A Deep Reinforcement Learning based Framework for Power-Efficient Resource Allocation in Cloud RANs

Zhiyuan Xu, Yanzhi Wang, Jian Tang, Jing Wang, and Mustafa Cenk Gursoy

***Abstract*—Cloud Radio Access Networks (RANs) have become a key enabling technique for the next generation (5G) wireless communications, which can meet requirements of massively growing wireless data traffic. However, resource allocation in cloud RANs still needs to be further improved in order to reach the objective of minimizing power consumption and meeting demands of wireless users over a long operational period. Inspired by the success of Deep Reinforcement Learning (DRL) on solving complicated control problems, we present a novel DRL-based framework for power-efficient resource allocation in cloud RANs. Specifically, we define the state space, action space and reward function for the DRL agent, apply a Deep Neural Network (DNN) to approximate the action-value function, and formally formulate the resource allocation problem (in each decision epoch) as a convex optimization problem. We evaluate the performance of the proposed framework by comparing it with two widely-used baselines via simulation. The simulation results show it can achieve significant power savings while meeting user demands, and it can well handle highly dynamic cases.**

***Index Terms*–Deep Reinforcement Learning, Resource Allocation, Cloud Radio Access Network, Green Communications**

I. Introduction

Cloud Radio Access Networks (RANs) [5] have become a key enabling technology for the next generation (5G) wireless communications. Unlike traditional RANs where BaseBand Units (BBUs) and radios are situated together, a cloud RAN is designed with separate BBUs and remote radios. All DSP processors are moved into a central BBU pool in the cloud, and the distributed *Remote Radio Heads* (RRHs) take the responsibility of compressing and forwarding the received radio signals from wireless users to BBUs through fronthaul links. In a cloud RAN, a RRH only needs to maintain some basic transmission functions, which significantly reduces the design and operational costs, and makes large-scale high-density network deployments possible. Moreover, this centralized architecture makes it easy to collect and analyze statistics data of the runtime system.

Reinforcement learning (RL) [19] has been shown to be one of effective solutions for resource allocation in communication and computing systems. Through the immediate reward feedback from interactions with the environment, the RL agent can generate (near-)optimal control actions accordingly. Instead of simply optimizing the current reward in a greedy manner, the RL agent takes a long-term goal into account, which is essentially important to time-variant dynamic systems, such as wireless networks and cloud computing systems. The emerging *Deep Reinforcement Learning* (DRL) [15] can be considered as an enhanced version of traditional RL, which provides a promising technique in handling complicated control problems. The DRL technique consists of an offline Deep Neural Network (DNN) [10] construction phase, which correlates the value function with corresponding states and actions, and an online dynamic deep Q-learning phase for action selection, system control, and dynamic network updating. Under a DRL-based control framework, decisions are made under the guidance of a powerful DNN that provides accurate estimation and prediction, which usually leads to an overall good solution for the whole operational periods [15], [18]. In addition, DRL can be used to account for state transition overheads (e.g., the power consumption involved in a transition from sleep/active to active/sleep modes), which are quite significant [24], [25], but are neglected by most resource allocation algorithms that only give optimal/sub-optimal solutions for the current timeslot (or time frame).

However, an important challenge needs to be overcomed when applying DRL to solving the dynamic resource allocation problem in a cloud RAN: The online procedure of DRL requires to derive a desirable action in each decision epoch by exploring all possible actions, and find the best action with maximal (or minimal) state-action Q value estimated from the DNN, and thus the action set is required to be *enumerable*. In order to reduce the size of the action space in the DRL-based framework, we innovatively proposed a two-step decision framework in each decision epoch: The DRL agent first determines the set of RRHs (numbers and IDs) for turning on (or into sleep) to limit the action space size. Subsequently, the DRL agent can derive an optimal resource allocation solution based on a given set of active RRHs by solving a convex optimization problem. Through the combination of DRL and optimization, the available actions become enumerable.

In this paper, we present a DRL-based framework for dynamic resource allocation in a cloud RAN, which minimizes total power consumption while ensuring that the demand of each wireless user is satisfied. Our main contributions are summarized in the following:

- We, for the first time, present a DRL-based framework for dynamic resource allocation in a cloud RAN.
- We define the action space, state space and reward function for the DRL agent, formally formulate the resource

Zhiyuan Xu, Yanzhi Wang, Jian Tang, Jing Wang and Mustafa Cenk Gursoyare with the Department of Electrical Engineering and Computer Science at Syracuse University, Syracuse, NY, 13244, USA. Email: {zxu105, ywang393, jtang02, jwang93, mcgursoy}@syr.edu. This work was supported by NSF grant CNS-1443966. The information reported here does not reflect the position or the policy of the federal government.

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

allocation problem (in each decision epoch) for a given set of active RRHs as a convex optimization problem, and apply a DNN to approximate the action-value function for action decisions, which directly extracts information from the current state and avoids using hand-crafted features.

- Simulation results well justify the effectiveness of the proposed DRL-based framework on power efficiency, user demand satisfaction and the capability of handling highly dynamic cases.

II. Related work

Resource allocation in cloud RANs has been studied in the past few years. In a pioneering work, Bhaumik *et al.* [4] proposed CloudIQ, a resource allocation framework which (1) partitions the set of Base Stations (BSs) into groups that are simultaneously processed on a shared homogeneous compute platform for a given statistical guarantee, and (2) schedules the set of BSs allocated to a platform in order to meet their real-time processing requirements. It has been shown the framework saves up to 19% of compute resources for a probability of failure of one in 100 million. In [20], Sundaresan *et al.* proposed another resource allocation framework, FluidNet, to determine configurations that maximize the total traffic demand satisfied and simultaneously optimizes the compute resource usage in the BBU pool. Shi *et al.* [17] presented a framework to design an energy-efficient cloud RAN, which is formulated as a joint RRH selection and power minimization beamforming problem, and solved by a group sparse beamforming method. Dai *et al.* [6] considered a non-convex optimization problems of minimizing the total network power consumption subject to user target rate constraints. They utilized the techniques of re-weighted ℓ_1 minimization and successive convex approximation to devise provably convergent algorithms. Similar problems have also been studied in [13], [16], [21]. Unlike these related works that presented algorithms optimizing a certain objective (such as power consumption) for the current timeslot (or time frame), we present a DRL-based framework which makes a sequence of resource allocation decisions to minimize total energy consumption for the whole operational period. Furthermore, we also take transition power into consideration, which is significant [24], [25] but ignored by most related works.

DRL was originally developed by DeepMind and has attracted extensive attention from both industry and academia recently. In a pioneering work [15], Mnih *et al.* proposed Deep Q-Network (DQN), which can learn successful policies directly from high-dimensional sensory inputs. This work bridges the gap between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging gaming tasks. The authors of [11] proposed Double Q-learning as a specific adaptation to the DQN, which is introduced in a tabular setting and can be generalized to work with large-scale function approximations. The paper [8] considered a problem of multiple agents sensing and acting in environments with the goal of maximizing their shared utility, and presented two approaches: Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). In order to further extend DRL to the continuous action domain and large discrete action domain, Duan *et al*. [7] presented a benchmark suite of control tasks to quantify progress in the domain of continuous control; and Lillicrap *et al*. [12] proposed an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. Other works related to this topic include [2], [9]. In this paper, we, for the first time, apply DRL to dynamic resource allocation in cloud RANs.

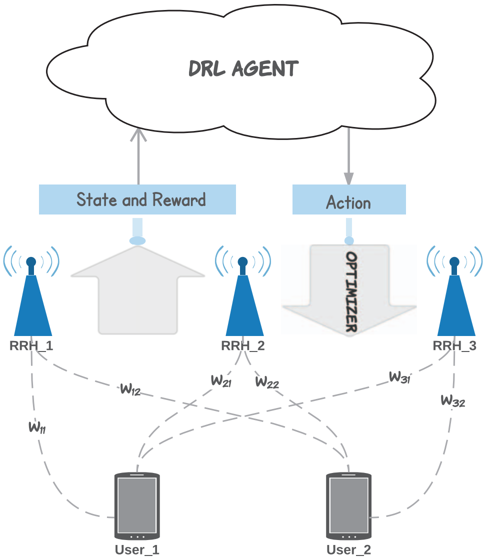


Fig. 1. DRL-based dynamic resource allocation in a cloud RAN

III. System Model

We consider a single cell in a typical cloud RAN, which consists of a set of R RRHs $\mathcal{R} = \{1, 2, \cdots, R\}$ and a set of U users $\mathcal{U} = \{1, 2, \cdots, U\}$. As shown in Fig. 1, in each decision epoch t_k, the network reports the demands of associated users and the current states (active, sleep, etc) of all RRHs to the DRL agent. Moreover, according to the result of the last executed action, the reward can be calculated by the DRL agent (which is explained later). Here, we assume that all RRHs and users are equipped with a single antenna. The proposed solution can be easily generalized to the multi-antenna case as shown in [6]. All RRHs are connected to a cloud BBU pool, thus the information of RRHs can be shared in a centralized manner, and all users can access all RRHs in the cell. The DRL agent will make an action decision $\boldsymbol{a}_k$ based on the system state $\boldsymbol{s}_k$, turn on or into sleep certain RRH(s), and allocate a beamforming weight $w_{r,u}$ from every RRH r to each user u. The major notations are summarized in Table I.

According to the antenna beamforming model [6], the corresponding Signal-to-Interference-plus-Noise Ratio (SINR) at the receiver of user u can be given as:

$$SINR_u = \frac{|\boldsymbol{h}_u^H \boldsymbol{w}_u|^2}{\sum_{v \neq u} |\boldsymbol{h}_u^H \boldsymbol{w}_v|^2 + \sigma^2}, \ u \in \mathcal{U}; \tag{3.1}$$

where $\boldsymbol{h}_u = [h_{1u}, h_{2u}, ..., h_{Ru}]^T$ and each element h_{ru} is the channel gain from RRH r to user u; similarly $\boldsymbol{w}_u =$

TABLE I
MAJOR NOTATIONS

Notation	Description
r and R	The index and the total number of RRHs
$\mathcal{R}$	The set of RRHs
$\mathcal{A}$	The set of active RRHs
$\mathcal{S}$	The set of sleep RRHs
$\mathcal{G}$	The set of transition RRHs
u and U	The index and the total number of users
$\mathcal{U}$	The set of users
$P_{r,sleep}$	Power consumption of RRH r in sleep mode
$P_{r,active}$	Power consumption of RRH r in active mode
$P_{r,transition}$	Power consumption of RRH r during transition
$w_{r,u}$	The beamforming weight from RRH r to user u
$h_{r,u}$	The channel gain from RRH r to user u
t_k	The k-th decision epoch
$\boldsymbol{s}_k$	The state in epoch t_k
$\boldsymbol{a}_k$	The action in epoch t_k
r_k	The reward in epoch t_k

$[w_{1u}, w_{2u}, ..., w_{Ru}]^T$ and each element w_{ru} is the beamforming weight from RRH r to user u. Both of them can be complex numbers. σ^2 is the background noise. Applying the Shannon Hartley–theorem [22], the achievable data rate for user u is given by:

$$R_u = B\log_2(1+\frac{SINR_u}{\Gamma_m}),\ u \in \mathcal{U}; \tag{3.2}$$

where B is the channel bandwidth and Γ_m is the SNR gap that depends on the modulation scheme.

In a cellular network, as shown in [3], the relationship between transmit power and BS power consumption is nearly a linear function, which is also applicable to RRHs. Therefore, we adopt the empirical linear power modeling for each RRH:

$$P_r = \begin{cases} P_{r,active} + \frac{1}{\eta_l}P_{r,trans} & r \in \mathcal{A}; \\ P_{r,sleep} & r \in \mathcal{S}; \end{cases} \tag{3.3}$$

where $P_{r,trans}$ is the transmit power of RRH r, satisfying $P_{r,trans} = \sum_{r\in\mathcal{A}}\sum_{u\in\mathcal{U}}|w_{r,u}|^2$. η_l is a constant and stands for the drain efficiency of the power amplifier. $P_{r,active}$ is the power consumption of the RRH in the active mode, which is necessary in order to maintain the basic operation of the RRH. If the RRH is not selected for transmission, it will enter the sleep mode, and the power consumed by the RRH in this case is $P_{r,sleep}$. $\mathcal{A} \subseteq \mathcal{R}$ and $\mathcal{S} \subseteq \mathcal{R}$ stand for the sets of active and sleep RRHs, respectively. We have $\mathcal{A} \cup \mathcal{S} = \mathcal{R}$.

Besides the above power consumption components, we also account for the transition power (from sleep/active to active/sleep modes). Let $\mathcal{G}$ denote the set of RRHs in the mode transition in the current epoch, which is determined by the DRL agent. Most related works neglect the transition power consumption. However, as shown in [24], [25], it is significant and is thus essential to be considered in a power minimization framework. Based on the above analysis, the total power consumption of RRHs in the current epoch can be expressed as:

$$\begin{aligned} \mathcal{P}(\mathcal{A},\mathcal{S},\mathcal{G}) = &\sum_{r\in\mathcal{A}}\sum_{u\in\mathcal{U}}\frac{1}{\eta_l}|w_{r,u}|^2 \\ &+\sum_{r\in\mathcal{A}}P_{r,active}+\sum_{r\in\mathcal{S}}P_{r,sleep} \\ &+\sum_{r\in\mathcal{G}}P_{r,transition} \end{aligned} \tag{3.4}$$

It is worth mentioning that the proposed DRL-based framework is flexible, which can be applied to other network and power consumption models. Due to the power and timing overheads associated with mode transitions, the cloud RAN control should dynamically optimize the total expected and cumulative power consumption during the whole operational period, instead of optimizing the instantaneous power consumption in each decision epoch. Optimizing the instantaneous power consumption only may result in different sets of turned-on RRHs in nearby decision epochs under time-varying user demands, thereby resulting in high power and timing overheads. The dynamic resource allocation problem in a cloud RAN exhibits high dimension in the state space (user demands and channel gains) and a complicated action space (set of active RRHs and beamforming weights). Both features motivate the adoption of DRL for solving this problem, which will be described in details in the next section.

IV. DRL-BASED RESOURCE ALLOCATION FRAMEWORK

In this section, we present the DRL-based dynamic resource allocation framework, which minimizes total power consumption while ensuring that the demand of each user is satisfied. In order to reduce the action space size in the framework, we innovatively propose a two-step method in each decision epoch. Specifically, the DRL agent first determines the set of RRHs to turning on (or into sleep), which can limit the corresponding action space size. Subsequently, the DRL agent derives an optimal resource allocation (beamforming) solution based on the given set of active RRHs by solving a convex optimization problem.

We first present a generalized form of DRL. DRL comprises an offline DNN construction phase and an online deep Q-learning phase [15], [18]. The offline phase adopts a DNN to derive the correlation between each state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ of the system under control and its value function $Q(\boldsymbol{s}, \boldsymbol{a})$, which is the expected cumulative (with discounts) reward when the system starts at state $\boldsymbol{s}$ and follows action $\boldsymbol{a}$ (and a certain policy thereafter). $Q(\boldsymbol{s}, \boldsymbol{a})$ is given as:

$$Q(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}\big[\sum_{k=0}^{\infty}\mu^k r_k(\boldsymbol{s}_k,\boldsymbol{a}_k)|\boldsymbol{s}_0=\boldsymbol{s},\boldsymbol{a}_0=\boldsymbol{a}\big], \tag{4.1}$$

where $r_k(\cdot)$ gives the reward achieved in decision epoch t_k, and $\mu \in (0, 1]$ is the discount factor which reflects the tradeoff between the immediate and future rewards.

The offline DNN construction needs to accumulate enough samples of value estimates and the corresponding $(\boldsymbol{s}, \boldsymbol{a})$ for constructing a sufficiently accurate DNN. For example, in game-playing applications [15], this procedure includes pre-processing game playing profiles/replays and obtaining the state transition profiles (i.e., a set of state transitions) and Q-value estimates (e.g., win/lose or the score achieved). The use of experience memory [15] in this offline procedure can smooth out learning and avoid oscillations or divergence.

Moreover, the deep Q-learning is adopted for the online dynamic control based on the offline-built DNN. More specifically, in each decision epoch t_k, the DRL agent derives the estimated Q value from the DNN with the input of state-action pair $(\boldsymbol{s}_k, \boldsymbol{a_k})$ for each $\boldsymbol{a_k}$. Then we apply the ε-greedy policy to select the execution action $\boldsymbol{a}_k$, which means with $(1-\varepsilon)$ probability, we choose the action with the highest estimated Q value, or we randomly choose an action from the action set with probability ε. After observing the immediate reward r_k and next state $\boldsymbol{s}_{k+1}$ from the cloud RAN, the state transition $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ will be stored into the experience memory $\mathcal{D}$. At the end of the decision epoch, the DRL agent updates parameters θ of the DNN with $N_{\mathcal{B}}$ samples from the experience memory $\mathcal{D}$. Note that such updating can also be done every T epoches ($T > 1$) to reduce the time complexity. In our implementation, for the DNN construction, we used a feed-forward neural network [10] that has two hidden layers of fully-connected softplus units with 64 and 32 neurons, respectively. In order to train the DNN, we used the same training method as [15]. Here, we set the capacity of experience memory $N_{\mathcal{D}} = 500$, and the capacity of mini batch $N_{\mathcal{B}} = 32$. For Equation (4.1), the reward discount $\mu = 0.9$. It can be observed from the above procedure that the DRL-based framework is scalable for a larger state space and could deal with the case of continuous state space, which is distinct from traditional RL [19] and is suitable for dynamic resource allocation in wireless networks.

For our resource allocation problem in a cloud RAN, we need to compress the action space such that it becomes suitable for DRL. In the first step, the DRL agent determines the set of RRHs for turning on (or into sleep), to limit the action space size. We define the state space, action space and reward function of the DRL-based framework as follows:

State Space: The state of DRL agent consists of two components, the binary (active or sleep) state $M_r \in 0, 1$ of each RRH r and the demand $D_u \in [D_{min}, D_{max}]$ of each user u. The state vector is represented as $[M_1, M_2, ..., M_R, D_1, D_2, ..., D_U]$ with a cardinality of $(R + U)$. Note that in practice, we apply normalization first to each D_u in order to get a better representation of state.

Action Space: In order to limit the corresponding action space size, we restrict the DRL agent to make the decision on a single RRH in each decision epoch. Specifically, the DRL agent determines which RRH to turn on or into sleep. After executing an action, the DRL agent can derive an active set of RRHs ($\mathcal{A}$). Note that the proposed framework is not restricted to this method, i.e., the other methods involving multiple RRHs in each decision epoch can also be applied here, which, however, lead to a larger action space.

Reward: The reward needs to represent the objective of the framework, which is simultaneously minimizing the network power consumption and satisfying user demands. Besides, we also need to account for the transition power consumption mentioned above. Overall, we define the immediate reward that the DRL agent receives as $P_{max} - \mathcal{P}(\mathcal{A}, \mathcal{S}, \mathcal{G})$, where P_{max} is the maximum possible value of total power consumption and $\mathcal{P}(\mathcal{A}, \mathcal{S}, \mathcal{G})$ gives the actual total power consumption that can be calculated by Equation (3.4).

Algorithm 1 The DRL-based Framework

```
Offline:
  Load the historical state transition profiles and the corresponding estimated $Q(\boldsymbol{s}, \boldsymbol{a})$ into experience memory $\mathcal{D}$ with a capacity of $N_{\mathcal{D}}$;
  Pre-train the DNN with input pairs $(\boldsymbol{s}, \boldsymbol{a})$ and the corresponding estimated $Q(\boldsymbol{s}, \boldsymbol{a})$;
Online:
  for each decision epoch $t_k$ do
      /**Operation**/
      With probability $\epsilon$, randomly select an action;
      otherwise $\boldsymbol{a}_k = \mathrm{argmax}_{\boldsymbol{a}}\, Q(\boldsymbol{s}_k, \boldsymbol{a}_k; \theta)$,
      where $Q(\cdot)$ is estimated by the DNN;
      Execute action $\boldsymbol{a}_k$;
      Derive the set of active RRHs $\mathcal{A}$;
      /**Optimization**/
      Obtain the optimal beamforming solution based on the given $\mathcal{A}$ by solving CP-beamforming;
      Reconfigure the network with the beamforming solution $\{w_{r,u}\}$;
      /**Updating**/
      Observe the reward $r_k$ and the new state $\boldsymbol{s}_{k+1}$;
      Store the state transition $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ into $\mathcal{D}$;
      Randomly sample a minibatch of state transitions;
      $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ with a size of $N_{\mathcal{B}}$;
      Target $y_i = r_i + \mu \max_{\boldsymbol{a}'} Q(\boldsymbol{s}_{i+1}, \boldsymbol{a}'; \theta)$;
      Update the DNN's weights with a loss function of
      $(y_i - Q(\boldsymbol{s}_{i+1}, \boldsymbol{a}'; \theta))^2$;
  endfor
```

Based on the given set of active RRHs ($\mathcal{A}$) determined in the first step, the DRL agent further derives an optimal beamforming solution by solving the following optimization problem:

CP-Beamforming:

$$\underset{\{w_{r,u}\}}{\text{minimize}} \quad \sum_{r \in \mathcal{A}} \sum_{u \in \mathcal{U}} |w_{r,u}|^2 \tag{4.2a}$$

$$\text{subject to} \quad SINR_u \geq \gamma_u, \ u \in \mathcal{U}; \tag{4.2b}$$

$$\sum_{u \in \mathcal{U}} |w_{r,u}|^2 \leq P_r, \ r \in \mathcal{A}; \tag{4.2c}$$

$$\text{where} \quad \gamma_u = \Gamma_m (2^{R_u/B} - 1); \tag{4.2d}$$

where the decision variables $\{w_{r,u}\}$ are beamforming weights, P_r is the the maximum allowable transmit power of RRH r, and R_u is the demand of user u. Constraints (4.2b) and (4.2d) ensure that the demand of each user u is satisfied. Constraints (4.2c) make sure of the transmit power limitation at each RRH r. The objective is to minimize the total power consumption. Note that here we omit the factor η_l and some other related parameters in the power consumption model (Equation (3.4)), because they are simply constants. Furthermore, the inequality

(4.2b) can be re-written as:

$$\sum_{v\in\mathcal{U}} |\boldsymbol{h}_u^H \boldsymbol{w}_v|^2 + \sigma^2 \le \frac{1+\gamma_u}{\gamma_u}|\boldsymbol{h}_u^H \boldsymbol{w}_u|^2. \tag{4.3}$$

Given the demand R_u for each user u, γ_u can be calculated accordingly by Equation (4.2b). As implied by [23], this problem can be transformed into a second order cone optimization problem, which is a convex optimization problem. It is known the convex optimization problem can be efficiently solved using an existing method [14]. Note that if the DRL agent is unable to find a feasible solution based on current set of active RRHs, it means more RRHs must be activated in order to satisfy user demands. We need to avoid this infeasible situation if at all possible, thus we give zero reward to the DRL agent for this case.

By utilizing the two-step decision framework, the proposed DRL-based framework exhibits a relatively small action space and low online computational complexity. It is formally presented as Algorithm 1.

V. Performance Evaluation

In this section, we present simulation settings and results of the proposed DRL-based framework. We implemented it using Tensorflow 0.10.0rc0 [1]. For different simulation scenarios, we trained DRL agents with different initial parameters.

For wireless networking, we considered the following channel model [17],

$$h_{r,u} = 10^{-L(d_{r,u})/20}\sqrt{\varphi_{r,u}s_{r,u}}\mathcal{G}_{r,u}, \tag{5.1}$$

where $L(d_{r,u})$ is the path loss with a distance of $d_{r,u}$, $\varphi_{r,u}$ is the antenna gain, $s_{r,u}$ is the shadowing coefficient and $\mathcal{G}_{r,u}$ is the small-scale fading coefficient. The common simulation settings are summarized in Table II.

TABLE II
Simulation Settings

Parameter	**Value**
Channel bandwidth B	10 MHz
Max transmit power $P_{r,\max}$	1.0 W
Active power $P_{r,active}$	6.8 W
Sleep power $P_{r,sleep}$	4.3 W
Transition power $P_{r,transition}$	3.0 W
Background noise σ^2	-102 dBm
Antenna gain $\varphi_{r,u}$	9 dBi
Log-normal shadowing $s_{r,u}$	8 dB
Rayleigh small-scale fading $\mathcal{G}_{r,u}$	$\mathcal{CN}(0,I)$
Path loss with a distance of $d_{r,u}$ (km)	$148.1+37.6\log_2 d_{r,u}$ dB
Distance $d_{r,u}$	Uniformly distributed in [0,800]m
Power amplifier efficiency η_l	25%

We compared the proposed DRL-based framework (labeled as "DRL") with two baseline solutions [6]: 1) *Single BS Association (SA)*: every user is associated only with a randomly chosen RRH and the set of RRHs without any association are turned into sleep. 2) *Full Coordinated Association (FA)*: all RRHs are turned on and a user can be associated with multiple RRHs. Note that for both baselines, the above convex optimization problem was solved to minimize power consumption for active RRHs for fair comparisons.

First, we evaluated the performance of the proposed framework with static user demand scenarios. We changed the demand of each user from 10Mbps to 60Mbps with a step size of 10Mbps. We had 6 RRHs, and 3 and 4 users in scenarios 1 and 2 respectively. We increased the number of RRHs and users to 8 and 4 respectively in scenario 3. In these scenarios, we counted the power consumption of the final stablized solution given by the proposed framework. The corresponding results are presented in Fig. 2, from which, we can make the following observations. Note that for simplicity, *we call the three methods, SA, FA and DRL respectively in the following.*

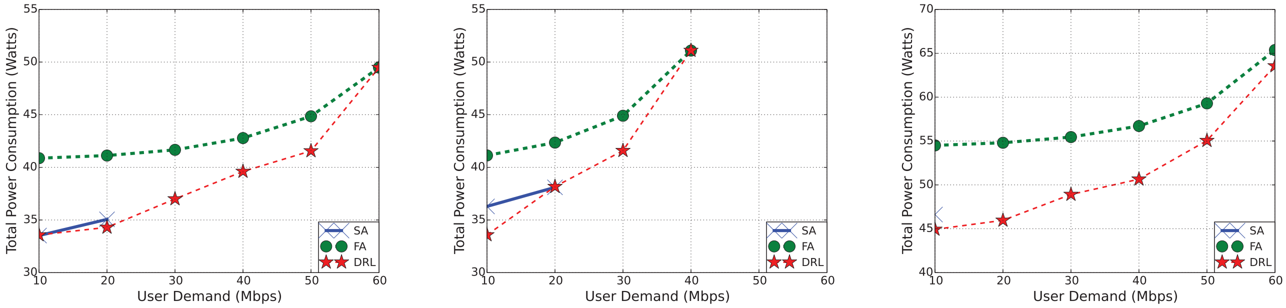
1) SA suffers from a serious problem of satisfying user demands. When the user demand gets higher, e.g., higher than 20Mbps, SA fails to find a feasible solution. FA turns on all the RRHs and solves the convex optimization problem for resource allocation so it can always find a feasible solution (i.e., meeting all user demands) if there exists one. We can observe that in all these scenarios, DRL offers the same performance as FA in terms of user demand satisfaction.

2) DRL consistently outperforms FA in terms of power consumption. For example, in Fig. 2(a), when the user demand is 10Mbps, the power consumption given by DRL is about 18% less than that of FA. Interestingly, DRL offers comparable or sometimes even better performance than SA. That is because SA randomly chooses a RRH to serve a user, while DRL tends to select the best RRH(s) for serving users under the guidance of a DNN.

3) In addition, no matter which method is used, the total power consumption increases monotonically with the user demand as expected. Moreover, compared to FA, the power savings achieved by DRL diminish with the user demand. This is because higher user demands lead to higher resource usages, which result in higher power consumption, and when the user demand is sufficiently high, then all RRHs (along with all available power) need to be utilized no matter which method is used.

In summary, under the guidance of a DNN, DRL can achieve significant power savings (compared to FA), and meanwhile satisfy user demands (as long as there exists a feasible solution).

We also evaluated the long-term performance of our proposed DRL framework in a highly dynamic user demand scenario. We set the total number of RRHs to 4, and considered a single user, which, however, had a demand uniformly distributed in a large range of [10, 90]Mbps. We set each decision epoch to 3 seconds. The user demand remained the same during a decision epoch, and may change in the next epoch. Note that the transition power was counted for DRL during the simulation. We calculated the average power consumption over a relatively long timeslot of 5-minus. As shown in Fig. 3, we can observe the fluctuation of power consumption due to the change of the user demand. Similar as in the above scenarios, DRL consistently outperforms FA in terms of total power consumption. This justifies effectiveness of DRL in highly dynamic cases. Note that we did not simulate SA in this scenario since sometimes it cannot satisfy the user demand.



(a) Scenario 1: 6 RRHs and 3 users

(b) Scenario 2: 6 RRHs and 4 users

(c) Scenario 3: 8 RRHs and 4 users

Fig. 2. The total power consumption VS. the user demand

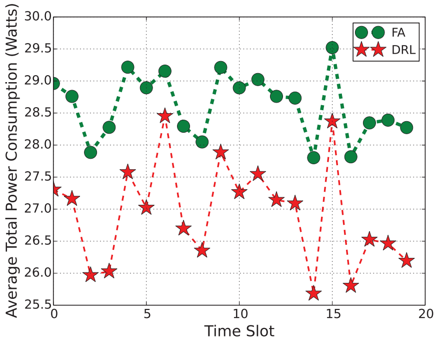


Fig. 3. Scenario 4: The average total power consumption in a time-varying user demand scenario

VI. Conclusions

In this paper, we present a novel DRL-based framework for power-efficient resource allocation in cloud RANs. Specifically, defined the action space, state space and reward function for the DRL agent, applied a DNN to approximate the action-value function for action decisions, and formally formulated the resource allocation problem (in a decision epoch) for a given set of active RRHs as a convex optimization problem. We evaluated the proposed DRL-based framework by comparing it with two widely-used baselines, SA and FA, via simulation. Simulation results showed that under the guidance of a DNN, the proposed DRL-based framework can achieve significant power savings while satisfying user demands; and moreover, it can well handle highly dynamic cases.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.*, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, *arXiv preprint*, 2016, arXiv:1603.04467.

[2] G. D. Arnold, R. Evans, H. v. Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris and B. Coppin, Deep reinforcement learning in large discrete action spaces, *arXiv preprint*, 2016, arXiv: 1512.07679.

[3] G. Auer, V. Giannini, C. Desset, I. Godor, P. Skillermark, *et al.*, How much energy is needed to run a wireless network? *IEEE Wireless Communications*, Vol. 18, No. 5, 2011, pp. 40–49.

[4] S. Bhaumik, S. P. Chandrabose, M. K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan and T. Woo, CloudIQ: a framework for processing base stations in a data center, *ACM MobiCom'2012*, pp. 125–136.

[5] China Mobile, C-RAN: the road towards green RAN, *White Paper*, 2011.

[6] B. Dai and W. Yu, Energy efficiency of downlink transmission strategies for cloud radio access networks, *IEEE Journal on Selected Areas in Communications*, Vol. 34, No. 4, 2016, pp. 1037–1050.

[7] Y. Duan, X. Chen, R. Houthooft, J. Schulman and P. Abbeel, Benchmarking deep reinforcement learning for continuous control, *ICML'2016*.

[8] J. N. Foerster, Y. M. Assael, N. d. Freitas and S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, *NIPS'2016*.

[9] S. Gu, T. Lillicrap, I. Sutskever and S. Levine, Continuous deep Q-Llearning with model-based acceleration, *ICML'2016*.

[10] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, *Book in preparation for MIT Press*, 2016. Available at http://www.deeplearningbook.org/

[11] H. v. Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double Q-learning, *AAAI'2016*.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, Continuous control with deep reinforcement learning, *ICLR'2016*.

[13] S. Luo, R. Zhang and T. J. Lim, Downlink and uplink energy minimization through user association and beamforming in C-RAN, *IEEE Transactions on Wireless Communications*, Vol. 14, No. 1, 2015, pp. 494–508.

[14] M. S. Lobo, L. Vandenberghe, S. Boyd and H. Lebret, Applications of second-order cone programming, *Linear Algebra and its Applications Journal*, Vol. 284, No. 1, 1998, pp. 193–228.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, *et al.* Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[16] M. Peng, K. Zhang, J. Jiang, J. Wang and W. Wang, Energy-efficient resource assignment and power allocation in heterogeneous cloud radio access networks, *IEEE Transactions on Vehicular Technology*, Vol. 64, No. 11, 2015, pp. 5275–5287.

[17] Y. Shi, J. Zhang and K. B. Letaief, Group sparse beamforming for green Cloud-RAN, *IEEE Transactions on Wireless Communications*, Vol. 13, No. 5, 2014, pp. 2809–2823.

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre *et al.*, Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7587, 2016, pp. 484–489.

[19] R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction, *MIT Press*, Vol. 1, No. 1, 1998.

[20] K. Sundaresan, M. Y. Arslan, S. Singh, S. Rangarajan and S. V. Krishnamurthy, FluidNet: a flexible cloud-based radio access network for small cells, *ACM MobiCom'13*, pp. 99-110.

[21] J. Tang, G. Xue and W. Zhang, Cross-layer optimization for end-to-end rate allocation in multi-radio wireless mesh networks, *Wireless Networks*, Vol. 15, No. 1, 2009, pp. 53–64.

[22] H. Taub and D. L. Schilling, Principles of communication systems, *McGraw-Hill Higher Education*, 1986.

[23] A. Wiesel, Y. C. Eldar and S. Shamai, Linear precoding via conic optimization for fixed MIMO receivers, *IEEE Transactions on Signal Processing*, Vol. 54, No. 1, 2006, pp. 161–176.

[24] C. Xu, F. X. Lin and L. Zhong, Device drivers should not do power management, *ACM APSys'2014*.

[25] C. Xu, F. X. Lin, Y. Wang and L. Zhong, Automated OS-level device runtime power management, *ACM SIGARCH Computer Architecture News*, Vol. 43, No. 1, 2015, pp. 239–252.