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

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

Jeon hoseong

Content

- Introduction
- System model
- DRL-based resource allocation framework
- Performance evaluation
- Conclusion

Introduction

- Cloud Radio Access Networks(cloud RANs) have become key technology for 5G (massive wireless traffic data)

- Unlike traditional RANs, designed with separate BBUs(BaseBand Units) and remote radios

- Distributed Remote Radio Heads(RRHs) take the responsibility of compressing and forwarding the receives radio signals from wireless users to BBUs through fronthaul link

- In a cloud RANs, a RRH only needs to maintain some basic transmission functions

Introduction

- Why we use Reinforcement Learning?

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.

Especially, in this paper, they use Deep Reinforcement Learning(DRL).

Introduction

- Composition of DRL technique

1) an offline: Deep Neural Network (DNN) construction phase, which correlates the value function with corresponding states and actions.

2) an online: Dynamic deep Q-learning phase for action selection, system control, and dynamic network updating.

Introduction

- What is the problem?

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*.

-> We need to reduce the size of the action space!

Introduction

- What is the goal?

1) Minimize total power consumption,

2) while ensuring that the demand of each wireless user is satisfied.

Introduction

- What is the contribution of this paper?

1. Present a DRL-based framework for dynamic resource allocation in a cloud RAN.

2. Define the action space, state space and reward function for the DRL agent.

3. Formulate allocation problem (in each decision epoch) for a given set of active RRHs as a convex optimization problem.

4. Simulation results well justify the effectiveness of the proposed DRL-based framework on power efficiency and handling highly dynamic case.

System model

In single cell in cloud RAN,

Set of R RRHs $\mathscr{R}$ = {1,2,⋯,R}

Set of users $\mathscr{U}$ = {1,2,⋯,U}

-> 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.

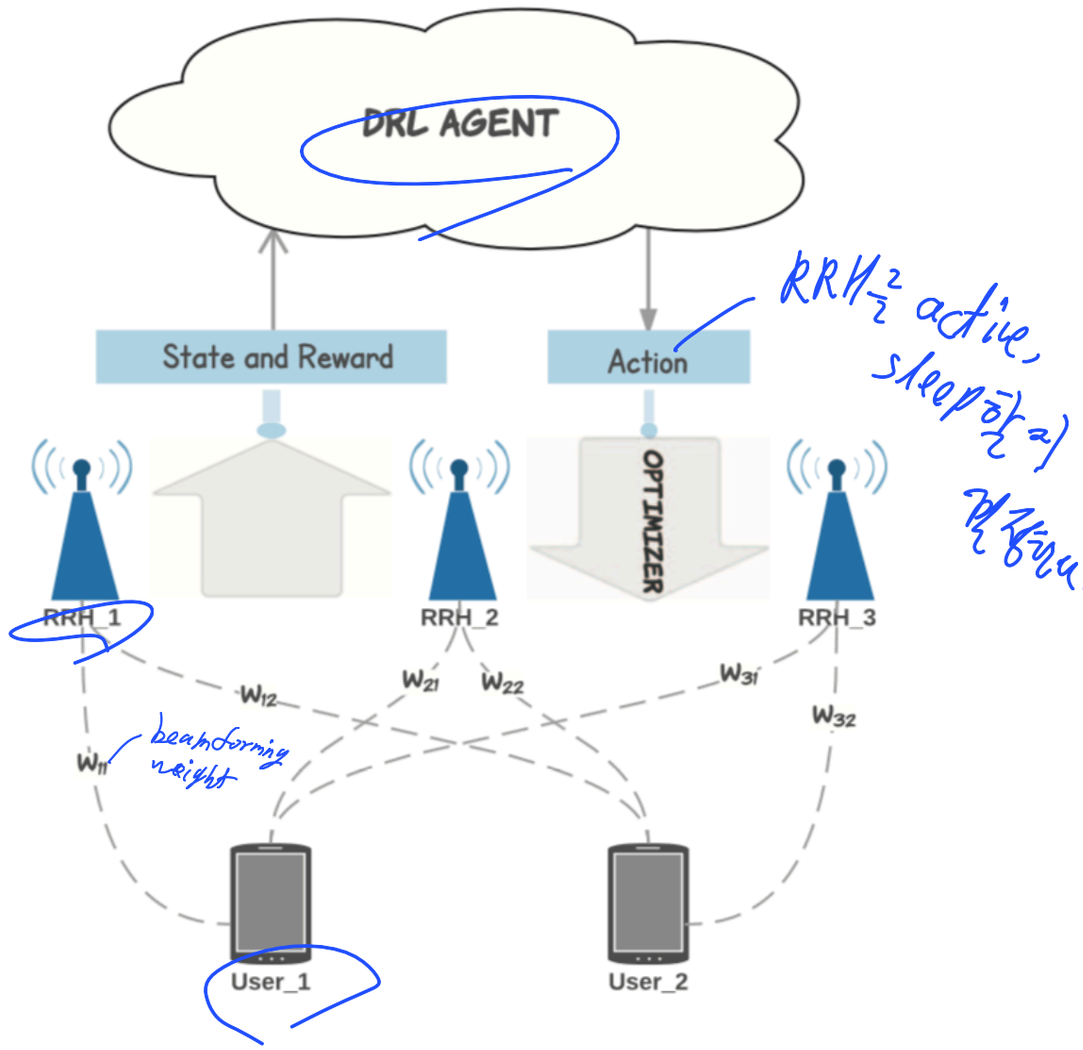


Fig. 1. DRL-based dynamic resource allocation in a cloud RAN

System model

The reward can be calculated by the DRL agent.

- Assumption:

1. All RRHs and users are equipped with a single antenna.
2. The information of RRHs can be shared in a centralized manner.
3. all users can access all RRHs in the cell.

The DRL agent will make an action decision a_k based on the system state 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.

System model

Signal-to-Interference-plus-Noise Ratio (SINR) at the receiver of user u can be given as:

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

where $h_u = [h_{1u}, h_{2u}, ..., h_{Ru}]^T$ and each element h_{ru} is the channel gain from RRH r to user u

$w_u = [w_{1u}, w_{2u}, ..., w_{Ru}]^T$ and each element w_{ru} is the beamforming weight from RRH r to user u.

.

System model

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

이 값들은 최소화하는 값이 모두.

System model

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

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 $\mathcal{A} \cup \mathcal{S} = \mathcal{R}$.

System model

In this paper, we account for the transition power
(i.e from sleep/active to active/sleep modes).

Let $\mathcal{G}$ = 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, it is significant and is thus essential to be considered in a power minimization framework.

(meaning of this paper!)

System model

- The total power consumption of RRHs in current epoch :

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

-> 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.

System model

-> 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.

DRL-based resource allocation framework

- DRL-based dynamic resource allocation framework

: minimizes total power consumption while ensuring that the demand of each user is satisfied.

- To reduce the action space size in the framework, this paper propose two-step method in each decision epoch.

1. The DRL agent determines the set of RRHs to turning on (or into sleep), which can limit the corresponding action space size.

2. Next, the DRL agent derives an optimal resource allocation (beamforming) solution based on the given set of active RRHs by solving a convex optimization problem.

DRL-based resource allocation framework

Generalized form of DRL comprises

1) Offline : DNN construction phase
2) Online: Deep Q-learning phase

DRL-based resource allocation framework - Offline

1. The offline phase adopts a DNN

: To derive the correlation between each state-action pair (s, a) of the system under control and its value function Q(s, a), which is given by:

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

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.

DRL-based resource allocation framework - Offline

The offline DNN construction needs to accumulate enough samples of value estimates and the corresponding (s,a) for constructing a sufficiently accurate DNN.

The use of experience memory in this offline procedure can smooth out learning and avoid oscillations or divergence.

-> Why? I think, by using memory, we can do additional learning.

DRL-based resource allocation framework - Online

1) In epoch ’t_k’, DRL agent derives estimated Q-value from DDN

2) Input : (s_k, a_k) for each a_k, use

$$\epsilon - greedy\ policy$$

(a_k: with highest estimated Q-value : $1 - \epsilon$, Random action : ϵ)

3) After observing immediate reward r_k and s_(k+1) from the cloud RAN

-> (s_k, a_k, r_k, s_(k+1)) will be stored into experience memory D.

4) At the end of the decision epoch update parameter θ of DNN with N_B samples from the D. (N_B : capacity of mini batch)

DRL-based resource allocation framework

such updating can also be done every T epoches (T > 1) to reduce the time complexity.

5) Sample minibatch -> Update Q-target -> Update loss function

- Used Parameters

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].

capacity of experience memory $N_D = 500$

the capacity of mini batch $N_B = 32$.

the reward discount $\mu = 0.9$.

Advantage of DRL-based framework

1) The DRL-based framework is scalable for a larger state space

2) The DRL-based framework could deal with the case of continuous state space, which is distinct from traditional RL

3) The DRL-based framework is suitable for dynamic resource allocation in wireless networks.

DRL-based resource allocation framework

Let’s define the state space, action space and reward function of the DRL-based framework as follows:

1. *State Space*: The state of DRL agent consists of two components

1) The binary (active or sleep) state $M_r \in 0,1$ of each RRH r

2) 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).

DRL-based resource allocation framework

2. *Action Space*:

We restrict the DRL agent to make the decision on a single RRH in each decision epoch.

-> DRL agent determines which RRH to turn on or into sleep.

-> After executing an action, DRL agent can derive an active set of RRHs (A).

3. *Reward*:

The reward is simultaneously minimizing the network power consumption and satisfying user demands.

we define the immediate reward that the DRL agent receives as

$$P_{max} - P(A, S, G)$$ total power consumption

where P_{max} is the maximum possible value of total power consumption and P (A, S, G) gives the actual total power consumption

DRL-based resource allocation framework

Algorithm 1 The DRL-based Framework

Offline:

1: Load the historical state transition profiles and the corresponding estimated $Q(\boldsymbol{s}, \boldsymbol{a})$ into experience memory $\mathcal{D}$ with a capacity of $N_{\mathcal{D}}$;
2: Pre-train the DNN with input pairs $(\boldsymbol{s}, \boldsymbol{a})$ and the corresponding estimated $Q(\boldsymbol{s}, \boldsymbol{a})$;

Online:

1: **for** each decision epoch t_k **do**
/**Operation**/
2: With probability ϵ, randomly select an action; otherwise $\boldsymbol{a}_k = \text{argmax}_{\boldsymbol{a}}\, Q(\boldsymbol{s}_k, \boldsymbol{a}_k; \theta)$, where $Q(\cdot)$ is estimated by the DNN;
3: Execute action $\boldsymbol{a}_k$;
4: Derive the set of active RRHs $\mathcal{A}$;
/**Optimization**/
5: Obtain the optimal beamforming solution based on the given $\mathcal{A}$ by solving CP-beamforming;
6: Reconfigure the network with the beamforming solution $\{w_{r,u}\}$;
/**Updating**/
7: Observe the reward r_k and the new state $\boldsymbol{s}_{k+1}$;
8: Store the state transition $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ into $\mathcal{D}$;
9: Randomly sample a minibatch of state transitions; $(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1})$ with a size of $N_{\mathcal{B}}$;
10: Target $y_i = r_i + \mu \max_{\boldsymbol{a}'} Q(\boldsymbol{s}_{i+1}, \boldsymbol{a}'; \theta)$;
11: Update the DNN's weights with a loss function of $(y_i - Q(\boldsymbol{s}_{i+1}, \boldsymbol{a}'; \theta))^2$;
12: **endfor**

Performance evaluation

- Goal: Compared the proposed DRL-based framework with two baseline solutions:

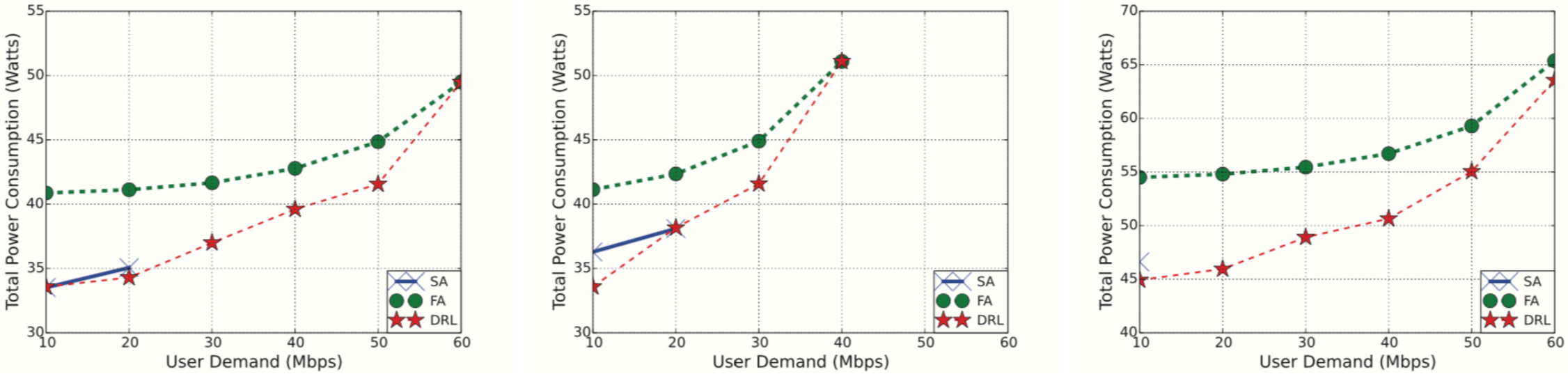
1) *Single BS Association (SA)*:

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.

Performance evaluation



(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

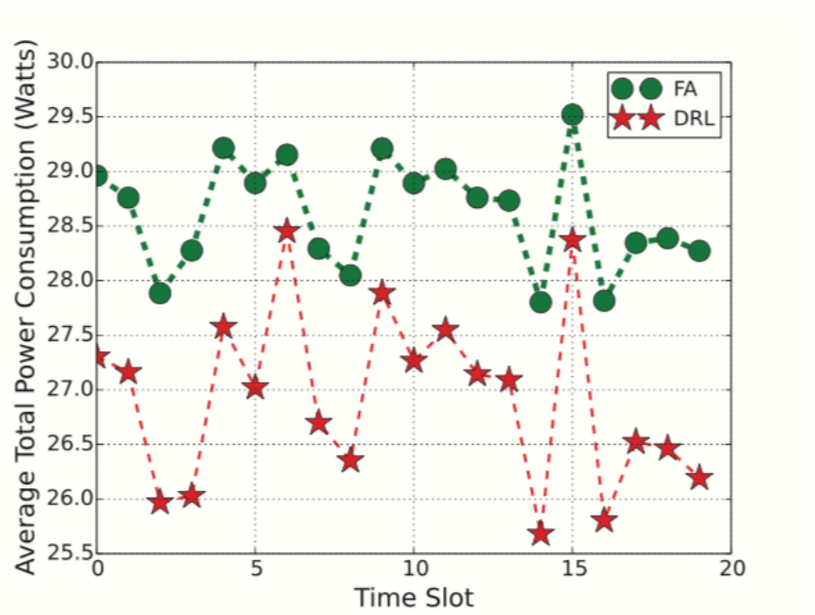


Fig. 3. Scenario 4: The average total power consumption in a time-varying user demand scenario

Performance evaluation

- Graph & Simulation Analysis

1. SA는 user demand가 커지면 가능한 solution을 못 찾는다.
반면에, FA는 해가 존재한다면, 반드시 찾는다.
DRL은 User Demand 에 관해서 FA와 같은 성능을 가진다.

2. DRL은 power consumption에서 FA의 성능을 뛰어넘는다. (18% 더 적음)

또한, SA보다 더 좋은 성능을 보이는데, 이는 RRH를 random하게 선택을 하는가, DDN의 도움을 받는가의 차이이다.

Performance evaluation

3. 어떤 방법이든 total power consumption은 단조증가한다.

: higher user demand -> higher resource usages

-> higher power consumption

4. 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.

-> 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).

Conclusion

1. Present a novel DRL-based framework for power-efficient resource allocation in cloud RANs.

2. 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,

3. Formulated the resource allocation problem (in a decision epoch) for a given set of active RRHs as a convex optimization problem

4.Comparing it with two widely-used baselines, SA and FA

Conclusion

- **Summary**

Simulation results showed that under the guidance of a DNN,

1) Proposed DRL-based framework can achieve significant power savings

2) Satisfying user demands

3) It can well handle highly dynamic cases.

Thanks

Reference

1. *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