transmission rate, transmission power, and channel conditions is a non-convex optimization problem.
  Therefore, we have used an iterative algorithm with fast convergence speed, based on the gradient descent method, to solve the problem.
- The complexity of the proposed distributed algorithm does not depend on the network size (number of slices, actors, and critics). The parallel processing of the A3C algorithm allows us to decide resource allocations independently for each slice. Also, due to the isolation of each slice, each slice can be removed or added without affecting other slices. As a result, EE-DRL-RA offers a good degree of scalability and can be applied to real-life scenarios in terms of time complexity and computational cost.

The organization of the rest of this paper is as follows. In Section II, network slicing approaches are reviewed and the related work is summarized. Section III presents the system model of the proposed method. In Section IV, the problem definition and formulation of resource allocation for RAN slicing is proposed as optimization problems. We present the system architecture of the EE-DRL-RA method using SBiLSTM, A3C, and EE-PA methods in Section V. Then, the simulation results are provided in Section VI and finally, Section VII concludes the paper.

II. Related Work

In this section, we review the previous work on resource allocation in RAN. Recently, machine learning (ML) has been considered as an approach for slicing in 5G-RAN, whereas most previous RAN slicing proposals use non-ML approaches. In this section, we have mainly concentrated on resource allocation methods based on machine learning in network slicing.

Many proposals do not consider isolation and smart power allocation. Furthermore, some proposed methods have high computational complexity. In [9], an online network slicing solution based on the multi-armed bandit model was presented to maximize network slicing efficiency by accepting requests above the available capacity of the network. Some articles have used online Q-Learning algorithms to manage and distribute radio resources between different slices [8]. In online methods, such as SARSA and online Q-Learning, the estimated value of a policy is used in the environment immediately. This may lead to poor or unacceptable decisions that can disrupt the operation of the system. To avoid this, in [10], an offline Q-Learning was presented for allocating resources to users of eMBB and vehicle-to-vehicle (V2V) services in both uplink (UL) and downlink (DL) channels. In this method, a heuristic algorithm is proposed to allocate RBs to different slices, aiming to maximize the resource utilization while satisfying the requirements of the traffic of each RAN slice. Also, a softmax decision-making is used to select the best action, which can trade-off between exploration and exploitation rather than the $''\epsilon - Greedy''$ method. In [11], the authors have proposed an admission control algorithm to network slicing. However, the proposed learning algorithm is an offline approach, which is not suitable for a 5G environment that is constantly changing. In [12], the authors presented an economic model to maximize the overall profit of the network by using a decision-making strategy based on the continuous-action-based Actor-Critic (AC) method for the admission control problem. However, the proposed method is not demand-driven and requires a thorough knowledge of incoming request statistics, so this method is impractical in real environments. To tackle this issue, in [13], a neural network has been used to estimate the Q-value, which is the expected future reward, and increases the convergence rate. Although the proposed method can be adapted to changes in the environment, it is practically impossible to use this method due to its scalability problem in conditions where

the state space is huge. In [14], multi-agent RL was used to solve the multi-radio access technology problem. Their results show that RL performs better in a time-varying network environment than other solutions. In [15], a deep RL-based method for the resource allocation between the slices was presented using two neural networks to accurately estimate the Q-value and select the best action. In [16], a method called DNAF is presented that uses deterministic policy gradient descent (DPGD) to prevent unnecessary calculations of Q values. Besides, DPGD only performs in a continuous action space, so the proposed method uses k-nearest-neighbor (kNN) to find the nearest action in the continuous space to prevent the recalculation of Q values. In [17], a deep intent-based network slicing system was designed to slice and manage the core network and RAN resources. In [18], a resource allocation method based on deep dueling Q-learning was introduced to allocate spectrum, computing, and storage resources to various users. Deep dueling Q-learning does not update the Q-value for unnecessary actions, which achieves the optimal policy faster than the conventional Q-learning algorithm. In [19], a deep distributed Q network (DDQN) based on the generative adversarial network (GAN) was proposed to improve the quality of experience (QoE) and spectral efficiency (SE) of users in each slice.

In [7], an intelligent resource planning scheduling (iRSS) for RAN slicing was proposed by combining deep learning for decision making on a large time-scale and reinforcement learning for decision making on a small time-scale to guarantee isolation of the slices. In this paper, long short-term memory (LSTM) and A3C are used to predict the required RBs on a large time-scale in RAN slices and allocate the RB on a small time-scale dynamically, respectively. In this paper, it is assumed that the allocated power to users is constant, and the channel condition is not considered in the resource allocation. Using distributed and parallel A3C in the method has reduced the time and computational complexity.

Other proposals provide isolation but still suffer from high time and computational complexity while still ignoring power allocation. The authors in [20] proposed a model based on the advantage actor-critic (A2C) algorithm and LSTM to improve system performance for mobile users. In this method, RBs are allocated based on the channel condition and the level of quality of service (QoS) of users. In [21], a policy for bandwidth-greedy communication services was proposed that solves the optimization problem by using MDP while guaranteeing QoS requirements and slice isolation. In [22], a Markovian approach was designed for resource allocation in multi-tenant scenarios with various guaranteed bit rate services with regard to the admission control policy.

Another group of proposals addresses the power allocation and energy efficiency problems but does not provide slice isolation. In [23], the resource allocation problem was solved by discrete-action-based Q-learning. In [24], a federated learning model was introduced that incorporates actor-critic (AC) to power allocation. Federated learning trains a shared network model across many participating edge devices while keeping all the training data locally. In this method, the actor network is used for sharing weights and gradients between the shared network model and edge devices.

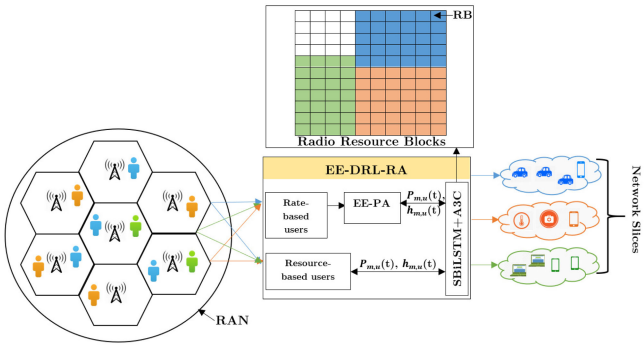

Fig. 1. Network Slicing Model.

Some proposals have high time and computational complexity and ignore the energy efficiency while allocating power. In [25], a multi-agent reinforcement learning solution using a deep Q-network was proposed to share the spectral resources and allocate the power to V2V users in RAN. In [26], a method called SDR was proposed which uses RL to increase the throughput and reduce the power consumption and transmission power for devices and adjusts the coding and modulation. In [27], a dynamic virtual resource allocation scheme in RAN was proposed for eMBB, uRLLC, and mMTC communications with regard to QoS. To address sub-channel allocation and power control, a joint optimization problem as a continuous MDP problem. In [28], a RAN slice selection mechanism was introduced considering transmission rate, delay, and blocking rate and solved by dynamic programming. In [29], a slice-based virtual resource scheduling scheme for eMBB and uRLLC was proposed to maximize the total user rate. Power allocation and subcarrier allocation are formulated as a constrained MDP problem based on policy gradient AC learning.

To the best of our knowledge, the presented algorithms until now have not allocated power and RB resources simultaneously (considering the isolation of each slice and channel conditions). Many of them are not practically feasible in terms of scalability, time complexity and computational cost. Therefore, in this paper, we propose an energy-efficient deep reinforcement learning assisted resource allocation (EE-DRL-RA) method to allocate power and spectrum resources in RAN according to the channel conditions while ensuring the isolation of each slice.

III. System Model

As shown in Fig. 1, we study a system where multiple virtual networks (slices) are overlaid on top of the shared physical network infrastructure. We consider the RAN to be a set of N base stations located in a geographical area, where RBs and power resources form a resource pool. It is assumed that channel state information (CSI), which is exchanged between BSs, can be perfectly known by the central unit (resource allocation algorithm) [30]. Imperfect CSI leads to an improper transmission power allocation which negatively affects the spectral and

energy efficiencies as well as the number of allocated resources for rate-based users. However, the existence of robust channel estimation methods helps us to have near-perfect CSI. Also, the overhead caused by CSI exchange and control signals is negligible compared to the resources used for data exchange [30].

We consider transmission power and RBs allocation in each RAN. Let M be the set of slices that share the RAN, and $U_m, m \in M$ be the set of users belonging to slice m. Also, the allocated resources to users in each slice are exclusive until they are released. We consider a sharing account book to record and share some important information among slices (i.e., states of the system, slices), and each slice can modify and maintain this account book.

In this paper, two types of resource requests have been considered:

1) resource-based requests in which the amount of required resources are previously specified and fixed, and
2) rate-based requests in which the amount of required resources depends on the transmission rate of a user and can vary.

Specifically, the resources are allocated to resource-based users according to given resource requirement (RBs and power) while the resources are allocated to rate-based users based on guaranteeing a minimum transmission rate and maximizing energy efficiency. We denote by $M_I \subset M$ the resource-based slices and by $M_{II} \subset M$ the rate-based slices. Also, arrival and departure rates of users in slice m are represented by $\lambda[m]$ and $\mu[m]$ respectively. Let the specification of slice m, $m \in M$ be represented by a three-tuple $\{r_m(t), h_m(t), T_m^{th}\}$ at time t, where $r_m(t)$ is the allocated amount of RBs, $h_m(t)$ is the required amount of RBs and T_m^{th} is the minimum required duration to guarantee isolation. As shown in Fig. 1, users of various slices enter the system with arrival rates $\lambda[m]$ and resources are allocated based on the type of slices (i.e., resource-based or rate-based). As mentioned earlier, it is assumed that resource-based users have a fixed amount of required power and RBs, and for rate-based users, the EE-PA method is used to determine the amount of required power and RBs based on channel conditions and transmission rate. Channel gain coefficients change in time-varying channels, hence, considering fixed transmission power cannot guarantee the QoS of users. Indeed, a low transmission power value may increase the outage probability. The simple solution is to consider the maximum available transmission power, but it increases the power consumption which in turn increases the interference level and operational costs of network providers. Hence, we should consider the energy efficiency over time-varying channels to set the optimum power level in an adaptive manner such that the user QoS requirements are satisfied. In Fig. 1, $P_{m,u}(t)$ and $h_{m,u}(t)$ represent the amount of required power and RBs for user u in slice m. The proposed SBiLSTM+A3C method determines the amount of required power and RBs. The path loss of the link between transmitter and receiver is $L(dB) = 10nlog_{10}(d) + C$, where n represents the path loss exponent, d denotes the distance between the transmitter and the receiver in meter, and C is a constant. The notations and symbols used throughout the paper are summarized in Table I.

TABLE I
NOTATION AND SYMBOL DEFINITIONS

Symbol	Description
Θ	the total of all RBs
P_{total}	the total of power resources
M	the set of slices
M_I	the set of rate-based slices
M_{II}	the set of resource-based slices
$U_m, m \in M$	the set of users belong to the slice m
T_Δ	Prediction Window (PW)
D_m	difference between the amount of resources before and after reconfiguration for slice m
$P_{m,u}(t)$	amount of required power for user u in slice m at time t
$N_{m,u}(t)$	Counter for counting amount of required RBs for user u in slice m at time t
$r_m(t)$	amount of allocated RBs for slice m at time t
$r_{m,u}(t)$	amount of allocated RBs for user u in slice m at time t
$h_m(t)$	amount of required RBs for slice m at time t
$h_{m,u}(t)$	amount of required RBs for user u in slice m at time t
η_{SE}	spectrum efficiency
η_{EE}	energy efficiency
$\lambda[m]$, $\mu[m]$	arrival and departure rates of users in slice m
T_m^{th}	the minimum threshold of the length of duration to ensure isolation for slice m

IV. PROBLEM DEFINITION AND FORMULATION OF RESOURCE ALLOCATION

To realize how resources are allocated in our model, let us first consider the concept of isolation of each slice. The isolation of each slice means that the allocated resources to each slice do not change over a specific period of time. To ensure the quality of service, a degree of isolation must be enforced to each slice so that the traffic load variation in a slice does not affect the other slices. To this end, we must reserve resources for future users of each slice over a specific period of time to avoid frequent reconfiguration of the resources of each slice. Accordingly, each slice possesses part of the total resources (Θ) exclusively for a certain period of time but can use residual resources if more resources are needed.

We have considered two time-scales, small and large, for allocating resources. On a small time-scale, decisions are made about whether or not to accept users in each slice. On a large time-scale, it is decided to allocate resources to each slice over a specific period of time. The amount of allocated resources to each slice on the large time-scale has a direct effect on the isolation of each slice. We consider a large time-scale as a prediction window (PW) that includes several small time-steps. Specifically, the resource allocation on a large time-scale is performed at the beginning of the next PW, and the resource allocation on a small time-scale at every time-step. Note that the length of PW and time-steps can be dynamically adjusted according to the specific requirements of slices of the mobile networks, i.e., 5G new radio (NR) features.

In Fig. 2, we show how to allocate resources to three slices in a typical scenario. In $t < \omega$, the values of $r_m(t)$ and $h_m(t)$ are equal in each slice because the resources are allocated to the slice according to its requirements. In these circumstances, the allocated resources of each slice need to be reconfigured

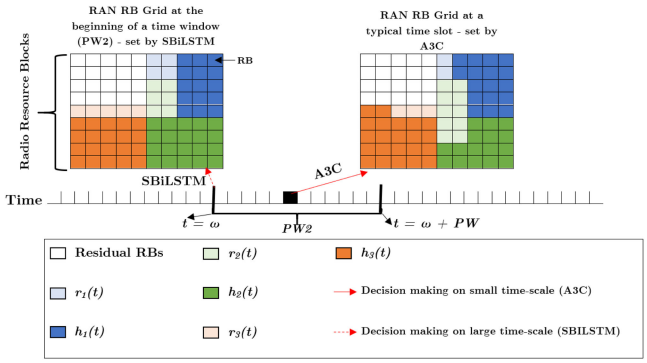

Fig. 2. RAN resource allocation to the slices on small and large timescales.

frequently to meet the requirements of users. So, the isolation of each slice is violated frequently. To ensure the isolation of each slice, the allocated resources to each slice must be estimated in the next PW. To this end, at the beginning of $PW2$, the allocated resources to each slice are estimated using the previous data. Obviously, each slice can use residual resources if the allocated resources are underestimated. For $\omega < t \leq \omega + PW$ the value of $r_m(t)$ will not be less than the predicted resources until the allocated resources to each slice are reconfigured at the beginning of $PW3$ at time $\omega + PW$. In the following, we formulate the resource allocation problem in the form of three problems at both the large and small time-scale.

Problem 1: The resource allocation on a large time-scale is performed by minimizing the mean-square-error (MSE) between the predicted value $\bar{r}_m(t)$ and the actual allocated resource amount $r_m(t)$, which is expressed as follows.

$$\arg\min_{\bar{r}_m(t)} \quad \frac{1}{T_\triangle}\sum_{t=1}^{T_\triangle} \left|r_m(t) - \bar{r}_m(t)\right|^2, \quad \forall m \in M \tag{1a}$$

$$s.t. \quad \sum_{m=1}^{M} \bar{r}_m(t) \leq \Theta \tag{1b}$$

where Θ is the total of all the RBs. Therefore, we may encounter two scenarios in the real-time resource allocation derived from problem (1): (a) if the amount of required resources $r_m(t)$ is more than the amount of predicted resources $\bar{r}_m(t)$, i.e., $r_m(t) > \bar{r}_m(t)$, then slice m may be required to ask for more resources from the residual resource. (b) If $r_m(t) \leq \bar{r}_m(t)$, then the allocated resources to the slice remain unchanged (until it may change in the next PW) for a long time to guarantee slice isolation.

Problem 2: To deal with unpredictable traffic changes on a large time-scale, we present on-line resource allocation on a small time-scale that aims at allocating the minimum amount of resources to users in each slice while guaranteeing the required isolation of each slice. Due to frequent requests for resource allocation in each slice, reconfiguration of the resources of each slice imposes additional overhead on the system and degrades the performance of the slices, therefore, a certain degree of isolation must be maintained. To this end, suppose $D_m(t) = |r_m(t) - r_m(t-\tau)|$ be the difference between the amount of resources before and after reconfiguration of the resources for slice m during $t-\tau$ to t, and $\Omega_m(D_m(t)=0)$ be a counter used to count the number of time steps that the amount of $r_m(t)$ has not changed. Thus, the formulation of on-line resource allocation on small time-scale can be expressed as follows:

$$\min \quad r_{m,u}(t), \quad \forall m \in M \tag{2a}$$

$$s.t. \quad \sum_{m=1}^{M} r_{m,u}(t) \leq \Theta, \quad \forall u \in U \tag{2b}$$

$$r_{m,u}(t) \geq h_{m,u}(t) \tag{2c}$$

$$\Omega_m(D_m(t)=0) \geq T_m^{th} \tag{2d}$$

where T_m^{th} denotes the minimum duration to ensure isolation for slice m, $h_{m,u}(t)$ is the amount of the required RBs for user u in slice m at time t, and $r_{m,u}(t)$ is the amount of allocated RBs to the accepted user u in slice m at time t. Specifically, for rate-based slices, $h_{m,u}(t)$ of a user u can vary according to the signal-to-noise ratio (SNR) and the required transmission rate R_m^{th}, which is calculated according to the Shannon–Hartley theorem. Therefore, for rate-based slices, $h_{m,u}(t)$ of a user u at time step t can be approximated as:

$$\begin{aligned} h_{m,u}(t) &= \frac{R_m^{th}}{wrb \times log_2\left(1 + \frac{P_{m,u}(t) \times G_{m,u}^2(t)}{N_0}\right)} \\ &\geqslant \frac{R_m^{th}}{wrb \times log_2\left(1 + \frac{P_{max} \times G_{m,u}^2(t)}{N_0}\right)} \end{aligned} \tag{3}$$

where $G_{m,u}(t)$ represents the time-varying Rayleigh fading channel gain of the transmission, which includes the effects of path loss $L(dB)$ and Rayleigh fading $f_{m,u}(t)$, and equals $G_{m,u}(t) = f_{m,u}(t)/L(dB)$ [31]. $P_{m,u}(t)$ represents the downlink transmission power between a user's device and the base station, N_0 represents the variance of Additive White Gaussian Noise (AWGN) of power and wrb is the bandwidth of the RB. Also, the inequality demonstrates that the minimum number of RBs is needed when the BS transmits with the maximum transmit power.

Problem 3: Given that the power $P_{m,u}(t)$ and the number of required RBs $h_{m,u}(t)$ are fixed to users in resource-based slices, we formulate the resource determination problem to rate-based users as follows, which aims at maximizing the energy efficiency while considering the transmission rate of the user, RBs allocation, and transmission power constraints:

$$\begin{aligned} \arg\max_{N_{m,u}(t), P_{m,u}(t)} \quad & \left(\eta_{EE} = \frac{\eta_{SE}}{P'_{total}}\right. \\ & \left. = \frac{log_2(1 + P_{m,u}(t) \times X_{m,u}(t))}{P_{circuit} + P_{m,u}(t)}\right) \end{aligned} \tag{4a}$$

$$s.t \quad N_{m,u}(t) \times wrb \times log_2(1 + P_{m,u}(t) \times$$

$$X_{m,u}(t)) \geqslant R_m^{th}, \quad \forall m \in M_{II} \tag{4b}$$

$$0 \leqslant P_{m,u}(t) \leqslant P_{max}, \quad \forall m \in M_{II} \tag{4c}$$

$$1 \leqslant N_{m,u}(t) \leqslant N_{r_m}(t) + N_{\Theta_s}(t), \quad \forall m \in M_{II} \tag{4d}$$

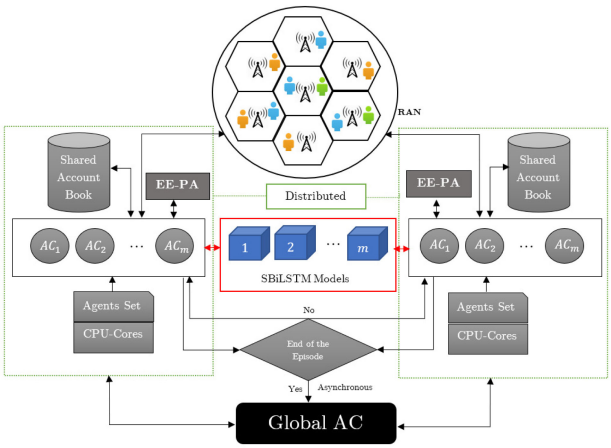

Fig. 3. EE-DRL-RA Components.

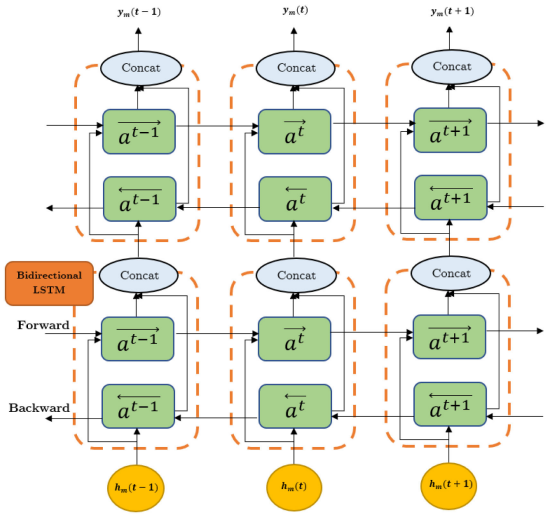

Fig. 4. SBiLSTM Architecture .

where $X_{m,u}(t)$ is equal to $G^2_{m,u}(t)/N_0$, $N_{r_m}(t)$ is the number of remaining allocated RBs to the slice m, $N_{\Theta_s}(t)$ is the number of residual RBs and $P_{circuit}$ is the circuit power. The optimal answer is obtained when we reach the maximum energy efficiency with a minimum of transmission power and number of RBs.

Problems 1, 2 and 3 are a variant of the multiple-choice dimension knapsack problem, which is equivalent to an NP-hard problem [7], therefore, they are not tractable and solvable in a real-time manner. Given that it is infeasible to apply static optimization techniques to solve these problems. Thus, to address Problems 1 and 2, we use machine-learning techniques. Also, we use a gradient descent method to solve Problem 3. Overall, the EE-DRL-RA method is proposed to solve Problems 1, 2, and 3.

V. Proposed Solutions

To address the aforementioned resource allocation problems, we have proposed a distributed method called EE-DRL-RA in which a distributed A3C algorithm and SBiLSTM are used for resource allocation to users on a small time-scale and on a large time-scale, respectively. The EE-DRL-RA architecture consists of the following components (Fig. 3):

- *Parallel A3C blocks (boxes with green dashed lines):* To efficiently explore system state space, multiple parallel A3C blocks (boxes with green dashed lines) are used. Each A3C block involves the parallel execution of ACs from the RAN for the set of slices, which is executed independently of the other A3C blocks. Thus, the global AC algorithm converges faster than when working with only one A3C block.
- *Parallel ACs for the set of slices in each block:* In the A3C algorithm, each slice is considered as an AC that independently decides on resource allocation to its users.
- *Shared account book for each A3C block:* To share the information among slices, each slice writes its own information in the shared account book of its block every time-step. It should be noted that resource allocation information of each slice at every time-step over the PW is stored and used at the end of PW to estimate the allocated resources for each slice in the next PW.
- *EE-PA for determining the required power and RBs to rate-base users:* Given that the required resources for resource-based users is fixed, we propose an iterative method based on gradient descent called EE-PA for determining the required resources to rate-base users.
- *Shared SBiLSTM model for slice m in all A3C blocks:* The EE-DRL-RA method uses a shared SBiLSTM for slice m in all A3C blocks. Therefore, using more input data will improve the accuracy of our LSTM model and results in a faster convergence.
- *Global AC:* To improve decision making in the global AC, each AC updates the global AC information at the end of each episode asynchronously.

A. Large Time-Scale Prediction

In Problem 1 in Section IV, our objective is to find $\bar{r}_m(t)$ for the next PW that minimizes the MSE value for the predicted values at each prediction time using the collected $r_m(t)$ values from the previous and current PWs. To address this issue, we have used Long-Short-Term-Memory (LSTM). LSTM is a type of recurrent neural network that has the ability to learn the dependency of a series of sequential data to predict a time series problem. In EE-DRL-RA, LSTM is used to estimate the volume of required resources for each slice in the next PW on a large time-scale. Compared to method [7], which uses conventional LSTM, we have used the stacked and bidirectional LSTM (SBiLSTM) method in this paper. The stacked LSTM method uses multiple LSTM layers stacked on top of another, which increases the predicted output accuracy [32]. In the bidirectional method, the LSTM model can learn the input sequence both forward and backward, and concatenate both interpretations, achieving accurate predictions [33]. Fig. 4 shows an example of the structure of the layers in stacked and bi-directional LSTM (SBiLSTM) that contains two stacked layers with BiLSTM in three consecutive time steps. As shown, the forward layer output

sequence, $\overrightarrow{a^t}$, is iteratively calculated using inputs in a sequence from time 1 to time $t-1$, while the backward layer output sequence, $\overleftarrow{a^t}$, is calculated using the inputs from the end of the sequence to time $t+1$ [34].

Specifically, the actual required resources ($h_m(t)$) for each slice at each time step from the previous and current PWs are collected as the LSTM input data that is shown by $D_{input,m} = \{h_m(t-2PW), \ldots, h_m(t-PW), \ldots, h_m(t)\}$. The input data is used to predict the allocated resources of the slice m at each time-step in the next PW.

The slice isolation in this scheme depends on the confidence level χ. Let $Y_{predicted,m} = \{y_m(t+1), \ldots, y_m(t+PW)\}$ denote the predicted allocated resources of the slice m in the next PW. The sample mean and standard deviation of $Y_{predicted,m}$ are $\bar{y}_m = \frac{1}{PW}\sum_{k=1}^{PW} y_m(t+k)$ and $\sigma(y_m) = \sqrt{\frac{1}{PW}\sum_{k=1}^{PW}(y_m(t+k)-\bar{y}_m)^2}$, respectively. The confidence level χ has a value between 0 and 1 and can be dynamically adjusted for each slice during the simulation. Therefore, the confidence interval can be calculated using $r_m(t) \in \bar{y}_m \pm z_{\frac{(1-\chi)}{2}} \cdot \frac{\sigma(y_m)}{\sqrt{PW}}, t \in [t, t+T_\triangle]$ that determines the amount of allocated resources ($r_m(t)$) for the slice m in the next PW. Given that the predicted values can be associated with the error, to ensure the quality of service, slice isolation, and traffic service level agreement (SLA), the upper bound value of the prediction interval is used to allocate resources for slice m in the next PW.

B. Small Time-Scale Prediction

In Problem 2in Section IV, our objective is to allocate the minimum $r_{m,u}(t)$ for users in each slice subject to constraints (2b), (2c) and (2d) on a small time-scale. The resource allocation problem is defined as a Markov Decision Process (MDP) and is solved by the A3C algorithm as a deep RL algorithm. It has been reported in many state-of-the-art works that the A3C algorithm outperforms conventional RL methods such as Q-learning, value iteration, etc., especially when state or action space is infinite [8].

The problem is defined as an MDP model with a five-tuple $\{S, A, R, \Pi, \gamma\}$ where S represents the state space to describe the system environment, A represents the action space, R represents the cost function used to measure the quality of the decision, let Π be a set policy, and γ denotes the discount factor. Remark that in the MDP model, the transition probability from one state to another is determined once an action is entertained, which can be denoted as

$$Pr(s'|s,a) = \begin{cases} 1, & if s' = s(t+1) \\ 0, & otherwise \end{cases} \tag{5}$$

The shared state between the slices is defined as $\Phi_s(t) = \{U_{M_I}(t), U_{M_{II}}(t), h_{M_{II}}(t), r_M(t), \Theta(t), P_{total}(t)\}$ at time t and the state space $s_m(t)$ for each slice at time t is represented by a set of $\{m, U_m(t), h_m(t), r_m(t), \Theta(t), P_{total}(t)\}$ where m denotes the slice ID. In this decentralized method, each slice can decide to allocate resources to users and accepting/rejecting new user requests independently of the other slices. For a certain slice m, $a_m(t) \in A_m$ is equal to $\{0, 1\}$, where $a_m(t) = 1$ or $a_m(t) = 0$ represent acceptance or decline of a new incoming request for resource allocation in slice m at time t.

To solve the MDP problem, we have used the A3C method. In A3C, multiple agents are running instead of one, updating the shared global AC periodically and asynchronously. The pseudocode of A3C is shown in Algorithm 1. Algorithm 1 begins with determining global values in Lines 3 - 6. Line 3 determines the number of A3C blocks and slices. In Line 4, the values of E_{max}, T_{max}, and $\Phi_s(t)$ are initialized to determine the maximum number of episodes and the number of time steps per episode, and the shared account book for each block, respectively. Also, the initial weight values for global AC (θ and θ_v) are determined in Line 5. In line 6, a global lock is set to access the global AC to update the values of the weights by each slice. For each slice in each block, counter E counts the number of episodes in Line 8, counter T counts the time steps per episode in Line 9, the initial weight values for the neural network of actor-critic (θ' and θ'_v) in Line 10 and three array-lists $States$, $Actions$, and $Rewards$ to keep the slice information in each episode in Line 11 are defined. In Lines 15 - 20, for a state $s_m(t) \in S$, the agent chooses an action from the set of possible actions A_m conforming to its policy $\pi_m(a_m(t)|s_m(t))$ at time-step t where $\pi_m \in \Pi$. After applying the action to the environment, the agent moves to the next state $s_m(t+1)$ and receives a reward $R_m(t)$. Then, these values are stored in three array-lists $States$, $Actions$, and $Rewards$. In our method, the cumulative reward function is expressed as follows:

$$R_m(t) = |ln(r_m(t) - h_m(t) + 1| + B(D_m(t)) \tag{6}$$

where $|ln(r_m(t) - h_m(t) + 1|$ and $B(D_m(t))$ are considered as the reward functions for constraints (2b) and (2d), respectively. In particular, $B(D_m(t))$ is considered as a bonus function that equals 0 if the slice isolation condition is satisfied, and otherwise is $w(w > 0)$. The value of w is controllable during execution. Clearly, A3C converges when the reward converges to zero. The total accumulated return for slice m at time-step t is $R_m(t) = \sum_{k=0}^{\infty} \gamma^k R_m(t+k)$, where γ is the discount factor between 0 and 1. The goal of the agents is to find the best policies which can minimize the expected cumulative rewards in execution time. In A3C, the actor is the policy $\pi_m(a_m(t)|s_m(t))$ and its task is to produce the best action for a given state $s_m(t)$. The critic is the value function $V_m(t)$, which receives as input the state $s_m(t)$ and the chosen action by the actor, concatenates them, and predicts the action-value for the given pair. Typically, a neural network is used to approximate the actor and the critic functions. Specifically, the advantage term represents the advantage of applying action $a_m(t)$ at state $s_m(t)$ and can be estimated by temporal difference (TD) error as follows:

$$\begin{aligned} &A_m(s_m(t), a_m(t)) \simeq R_m(t) \\ &+ \gamma V_m(s_m(t+1)|s_m(t), a_m(t)) - V_m(s_m(t)) \end{aligned} \tag{7}$$

In Lines 21 - 28, at the end of each episode, each actor and each critic updates the shared global actor-critic using the formulas of Lines 22 - 27 by obtaining the global lock in Line 21. After releasing the global lock in Line 28, at the end of an episode for a slice, the current values of the three array-lists $States$,

Algorithm 1: Energy-Efficient Deep Reinforcement Learning Resource Allocation (EE-DRL-RA) Method for 5G-RAN Slicing.

Input: the number of A3C blocks, the number of slices, γ, E_{max}, T_{max}, Φ_s, θ, θ_v, θ', and θ'_v
Output: θ and θ_v
1 Determine the number of A3C blocks, the number of slices
2 Initialize E_{max}, T_{max}, Φ_s
3 Initialize global shared parameter vectors θ and θ_v
4 Define global thread lock L_{global}
5 **for** *Every slice in each block* **do**
6 Initialize episode counter $E = 0$
7 Initialize thread step counter $T = 1$
8 Initialize thread specific parameter vectors θ' and θ'_v
9 Define 3 empty Array-list: $States$, $Actions$, and $Rewards$
10 **while** $E < E_{max}$ **do**
11 $d\theta = 0$, $d\theta_v = 0$, $\theta' = \theta$ and $\theta'_v = \theta_v$
12 $T = 1$
13 Get state $s_m(t)$
14 **while** $T\%T_{max} == 0$ **do**
15 Simulate action $a_m(t)$ according to $\pi(a_m(t)|s_m(t);\theta)$
16 Receive reward $R_m(t)$, next state $s_m(t+1)$
17 Save $s_m(t)$, $a_m(t)$ and $R_m(t)$ in $States$, $Actions$, and $Rewards$, respectively
18 $T = T + 1$
19 Acquire L_{global}
20 **for** $i = 0$ *to* $len(Rewards)$ **do**
21 $R = R_m(i) + \gamma R$
22 Accumulate gradients for actor: $d\theta = d\theta + \nabla_{\theta'} log\pi(a_m(i)|s_m(i);\theta')(R - V(s_m(i);\theta'_v))$
23 Accumulate gradients for critic: $d\theta_v = d\theta_v + \partial(R - V(s_m(i);\theta'_v))^2)/\partial\theta'_v$
24 $\theta = \theta + d\theta$
25 $\theta_v = \theta + d\theta_v$
26 Release L_{global}
27 Empty $States$, $Actions$, and $Rewards$
28 $E = E + 1$

$Actions$, and $Rewards$ are cleared for the next episode in Line 29, and in Line 30, the next episode begins. For convenience, we summarize the state space, the action space, and the reward function as follows:

- **State space:** $s_m(t) = \{m, U_m(t), h_m(t), r_m(t), \Theta(t), P_{total}(t)\}$
- **Action space:** $a_m(t) = \{0, 1\}$
- **Reward Function:** $R_m(t) = |ln(r_m(t) - h_m(t) + 1| + B(D_m(t))$

C. Energy-Efficient Power Allocation

In Problem 3in Section IV, our objective is to find the maximum energy efficiency (EE) for users in rate-based slices subject to constraints (4b), (4c) and (4d) on a small time-scale. To avoid the high computational complexity of the EE optimization problem, we present a low complexity sub-optimal method called EE-PA. The solution is shown in Algorithm 2. We partition the joint optimization problem into two steps:

1) optimizing the transmission power for each $N_{m,u}$ in $1 \leqslant N_{m,u}(t) \leqslant N_{r_m}(t) + N_{\Theta_s}(t)$
2) selecting the best pair $N_{m,u}$ and $P_{m,u}(t)$ to maximize EE.

To solve (4), we need to find the minimum $N_{m,u}$ in which the following problem is optimal (Lines 6 - 12) :

$$\arg\max_{P_{m,u}(t)} \quad \left(\eta_{EE} = \frac{\eta_{SE}}{P'_{total}} = \frac{log_2(1 + P_{m,u}(t) \times X_{m,u}(t))}{P_{circuit} + P_{m,u}(t)}\right) \tag{8a}$$

$$s.t \quad N_{m,u}(t) \times wrb \times log_2(1 + P_{m,u}(t) \times X_{m,u}(t)) \geqslant R^{th}_m, \quad \forall m \in M_{II} \tag{8b}$$

$$0 \leqslant P_{m,u}(t) \leqslant P_{max}, \quad \forall m \in M_{II} \tag{8c}$$

The above continuous nonlinear optimization problem is non-convex, which is hard to solve with traditional optimization techniques. It is shown that the non-convex optimization problem can be transformed into a convex one by non-linear fractional programming as follows:

$$q^*_s(t) = \ max \quad \frac{\eta_{SE}}{P_{circuit} + P_{m,u}(t)} = \frac{\eta_{SE}(P^*_{m,u}(t))}{P_{circuit} + P^*_{m,u}(t)} \tag{9}$$

Problem (8a) can be transformed into subtraction form by considering Equation (9) as follows:

$$P^*_{m,u}(t) = \text{argmax}_{P_{m,u}(t)} \quad (\eta_{SE} - q_s(t)(P_{circuit} + P_{m,u}(t))) \tag{10}$$

subject to constraints (8b) and (8c).

The Dinkelbach iterative algorithm [30] and [35] is applied to solve (10) by considering a small initial value for $q_s(t)$ (Lines 13 - 31). In the Appendix, it is proved that the above-transformed optimization problem is concave. Therefore, to

solve the energy efficiency maximization problem, the Karush-Kuhn-Tucker conditions are applied to (10). To this end, the Lagrangian of (10) with considering constraints (8b) and (8c) can be written as follows:

$$\begin{aligned} &L(P_{m.i}(t), \lambda_1(t), \lambda_2(t), \lambda_3(t), q_s(t)) = \eta_{SE} \\ &- q_s(t)(P_{circuit} + P_{m,u}(t)) \\ &+ \lambda_1(t)(N_{m,u}(t) \times wrb \times log_2(1 + P_{m,u}(t)X_{m,u}(t)) - R_m^{th}) \\ &+ \lambda_2(t)P_{m,u}(t) - \lambda_3(t)(P_{m,u}(t) - P_{max}) \end{aligned} \tag{11}$$

where $\lambda_1(t)$, $\lambda_2(t)$ and $\lambda_3(t)$ are Lagrange multipliers of constraints on user rate and transmission power, respectively.

The problem (11) can be solved by decomposition into two sub-problems which include [31]:

1) maximizing (11) to gain the optimum transmission power and
2) minimizing the result of the first sub-problem to obtain the optimum Lagrange multipliers and $q_s(t)$. This can be expressed more formally as:

$$\begin{aligned} &\arg\min_{\lambda_1(t),\lambda_2(t),\lambda_3(t),q_s(t)} \quad \arg\max_{P_{m,u}(t)} \quad L(P_{m.i}(t), \\ &\qquad \lambda_1(t), \lambda_2(t), \lambda_3(t), q_s(t)) \end{aligned} \tag{12}$$

The first derivative of (12) with respect to $P_{m,u}(t)$ can be obtained as follows:

$$\begin{aligned} &\frac{\partial L(P_{m.i}(t), \lambda_1(t), \lambda_2(t), \lambda_3(t), q_s(t))}{\partial P_{m,u}(t)} \\ &= \frac{N_{m,u}(t) \times wrb \times X_{m,u}(t)}{ln(2)(1 + P_{m,u}(t)X_{m,u}(t))} \\ &+ \frac{\lambda_1(t) \times N_{m,u}(t) \times wrb \times X_{m,u}(t)}{ln(2)(1 + P_{m,u}(t)X_{m,u}(t))} \\ &- q_s(t) + \lambda_2(t) - \lambda_3(t) \end{aligned} \tag{13}$$

By setting (13) to zero the optimum value of $P_{m,u}(t)$ is found as follows (Line 18):

$$P_{m,u}^*(t) = \left[\frac{N_{m,u}(t) \times wrb \times (1 + \lambda_1(t))}{ln(2)(\lambda_3(t) + q_s(t) - \lambda_2(t))} - \frac{1}{X_{m,u}(t)}\right]^+ \tag{14}$$

where $[x]^+ \triangleq max\{0, x\}$. The Lagrange multipliers can be found by gradient method. In each iteration, the multipliers are updated as follows (Lines 21 - 23):

$$\begin{aligned} \lambda_1^{i+1}(t) = \big[\lambda_1^i(t) - \mu_{\lambda_1}^i(t)(N_{m,u}(t) \times wrb \times \\ log_2(1 + \hat{P}_{m,u}(t)X_{m,u}(t)) - R_m^{th})\big]^+ \end{aligned} \tag{15a}$$

$$\lambda_2^{i+1}(t) = \left[\lambda_2^i(t) - \mu_{\lambda_2}^i(t)\hat{P}_{m,u}(t)\right]^+ \tag{15b}$$

$$\lambda_3^{i+1}(t) = \left[\lambda_3^i(t) - \mu_{\lambda_3}^i(t)(\hat{P}_{m,u}(t) - P_{max})\right]^+ \tag{15c}$$

where $\mu_{\lambda_1}^i(t)$, $\mu_{\lambda_2}^i(t)$ and $\mu_{\lambda_3}^i(t)$ are positive values called the learning rate. The learning rate determines the step size in iteration i to reach an optimum value. The learning rates must be chosen in such a manner that a balance between optimality and convergence speed is obtained, therefore, $\mu^{i+1}(t) = \mu^i(t)/l$ (Lines 24 and 25) [31]. The algorithm continues until the condition $(\eta_{SE} - q_s(t)(P_{circuit} + P_{m,u}(t)) \leqslant \varepsilon)$ is satisfied and it converges to the optimum EE where ε is the maximum tolerance, or the maximum number of iterations (i.e., I_{max}) is reached (Lines 26 - 31).

D. Computational Complexity Analysis

The computational complexity of the EE-DRL-RA method consists of the complexities of A3C, EE-PA, and stacked and bidirectional LSTM methods. In Algorithm 1, each slice is considered as AC and actor-critic are implemented by the neural network. Each AC is executed on a CPU thread. Therefore, parallel processing using CPU threads causes the computational complexity to be divided by the number of CPU threads. Considering the components of A3C in this paper, its computational complexity can be represented as follows [7]:

$$O\left(\left(\frac{M_u}{N_u}\right)T_c\left(\sum_{i=0}^{L_a} un_a^{(i)}un_a^{(i+1)} + \sum_{i=0}^{L_c} un_c^{(i)}un_c^{(i+1)}\right)\right) \tag{16}$$

where M_u, N_u, and T_c denote the total ACs, the number of CPU threads used to train the AC algorithms, and the training steps, respectively. $un_a^{(i)}(un_a^{(j)})$, $un_c^{(i)}(un_c^{(j)})$, L_a, and L_c denote the number of units in the $i^{th}(j^{th})$ layer of the actor and critic networks, the number of layers in the actor network, and the numbers of layer in the critic network, respectively. Given that we have the number of CPU threads (N_u) equal to the number of ACs (M_u), the computational complexity equals $O(T_c(\sum_{i=0}^{L_a} un_a^{(i)}un_a^{(i+1)} + \sum_{i=0}^{L_c} un_c^{(i)}un_c^{(i+1)}))$.

In the stacked and bidirectional LSTM method, learning is done both forward and backward and its computational complexity is twice as much as directional methods. Therefore, the computational complexity of the SBiLSTM [36] per PW for a slice is

$$\begin{aligned} &O(2WN_{ep}PW) \\ &= O(2(4n_cn_c + 4n_in_c + n_cn_o + 3n_c)N_{ep}PW) \\ &= O(WN_{ep}PW) \end{aligned} \tag{17}$$

where PW denotes the prediction window, which includes several time-steps, n_c denotes the number of memory cells, n_i denotes the number of input units, n_o denotes the number of output units, and N_{ep} is the number of epochs.

To find the computational complexity of the EE-PA method in Algorithm 2, two loops are executed for every N with the maximum number of repetitions of I_{max} and L_{max} until we reach the optimal solution, where N is the number of RBs and I_{msx} and L_{max} are the maximum number of iterations for the inner and outer parts of the Lagrange method, respectively [31]. Therefore, the computational complexity of the EE-PA method for a rate-base slice is of order $O(NI_{max}L_{max})$. Given that the stacked and bidirectional LSTM method is used per PW for each slice and the EE-PA method in the worst case is executed

Algorithm 2: Pseudocode of Energy-Efficient Power Allocation (EE-PA) Method For Rate-Based Users.

Input: R_m^{th}, wrb, N, P_{Total}, and P_{max}
Output: N, P, Q
Initialize R_m^{th}, wrb, $N = 1$
Initialize $X_{m,u}(t) \longleftarrow G_{m,u}^2(t)/N_0$
Initialize $P_{max} \longleftarrow \min\{P_{max}, P_{Total}(t)\}$
$P, Q = Solver(N, wrb, R_m^{th}, X_{m,u}(t), P_{max})$
while $P > P_{max} or P < 0$ **do**
  $N \longleftarrow N + 1$
  if $N > N_{r_m}(t) + N_{\Theta_s}(t)$ **then**
    return $NULL$
  $P, Q = Solver(N, wrb, R_m^{th}, X_{m,u}(t), P_{max})$
return N, P, Q
Function `Solver`$(N,\ wrb,\ R_m^{th},\ X_{m,u}(t),\ P_{max})$ **:**
  Initialize $q_s(t)$, ε, I_{max} and L_{max}
  for $i = 1$ *to* I_{max} **do**
    Initialize $\lambda_1(t)$, $\lambda_2(t)$ and $\lambda_3(t)$
    for $l = 1$ *to* L_{max} **do**
      $\hat{P}_{m,u}^l(t) \longleftarrow \left[\frac{(N \times wrb)(1+\lambda_1^l(t))}{ln(2)(\lambda_3^l(t)+q_s(i)-\lambda_2^l(t))} - \frac{1}{X_{m,u}(t)}\right]$
      if $\hat{P}_{m,u}^l(t) < 0$ **then**
        return $\hat{P}_{m,u}^l(t), 0$
      $\lambda_1^{l+1}(t) \longleftarrow \left[\lambda_1^l(t) - \mu_{\lambda_1}^l(t)(N \times wrb \times log_2(1 + \hat{P}_{m,u}^l(t)X_{m,u}(t)) - R_m^{th})\right]$
      $\lambda_2^{l+1}(t) \longleftarrow \left[\lambda_2^l(t) - \mu_{\lambda_2}^l(t)\hat{P}_{m,u}^l(t)\right]$
      $\lambda_3^{l+1}(t) \longleftarrow \left[\lambda_3^l(t) - \mu_{\lambda_3}^l(t)(\hat{P}_{m,u}^l(t) - P_{max})\right]$
      $\mu_{\lambda_2}^{l+1}(t) = \mu_{\lambda_2}^l(t)/l$
      $\mu_{\lambda_3}^{l+1}(t) = \mu_{\lambda_3}^l(t)/l$
    if $(\eta_{SE} - q_s(P_{circuit} + P_{m,u}^l(t)) \leqslant \varepsilon)$ **then**
      $P_{m,u}^*(t) \longleftarrow \hat{P}_{m,u}^l(t)$
      $q_s^*(t) \longleftarrow \eta_{SE}(P_{m,u}^*(t))/P_{circuit} + P_{m,u}^*(t)$
      return $P_{m,u}^*, q_s^*$
    else
      $q_s^{i+1}(t) \longleftarrow \eta_{SE}(\hat{P}_{m,u}^l(t))/P_{circuit} + \hat{P}_{m,u}^l(t)$

in every time step for rate-based users for rate-based slices, the total computational complexity of the EE-DRL-RA method is

$$O\big(T_c\big(\sum_{i=0}^{L_a} un_a^{(i)} un_a^{(i+1)} + \sum_{i=0}^{L_c} un_c^{(i)} un_c^{(i+1)} + WN_{ep} + NI_{max}L_{max}\big)\big) \quad (18)$$

According to (18), due to the parallel and distributed execution of the EE-DRL-RA method, it runs in less time than other resource allocation methods based on DQN, DDQN, SARSA, AC, and A2C and it is scalable in terms of time complexity. Also, compared to most of the studied ML approaches in Section II, parallel computation and a fixed state size are used in the EE-DRL-RA method. Therefore, increasing the number of slices will just increase the dimension size of Φ_s, will not increase the size of actor-critic networks and does not affect the computational complexity of the algorithm.

E. Scalability Assessment and Signaling Overhead

Scalability is an important challenge in network slicing. One of the main goals of the proposed EE-DRL-RA is to solve this problem. Compared to other DRL methods such as AC, A2C, DQN, double DQN, and DQN dueling, which are widely used in network slicing [8] - [29], EE-DRL-RA is based on the A3C method in which each slice is separately executed on a thread (all computations about resource allocation for a single slice are dedicated to a specific thread). Therefore, slice management, including adding/removing/pausing a slice can be done independently of other slices and does not require reconfiguring the network. While in other methods [9] - [29], the learning network must be reconfigured and the learning process must be started from the beginning. Also, in other methods, slices get services sequentially, while in the EE-DRL-RA method, this is done in parallel.

In general, the signaling overhead is a particular concern between entities that may be deployed geographically in different locations. We can distinguish three general deployment scenarios. Our implementation in this paper is of type Case 1.

Case 1) single system/multiple threads: If there are sufficient computational resources, all A3C blocks, SBiLSTM, and EE-PA methods can execute on one system with different CPU threads. In this case, all threads are executed in a single computational unit. Therefore, no signaling message between the slices are communicated through front-haul and back-haul links. Of

course, there is some internal messages exchange between entities (including AC, SBiLSTM, and EE-PA). Each slice records its information in the shared account book as mentioned in the system model, which is accessible to all slices. Simultaneous access of different slices to this shared memory is managed by two semaphores (CPU locks).

Case 2) multiple systems: In this case, the A3C blocks are executed in a distributed manner and access to the shared resources (power and RBs) inside an A3C block is similar to case 1. To access the global AC and the SBiLSTM models, a distributed semaphore can be used to minimize signaling between blocks. In this case, at the end of each PW, the distributed blocks must update weights of the global AC and the SBiLSTM models, and receive outputs of the global AC and the SBiLSTM models, and apply them to their network. These weights should be communicated over front-haul links. Also, the LSTM models can be executed on mobile edge computing (MEC) or the cloud. The signaling overhead, in this case, depends on the architecture of the RAN (e.g., C-RAN, O-RAN, etc.).

Case 3) single system/ single thread: Similar to AC, A2C, DQN, double DQN, and DQN dueling [8] - [29], the A3C method can be run on a single thread. The signaling overhead of this case is similar to that of case 1. When the number of slices/users increases, the scalability of this scenario is lower than the abovementioned cases.

In general, it should be noted that the signaling of the proposed method is similar to iRSS [7]. Also, it is much lower than the signaling overhead of other distributed methods based on deep distributed Q-network (DDQN) [19] and federated learning [24] (which include several entities that are geographically distant from each other), which update the global model and the local model at each time step.

VI. Simulation Results

In this section, we evaluate the performance of the EE-DRL-RA method against the iRSS [7] method by extensive simulations. The reason for choosing the iRSS [7] for comparison is that this method also uses A3C for DRL on a small time-scale and conventional LSTM for DL on a large time-scale, and in this respect, it is close to the EE-DRL-RA method. At the beginning of the learning process, there is insufficient collected information on resource allocation, and EE-DRL-RA performance will improve over time. The proposed EE-DRL-RA method can be used to allocate resources to all traffic models that lead to an MDP system. However, for non-Poisson traffic, the system will be non-MDP for which some techniques (such as State-Process) lead to convergence under some conditions [37], [38]. This case is beyond the scope of our paper and we use the Poisson distribution to generate the users' requests in slice m with arrival and departure rates $\lambda[m]$ and $\mu[m]$, respectively. In these simulations, the number of slices is 10, 4 of which are resource-based and 6 of which are rate-based. As shown in Equation (18), the computational and time complexity of EE-DRL-RA does not depend on the number of slices. However, since we have compared our work with that of [7], for a fair comparison, we use the same number of slices (i.e., 10) as [7].

TABLE II
Simulation Parameters

Parameter	Value
Number of A3C blocks (Boxes with green dashed lines in Fig. 3)	2
Number of slices	10
Number of resource-based slices	4
Number of rate-based slices	6
Learning rate of actor networks ($\alpha_{a,t}$)	10^{-4}
Learning rate of critic networks ($\alpha_{c,t}$)	10^{-4}
Discount factor	0.99
Number of RAN RBs (Θ)	200
Total Power of base station (P_{total})	700 Watts
P_{max}	40 Watts
Prediction Window (PW)	1200s
Confidence level for the LSTM model	97%
Required RBs for resource-based slices	$\{4, 3, 2, 1\}$ RBs
Required rate for rate-based slices	$\{8, 7, 6, 4, 3, 2\}$ Mbps
Minimum threshold of the length of duration to ensure isolation (T_m^{th})	PW/10
$\lambda[M]$	$[0.7, 0.75, 0.6, 0.75, 0.65, 0.75, 0.6, 0.8, 0.7, 0.75]$
$\mu[M]$	$[0.75, 0.85, 0.75, 0.85, 0.75, 0.85, 0.76, 0.85, 0.75, 0.85]$
RB bandwidth	1.8 KHz
Path loss exponent (n)	2
Distance between the transmitter and the receiver (d)	100m
System losses (C)	38.4

Increasing the number of A3C blocks leads to faster convergence of the proposed algorithm but does not affect the results. We consider 2 blocks because our competitive method (i.e., [7]) uses the same number of blocks. However, one can increase the number of blocks if there are enough computational resources available. In other words, the number of blocks only affects the convergence speed and does not affect the numerical results of the paper. We consider the learning rates to be very small to ensure the convergence of the learning algorithms. It should be noted that for any learning algorithm, including our algorithms, if the learning rate is considered very small, the optimal solution will eventually be found, but it convergences slowly. On the other hand, higher learning rates result in faster convergence speed, but the solution may fluctuate. Therefore, a trade-off between convergence and optimality exists. As stated in [7], the convergence condition of the A3C algorithm is fulfilled when the learning rates of the actor and the critic satisfy $\sum_{t=0}^{\infty} \alpha_{a,t} = \infty$, $\sum_{t=0}^{\infty} \alpha_{c,t} = \infty$, $\sum_{t=0}^{\infty} \alpha_{a,t}^2 = \infty$, and $\sum_{t=0}^{\infty} \alpha_{c,t}^2 = \infty$, respectively. For this reason, we use the same learning parameters as [7] for A3C. It should be noted that a PW includes several small time-steps. Specifically, the resource allocation on a large time-scale is performed at the beginning of the next PW, and the resource allocation of a small time-scale is accomplished every time step. In our simulations, the total number of time steps is 48,000 and the PW size equals 1200. Given that the traffic pattern will become clearer over time, we consider the PW size to be large enough so that the SBiLSTM method can more accurately predict $r_m(t)$. The parameter values used in the simulations are listed in Table II.

The simulations were performed in Python using the Keras, Gym, and Tensorflow libraries. We implemented the EE-DRL-RA environment as a new environment in Gym so that we could take advantage of the Gym library. We also used Keras and TensorFlow to implement the LSTM models and A3C. In order to demonstrate the accuracy of the SBiLSTM model and the efficiency of the EE-PA model in the EE-DRL-RA method, we have evaluated the proposed method against both iRSS [7] and iRSS [7] enhanced with the proposed EE-PA method, namely iRSS [7] + EE-PA, in terms of

1) the LSTM accuracy
2) the convergence speed
3) the energy efficiency
4) the number of accepted users
5) the utilization
6) the isolation.

We summarized the definition of the evaluation parameters as follows:

- *LSTM accuracy:* To measure the accuracy of the LSTM models for prediction $r_m(t)$, we consider the error and mean squared error (MSE) between the actual and the estimated values.
- *Convergence speed:* Our DRL model converges when training does not improve the model and the reward value is within a range around the specified value. As mentioned earlier, the proposed DRL model converges when the reward value reaches to zero.
- *Energy efficiency and number of accepted users:* To measure the performance of EE-PA, we consider the total EE of accepted users and the percentage of accepted users per PW. The EE of each accepted user is equal to Equation (8a), and the percentage of accepted users equals the ratio of accepted users to total incoming users per PW.
- *Utilization:* To measure the optimal resource allocation to each slice, given that the amount of $r_m(t)$ is reconfigured at the beginning of each PW, we define the utilization as $\sum_t h_m(t)/\sum_t r_m(t)$ in a PW.
- *Isolation:* To measure the isolation of each slice, we define the isolation degree as the number of time steps that each slice was isolated in each PW.

We evaluate the various pieces of the overall method individually: SBiLSTM, A3C, and EE-PA. Then, we combine them into the overall solution. Given that the LSTM accuracy has a great impact on ensuring the isolation of the slices and the conventional LSTM method is used in iRSS [7], we first evaluate the prediction accuracy of the SBiLSTM method in the EE-DRL-RA method against the conventional LSTM method used in many previous work including [7] on a large time-scale without considering A3C. Fig. 5(a) and (5(b) show the mean square error of SBiLSTM used in the EE-DRL-RA method and the conventional LSTM used in iRSS [7] in terms of the number of epochs. Fig. 5(c) and (d) also show the error between the targets and the outputs of both LSTM models in training data, validation data, and test data, respectively. In this simulation, we use 70 % of the data for training, 15% for validation, and 15% for testing.

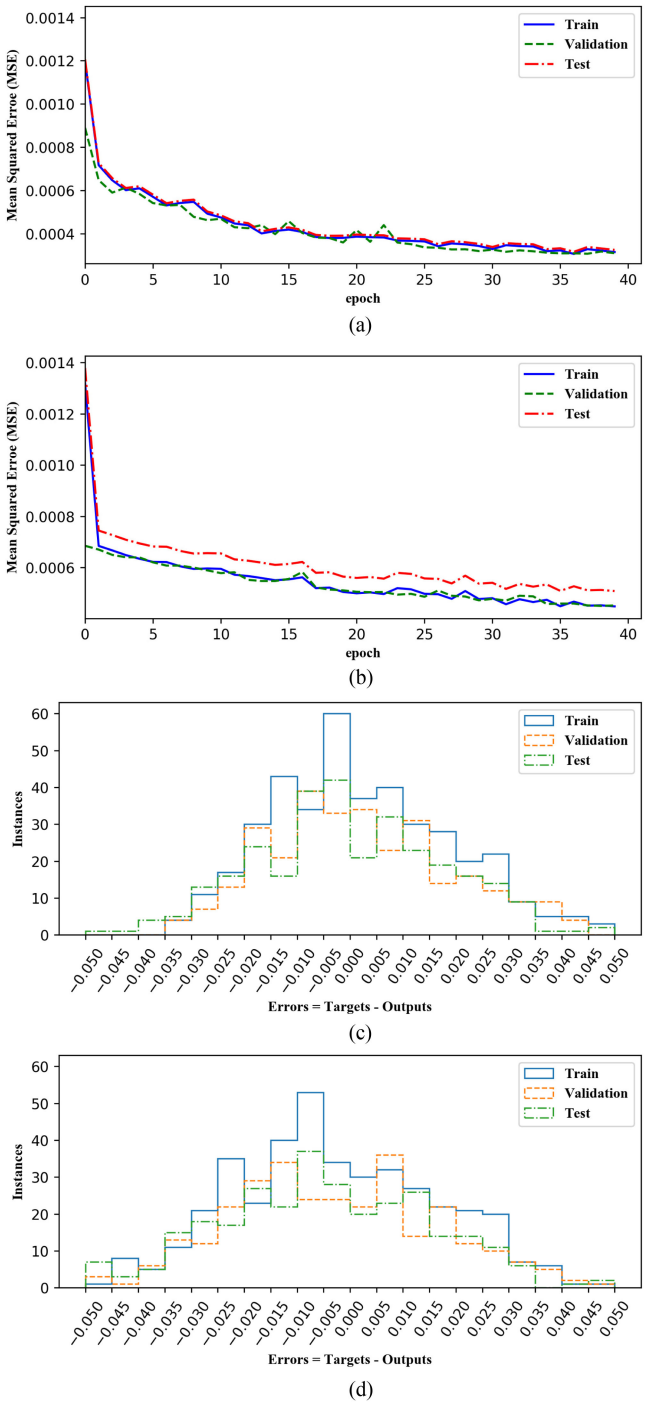

Fig. 5. Performance of the LSTM models in traffic prediction (a) Mean square error of SBiLSTM (b) Mean square error of conventional LSTM (c) Prediction errors histogram between the outputs and targets of SBiLSTM (d) Prediction errors histogram between the outputs and targets of conventional LSTM.

In Fig. 5(a) and (b), the simulation results show that the MSE in both LSTM models can eventually converge to the minimum value within 38 epochs. At the end of the training, the MSE value of test data for the SBiLSTM method and the conventional LSTM method are 0.00032 and 0.00051, respectively. In addition, errors between targets and outputs are shown for

the training data, the validation data, and the test data in the SBiLSTM model and the conventional LSTM model within 38 epochs in Fig. 5(c) and (d). The simulation result indicates that the error frequency between targets and outputs near zero is higher in SBiLSTM, compared to conventional LSTM, hence, SBiLSTM can predict instances more accurately. Also, in the on-line phase over time, the accuracy of SBiLSTM in EE-DRL-RA is better than the conventional LSTM in iRSS [7], which can be concluded by comparing the isolation and utilization diagrams of all three scenarios in Fig. 8.

Fig. 6(a), (b), and (c) show the average cumulative reward for the three methods at each time step. The results show that all three methods converge to 0 over time. The reward of the EE-DRL-RA algorithm converges to the optimal decision very fast, within dozens of learning time steps. Although, at the beginning of the simulation, the reward grows very high taking account into an inexperienced or a failed exploration. In addition, the results of large time-scale prediction in online learning may have unstable performance in the beginning due to the lack of sufficient input data. In the iRSS [7] method, Φ_s is considered to determine the state space in AC, in which more time is required to explore the space states, and the reward converges to zero at a slower rate, while EE-DRL-RA uses a smaller and fixed state space and converges very fast. The optimal policy in EE-DRL-RA is learned faster than the compared methods. Therefore, in iRSS [7] and iRSS [7] + EE-PA, convergence is slower and the reward is associated with penalties due to the use of larger state space and incorrect decisions. Using SBiLSTM in the EE-DRL-RA method has improved the isolation of each slice, causing fewer violations of the isolation of each slice and a lower penalty compared to the other two methods. Also, using SBiLSTM in the EE-DRL-RA has led to a higher prediction accuracy for r_m in each slice, so, this method has a lower average cumulative reward compared to those methods. Fig. 6(d) shows the average cumulative reward per PW. Due to the smaller state space in the EE-DRL-RA method and the use of SBiLSTM, the EE-DRL-RA method has a lower average cumulative reward than the other two methods and converges faster than those methods. In addition, the use of EE-PA in EE-DRL-RA and iRSS [7] + EE-PA has resulted in more resources being available for allocation to users in the slices which leads to accepting more users. By accepting more users, lower penalty values will be applied to the learning agent. However, in iRSS [7] due to lack of resources until it learns the optimal allocation policy, more users are rejected which results in higher penalty values. As a result, the state space in EE-DRL-RA is explored very fast and the optimal policy in EE-DRL-RA is learned faster than the competetive methods. In other words, EE-DRL-RA can provide a feasible online solution of resource allocation for slices within an acceptable time step.

In addition, our solution for the EE-PA problem converges quickly to the optimal. To demonstrate this, we obtain the EE-PA outputs (optimization problem with constraints (8a), (8b), and (8c)) for different rates and compare them with the results of successive linear programming (SLP) in Lingo (optimization problem solver). We show the scalability of the EE-PA method in Fig. 7. In this figure, P^* and EE^* are obtained by SLP, and

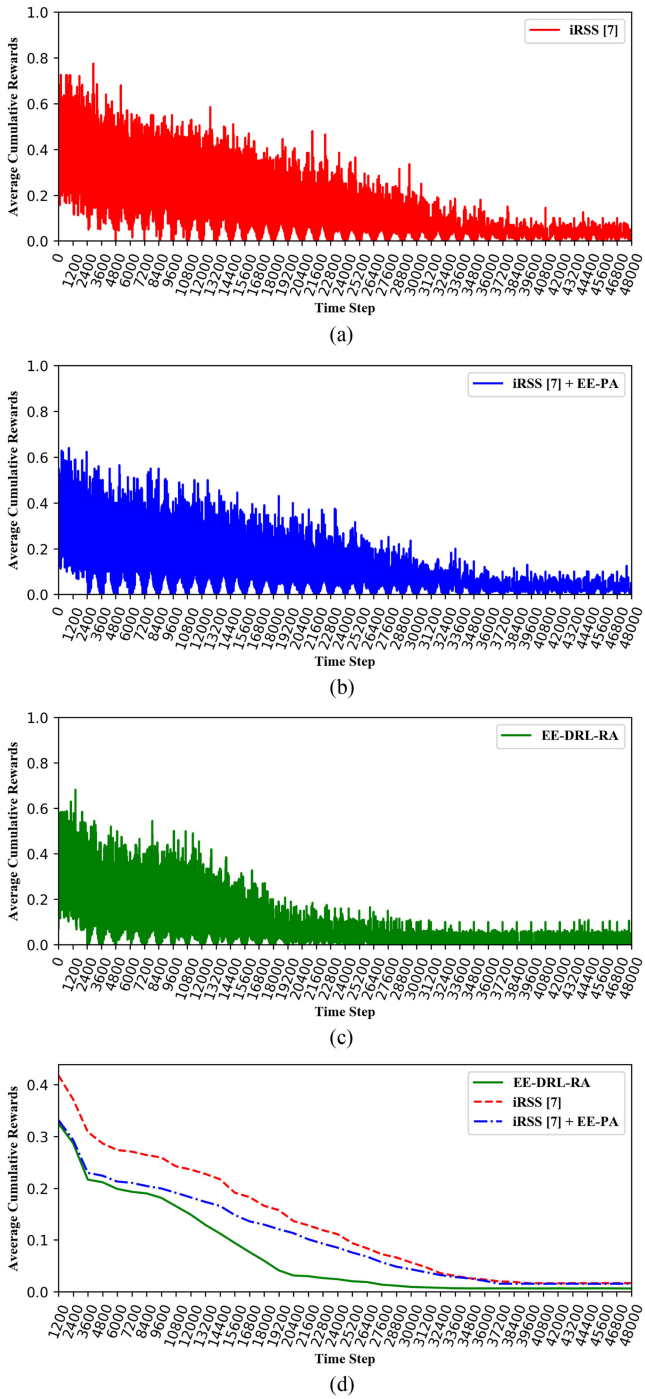

Fig. 6. Convergence of the A3C algorithm (a) Average cumulative reward of iRSS [7] per Time Step (b) Average cumulative reward of iRSS [7] + EE-PA per Time Step (c) Average cumulative reward of EE-DRL-RA per Time Step (d) Average cumulative rewards per PW.

I is the total number of iterations (the outer loop counter) in the EE-PA algorithm. As shown in the figure, the EE-PA method has low computational complexity (i.e., it converges to an optimal solution with a small number of iterations), and in this respect is adaptable to real-life system.

Fig. 8 shows utilization versus isolation degree. In the first two PWs, LSTM is not used because there is insufficient data

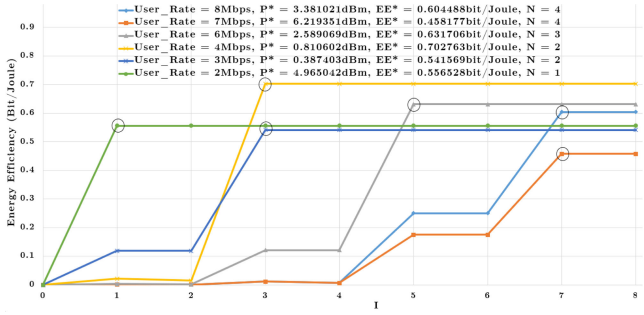

Fig. 7. Energy efficiency versus number of iterations with different transmission rates in EE-PA method.

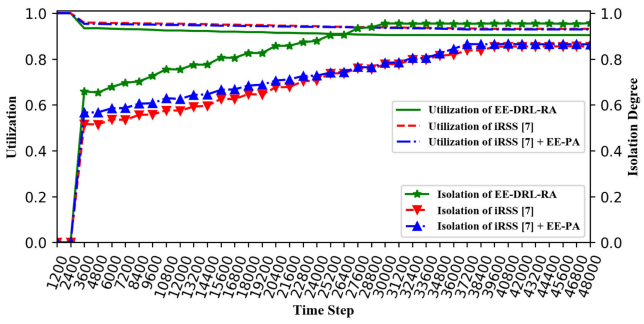

Fig. 8. Utilization vs. Isolation Degree.

for estimating $r_m(t)$. Therefore, $h_m(t)$ as the input data for the LSTM models are collected for each slice at each time step in the first two PWs for prediction of $r_m(t)$. In the first two PWs, the amount of $r_m(t)$ is equal to $h_m(t)$ because the resources are allocated to each slice according to its $h_m(t)$. Hence, the amount of utilization is equal to 1 and the amount of isolation is equal to zero in the first two PWs. Over time and collecting input data for the LSTM models, it can be seen that the isolation for EE-DRL-RA is higher than the other two methods due to the accurate estimation of $r_m(t)$ (predictions are close to the target value) and the fast convergence speed of the SBiLSTM models because of using the shared models between the A3C blocks. The utilization of the iRSS [7] and the iRSS [7] + EE-PA is higher than the utilization of the proposed EE-DRL-RA method because in these two methods, due to the use of the conventional LSTM model, the accuracy of the estimates of $r_m(t)$ is lower and the isolation is violated more often. Therefore, the slices may frequently use the residual resources for serving their users. Consequently, the amount of $r_m(t)$ will be equal to $h_m(t)$ and the amount of utilization for iRSS [7] and iRSS [7] + EE-PA will be higher than EE-DRL-RA. Therefore, the use of the accurate and fast convergence LSTM model has a significant impact on the isolation of slices and the utilization of resources.

In Fig. 9(a) and (b), the effect of the proposed EE-PA method on EE-DRL-RA and iRSS [7] is investigated. In iRSS [7], it is assumed that the amount of allocated power to all users is constant and equal. Fig. 9(a) and (b) show the total energy efficiency and percentage of user acceptance in the three competitive methods, respectively. In EE-DRL-RA and iRSS [7] + EE-PA, the amount of total energy efficiency and percentage of user acceptance is

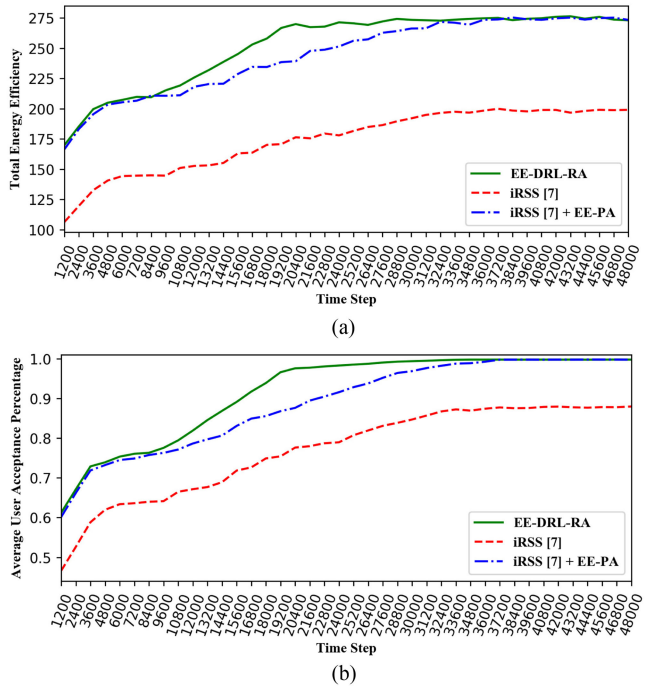

Fig. 9. Effect of the proposed EE-PA method on (a) Total energy efficiency (b) Average user acceptance percentage.

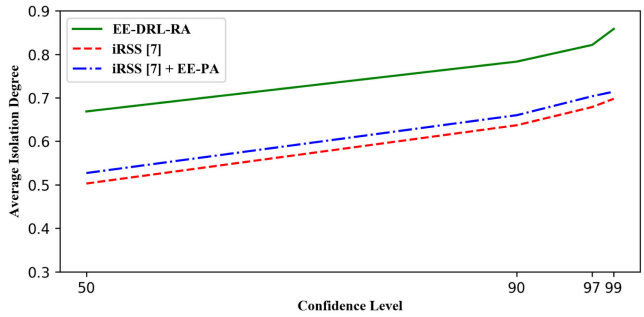

Fig. 10. The relationship between isolation degree and the confidence level in the LSTM model.

higher than iRSS [7] in Fig. (a) and (b), respectively. In the EE-PA method, the optimal answer of $N_{m,u}(t)$ and $P_{m,u}(t)$ is always obtained when the amount of $N_{m,u}(t)$ and $P_{m,u}(t)$ are minimum and the amount of η_{EE} is maximum, therefore, more users receive service due to the increase in available resources. Increasing user acceptance also leads to optimal use of the channel and increases channel efficiency.

Finally, we examine the relationship between the isolation of the slices and the confidence level that is used for determining the upper bound of $r_m(t)$ in the SBiLSTM model. Fig. 10 shows that as the confidence level increases, the amount of pre-assigned resources per slice increases, which guarantees the isolation of each slice. Decreasing the confidence level leads to more frequent isolation violations and resources may be allocated to the slices from residual resources. As shown in the figure, the EE-DRL-RA method has a higher isolation degree than the other two methods and this demonstrates (again) that the prediction

accuracy of SBiLSTM is higher in the proposed EE-DRL-RA method.

VII. Conclusion and Future Work

In this paper, we proposed a resource allocation method, namely EE-DRL-RA, based on deep learning and deep reinforcement learning. We have used the stacked and bidirectional LSTM model as deep learning for decision making on resource allocation on a large time-scale, and A3C with parallel ACs as deep learning for decision making on resource allocation on a small time-scale. Also, we have proposed a method called EE-PA for determining the amount of required resources to rate-based users, which uses an iterative method based on gradient descent. The use of stacked and bidirectional LSTM in the EE-DRL-RA increases the accuracy of estimating the allocated resources in the next PWs and thus satisfies the isolation of each slice more efficiently. EE-PA also decreases the utilized resources and increases the efficiency of the channel, which leads to a higher user acceptance ratio. In general, the use of stacked and bidirectional LSTM, the EE-PA method, and the smaller state space in the EE-DRL-RA method result in better performance of the proposed method compared to other competitive methods in terms of convergence speed, the accuracy of LSTM, energy efficiency, the number of accepted users, and isolation degree.

For future works, the following avenues will be considered: (1) Taking into account the channel interference in resource allocation, (2) Studying priority queues for different slices and incorporating the priority of slices in the resource allocation decisions.

Appendix
Proof of Concavity of η_{EE} Function

In Section V-C, the optimization problem in Equations (8a), (8b), and (8c) is non-convex, thus, we converted the problem into a convex problem using non-linear fractional programming (10). Here, we prove that η_{EE} is a concave function. We rewrite (10)

$$P^*_{m,u}(t) = \text{argmax}_{P_{m,u}(t)} \quad (\eta_{SE} - q_s(t)(P_{circuit} + P_{m,u}(t)))$$

Term $-q_s(t)(P_{circuit} + P_{m,u}(t))$ is a linear function of $P_{m,u}$ and can be ignored [31] and proved that the remainder of the formula is concave. The Hessian matrix (H) of (10) can be written as follows:

$$H = \left[\frac{-X_s^2 ln(2)}{(ln(2)(P_{m,u}X_s))^2}\right] \tag{19}$$

We have to prove that the Hessian matrix is negative semi-definite [31]. The eigenvalue of H can be derived as:

$$\omega = \left[\frac{-X_s^2 ln(2)}{(ln(2)(P_{m,u}X_s))^2}\right] \tag{20}$$

Because the values of channel power gain, transmission power, and noise power are positive, the eigenvalue is negative. Therefore, the Hessian matrix H is negative semi-definite, and the objective function given in (10) is concave.

References

[1] F. Grijpink, A. Ménard, H. Sigurdsson, and N. Vucevic, "The road to 5G: The inevitable growth of infrastructure cost," 2018, Accessed: Mar. 20, 2021. [Online]. Available: http://repositorioiri5g.iri.usp.br/jspui/bitstream/123456789/153/1/The-road-to-5G-The-inevitable-growth-of-infrastructure-cost.pdf

[2] A. Sengupta, A. R. Alvarino, A. Catovic, and L. Casaccia, "Cellular terrestrial broadcast.physical layer evolution from 3gpp release 9 to release 16," *IEEE Trans. Broadcast.*, vol. 66, no. 2, pp. 459–470, 2020.

[3] S. R. Pokhrel, J. Ding, J. Park, O.-S. Park, and J. Choi, "Towards enabling critical MMTC: A review of URLLC within MMTC," *IEEE Access*, vol. 8, pp. 131 796–131 813, 2020.

[4] S. Zhang, Y. Wang, and W. Zhou, "Towards secure 5G networks: A survey," *Comput. Netw.*, vol. 162, 2019, Art. no. 106871.

[5] V. W. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[6] Q. Chen, X. Wang, and Y. Lv, "An overview of 5G network slicing architecture," in *Proc. AIP Conf.*, vol. 1967, no. 1, 2018, Art. no. 0 20004.

[7] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent resource scheduling for 5G radio access network slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7691–7703, Aug. 2019.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[9] V. Sciancalepore, L. Zanzi, X. Costa-Perez, and A. Capone, "Onets: Online network slice broker from theory to practice," 2018, *arXiv:1801.03484*.

[10] H. D. R. Albonda and J. Pérez-Romero, "An efficient RAN slicing strategy for a heterogeneous network with EMBB and v2x services," *IEEE Access*, vol. 7, pp. 44 771–44 782, 2019.

[11] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proc. IEEE Conf. Computer Commun.*, 2017, pp. 1–9.

[12] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE Conf. Computer Commun.*, 2017, pp. 1–9.

[13] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez, "A machine learning approach to 5G infrastructure market optimization," *IEEE Trans. Mobile Comput.*, vol. 19, no. 3, pp. 498–512, Mar. 2020.

[14] M. Yan, G. Feng, J. Zhou, and S. Qin, "Smart multi-rat access based on multiagent reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4539–4551, May 2018.

[15] R. Li *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[16] C. Qi, Y. Hua, R. Li, Z. Zhao, and H. Zhang, "Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing," *IEEE Commun. Lett.*, vol. 23, no. 8, pp. 1337–1341, Aug. 2019.

[17] K. Abbas, M. Afaq, T. Ahmed Khan, A. Rafiq, and W.-C. Song, "Slicing the core network and radio access network domains through intent-based networking for 5G networks," *Electronics*, vol. 9, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/10/1710

[18] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Real-time network slicing with uncertain demand: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[19] Y. Hua, R. Li, Z. Zhao, H. Zhang, and X. Chen, "GAN-based deep distributional reinforcement learning for resource management in network slicing," in *Proc. IEEE Global Commun. Conf.*, 2019, pp. 1–6.

[20] R. Li, C. Wang, Z. Zhao, R. Guo, and H. Zhang, "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility," *IEEE Commun. Lett.*, vol. 24, no. 9, pp. 2005–2009, Sep. 2020.

[21] N. Yarkina, Y. Gaidamaka, L. M. Correia, and K. Samouylov, "An analytical model for 5G network resource sharing with flexible sla-oriented slice isolation," *Mathematics*, vol. 8, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/7/1177

[22] I. Vilà, O. Sallent, A. Umbert, and J. Pérez-Romero, "An analytical model for multi-tenant radio access networks supporting guaranteed bit rate services," *IEEE Access*, vol. 7, pp. 57651–57662, 2019.

[23] Y. Wei, Z. Zhang, F. R. Yu, and Z. Han, "Power allocation in hetnets with hybrid energy supply using actor-critic reinforcement learning," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–5.

[24] M. Yan, B. Chen, G. Feng, and S. Qin, "Federated cooperation and augmentation for power allocation in decentralized wireless networks," *IEEE Access*, vol. 8, pp. 48 088–48 100, 2020.

[25] X. Chen *et al.*, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Oct. 2019.
[26] S. Joseph, R. Misra, and S. Katti, "Towards self-driving radios: Physical-layer control using deep reinforcement learning," in *Proc. 20th Int. Workshop Mobile Comput. Syst. Appl.*, 2019, pp. 69–74.
[27] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi, and D. N. K. Jayakody, "Dynamic virtual resource allocation for 5G and beyond network slicing," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 215–226, 2020, doi: 10.1109/OJVT.2020.2990072.
[28] X. Chen, Y. Tang, M. Zhang, and L. Huang, "Ran slice selection mechanism based on satisfaction degree," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2020, pp. 1–6.
[29] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for EMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4585–4600, Jul. 2021.
[30] D. W. K. Ng, E. S. Lo, and R. Schober, "Energy-efficient resource allocation in multi-cell ofdma systems with limited backhaul capacity," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3618–3631, Oct. 2012.
[31] S. Sobhi-Givi, M. G. Shayesteh, and H. Kalbkhani, "Energy-efficient power allocation and user selection for mmwave-noma transmission in M2M communications underlaying cellular heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9866–9881, Sep. 2020.
[32] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Single layer & multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting," *Procedia Comput. Sci.*, vol. 135, pp. 89–98, 2018.
[33] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *Proc. Int. Conf. Artif. Neural Netw.*, 2005, pp. 799–804.
[34] A. Fares, S.-h. Zhong, and J. Jiang, "Eeg-based image classification via a region-level stacked bi-directional deep learning framework," *BMC Med. Informat. Decis. Mak.*, vol. 19, no. 6, pp. 1–11, 2019.
[35] W. Dinkelbach, "On nonlinear fractional programming," *Manage. Sci.*, vol. 13, no. 7, pp. 492–498, 1967.
[36] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," 2014, Accessed: Mar. 20, 2021. [Online]. Available: https://arxiv.org/abs/1402.1128
[37] S. D. Whitehead and L.-J. Lin, "Reinforcement learning of non-markov decision processes," *Artif. Intell.*, vol. 73, no. 1-2, pp. 271–306, 1995.
[38] S. J. Majeed and M. Hutter, "On q-learning convergence for non-markov decision processes," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2546–2552.

Yaser Azimi received the B.Sc. and M.Sc. degrees in software and network engineering from Urmia University, Urmia, Iran, where he is currently working toward the Ph.D. degree. His research interests include AI-enabled wireless networking, next-generation cellular networks, data analytics, and network security.

Saleh Yousefi received the B.S. and M.S. degrees in computer engineering (in the hardware engineering field) and the Ph.D. degree in computer engineering (in the networking field) from the Iran University of Science and Technology, Tehran, Iran, in 1999, 2002, and 2008, respectively. He is currently an Associate Professor with Computer Engineering Department, Urmia University, Urmia, Iran. His research interests include mobile and wireless networks, machine learning and optimization techniques for computer/telecommunication networks, and vehicular networks.

Hashem Kalbkhani received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Urmia University, Urmia, Iran. He is currently an Assistant Professor with the Faculty of Electrical Engineering, Urmia University of Technology, Urmia, Iran. His research interests include wireless networks, machine learning, and signal processing.

Thomas Kunz (Senior Member, IEEE) received the double honors degree in computer science and business administration and the Dr. Ing. degree in computer science from the Technical University of Darmstadt, Germany, in 1990 and 1994, respectively. He is currently a Professor with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada. He heads the Mobile Computing Group, researching wireless network architectures, network protocols, and middleware layers for innovative wireless applications. He is the author or coauthor of more than 70 journal and 190 conference papers and was the recipient of a number of awards and best paper prizes. Dr. Kunz is a Senior Member of the Association for Computing Machinery (ACM).