A Model-driven Deep Reinforcement Learning Heuristic Algorithm for Resource Allocation in Ultra-dense Cellular Networks

Xiaomin Liao, Jia Shi, *Member, IEEE*, Zan Li, *Senior Member, IEEE*, Lei Zhang, *Senior Member, IEEE*, and Baiqiang Xia

***Abstract*— Resource allocation in ultra dense network (UDN) is an multi-objective optimization problem since it has to consider the tradeoff among spectrum efficiency (SE), energy efficiency (EE) and fairness. The existing methods can not effectively solve this NP-hard nonconvex problem, especially in the presence of limited channel state information (CSI). In this paper, we investigate a novel model-driven deep reinforcement learning assisted resource allocation method. We first design a novel deep neural network (DNN)-based optimization framework consisting of a series of Alternating Direction Method of Multipliers (ADMM) iterative procedures, which makes the CSI as the learned weights. Then a novel channel information absent Q-learning resource allocation (CIAQ) algorithm is proposed to train the DNN-based optimization framework without massive labeling data, where the SE, the EE, and the fairness can be jointly optimized by adjusting discount factor. Our simulation results show that, the proposed CIAQ with rapid convergence speed not only well characterizes the extent of optimization objective with partial CSI, but also significantly outperforms the current random initialization method of neural network and the other existing resource allocation algorithms in term of the tradeoff among the SE, EE and fairness.**

***Index Terms*— Ultra-dense cellular networks, resource allocation, deep reinforcement learning, model-driven, optimization.**

I. Introduction

With the emergence of new broadband services and the Internet of Things (IoT) era, the fifth generation (5G) has become a reality and demand for 2020 and beyond [1]. The new communication paradigm is to reach 1000 times of the mobile data volume, 10 times of the spectral efficiency (SE) and energy efficiency (EE) compared with the current long term evolution (LTE) system [2]. To achieve these goals, the ultra-dense network (UDN) is considered as one of the key techniques for 5G and beyond mobile networks [3, 4]. With the dense deployment of base stations (BSs) or access points (APs), UDN increases the spatial reuse of wireless resources, thus it can significantly improve network capacity as well as per-user data rate. However, the dense deployment of BSs/APs inevitably results in inter-cell interference, which may deeply degrade the rate performance of cell-edge users in the UDNs [5]. To improve the performance of UDNs, dynamic resource allocation in UDNs thereby becomes a very important research focus.

There are various approaches for resource allocation in the conventional cellular system, which include the convex optimization based methods, the discrete optimization assisted approaches, and other heuristic methods. The convex optimization based methods normally transform a nonconvex mixed-integer problem into some approximated convex sub-optimal problems which may be solved locally with a certain number of iterations [6–9]. Although current approximation methods demand low computational complexity, they are usually assumed to know the perfect knowledge about the channel state information (CSI) [10, 11], which may be impractical for a dynamic wireless environment due to the huge overhead for feedback channel. In addition, the discrete optimization assisted approaches mainly utilize game theory or graph theory to solve the joint optimization problem effectively with acceptable amount of complexity [12, 13]. Nevertheless, they are hardly able to solve the multi-objective optimization problems. The existing literature only considers the partial optimization objectives of UDNs, which cannot improve network performance well [14, 15]. Especially, with the rise of the environment-friendly UDNs and user experience, the tradeoff among SE, EE and fairness has become one of the biggest challenges in the field of resource allocation for future UDNs [16, 17]. However, the heuristic methods such as greedy-based algorithms [18], genetic algorithms [19], only result in suboptimal solutions which may be far away from theoretical optimal ones. Due to the inherent scalability limitation, the aforementioned resource allocation methods cannot be directly extended to or are not suitable for UDN scenarios, where normally deploying a large number of low power small-cell BSs with high density as well as demanding a massive amount of data processing capability. Therefore, it is highly demanding that a new paradigm of resource allocation strategies should be developed for future UDNs.

Recently, deep reinforcement learning (deep RL) has become a new research trend in the application of artificial intelligence, and it is playing as a viable tool to tackle dynamic resource

This work was supported in part by the Key Project of National Natural Science Foundation of China under Grant 61631015, in part by the National Natural Science Foundation for Distinguished Young Scholar of China under Grant 61825104, in part by China Postdoctoral Science Foundation under Grant 2018M631122, and in part by the Key Laboratory Foundation of Ministry of Industry and Information Technology under Grant KF20181912.

X. Liao, J. Shi, and Z. Li are with the State Key Laboratory of Integrated Service Networks, School of Telecommunications Engineering, Xidian University, Xi'an 710071, China. X. Liao is also with the School of Information and Communications, National University of Defense Technology, Xi'an 710106, China. (E-mail: lxm8410@163.com, jiashi@xidian.edu.cn, zanli@xidian.edu.cn).

L. Zhang is with the School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK. (E-mail: Lei.Zhang@glasgow.ac.uk).

B. Xia is with Top Data Science Ltd, Kuortaneenkatu 2, FI-00510 Helsinki, Finland. (E-mail: baiqiang.xia@topdatascience.com).

allocation problems for wireless communication systems [20–23]. As a newly-emerging paradigm, the authors in [24] have presented a Deep Q-learning Network framework for dynamic resource allocation, which can solve the complicated real-time control problems and the usage of energy-harvesting in UDNs without any prior knowledge about energy arrival, data arrival and CSI. The new intelligent methodology not only imitates human behavior to learn knowledge through big data training that traditional methods cannot obtain, but also adopts a trial-and-error search method to dynamic real-time interact with the environment to enable unprecedented automation and optimization on resource allocation, especially for multi-objective resource optimization problem with partial CSI that are difficult to solve by conventional convex optimization methods. Hence, the new analytic platform introduces new opportunities for the fields of resource allocation in UDNs.

However, most of the existing applications of using deep neural network (DNN) are seen as a black box trained by exploiting a large amount of data available [25–27]. Hence, this data-driven deep learning makes neural network topology being lack of theoretical understandings and explanations. To avoid the above drawback, the authors in [28] have proposed the model-driven deep learning approach which makes the designed neural network predictable. In general, the model-driven deep learning constructs the model family with a large set of unknown parameters based on the domain knowledge. Meanwhile, this new approach retains the powerful learning ability of the deep learning approach and specifically, is able to overcome the difficulties in network topology selection and make the network structure explainable and predictable, thus it is superior to the traditional data-driven deep learning in the trends of standardization and commercialization of machine learning. Therefore, this new approach has been successfully applied in imaging sciences [29], and it is also able to provide the feasibilities and effectiveness for resource allocation in UDNs. To the best of our knowledge, there has been no reported research works investigating the model-driven deep reinforcement learning assisted resource allocation for future wireless networks, especially for UDNs. Against the above state-of-the-art, in this paper, we focus on studying how to efficiently allocate different kinds of wireless resources based on the model-driven deep learning approach in the context of the UDNs conceived. The main contributions of our paper are summarized as follows.

- Resource allocation in the UDN is an NP-hard nonconvex problem since it has to consider the tradeoff among the SE, EE and fairness. We propose a novel model-driven deep reinforcement learning assisted resource allocation method, which theoretically decouples the multi-objective problem into two parts based on the general theory of deep RL, where maximizing SE is used to build the DNN, while the EE and fairness are considered as the rewards to train the designed DNN.
- To make our DNN explainable and predictable, we build a novel DNN-based optimization framework based on repeatedly operating the Alternating Direction Method of Multipliers (ADMM) iterative procedures, which considers the CSI as the learned weights. In the designed optimization framework, the number of layers, the number of neurons per layer, weights and non-linear transforms of neurons are characterized and optimized. Furthermore, the proposed DNN-based optimization framework can be applied to solve the multi-objective optimization problems being lack of training data while presenting communication infrastructures determined.
- To train the DNN-based optimization framework with limited labeling data, a novel resource allocation algorithm, namely CIAQ, is proposed for the UDN considered. Specifically, the proposed CIAQ algorithm sets the EE and the fairness as rewards to learn the unknown weights autonomously and dynamically, and is capable of making the resource allocation decisions within a relatively small number of iterations based on limited CSI.
- The comprehensive performance analysis is carried out for the proposed CIAQ algorithm, as well as comparing the performance with those of existing algorithms. Our simulation results show that, the CIAQ algorithm with rapid convergence ability not only well characterizes the extent of optimization objective by adjusting discount factor, but also significantly outperforms random initialization method of neural network and the other existing resource allocation algorithms in term of the tradeoff among the SE, EE and fairness. Therefore, the CIAQ algorithm may constitute a promising candidate that facilitates resource allocation for future UDNs.

The rest of paper is organized as follows. Section II introduces the system model and formulates our optimization problems. Section III designs a DNN-based optimization framework. Section IV proposes a CIAQ algorithm. Performance results of the designed DNN-based optimization framework and proposed CIAQ algorithm are shown in Section V. Finally, we summarize the paper in Section VI.

II. System Model and Problem Formulation

A. System Model

Let us consider the downlink of an UDN system with M small-cell BSs and K authorized mobile users, as shown in Fig. 1. Each small-cell BS locates at the center of every cell, and its authorized mobile users are randomly distributed in the cell. In our UDN, assume that there are overlapping areas between every two adjacent small cells. We assume that each of the communication terminals is equipped with one antenna for signal transmission. To maximize the use of radio resources and avoid trivial cases, the frequency reuse factor is set to one. In order to avoid intra-cell interference, each user in each cell is assumed to allocate one subcarrier only, and thus all signals in the same cell are orthogonal. The N orthogonal subcarriers used in a cell can be reused in each of the adjacent cells. Nevertheless, the users in the overlapping area are served by the nearest small-cell BS, and they may suffer from heavy inter-cell interference (ICI) due to the fact that the same spectrum resources might be used. Therefore, the inter-cell interference comes from adjacent cells, as shown by the dashed arrow in Fig. 1.

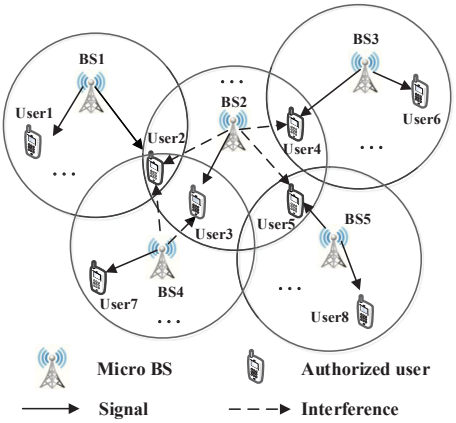


Fig. 1. Conceptual structure of the downlink in UDNs.

In order to coordinate resource allocation, one of the small-cell BSs plays as the central unit, and the other BSs are connected with this central unit through, e.g., X2 interface. In this paper, we assume that the desired signal CSI is not available at the BSs. However, to perform the learning process, we assume a long term average interference power received by each UE can be estimated and feedback to the serving BS through feedback channel [30]. It is worth to mention that this information exchange requires very limited resource at a very low frequent manner, compared to the desired signal CSI. To improve network performance and user experience, our resource allocation aims to jointly maximize SE, EE and fairness.

In our UDN, let $d_{m,k}$ represent the associated relationship between BS $m(m \in \mathcal{M} = \{1, \ldots, M\})$ and user $k(k \in \mathcal{K} = \{1, \ldots, K\})$. The $d_{m,k}$ is given as

$$d_{m,k} = \begin{cases} 0 & \text{if user } k \text{ is not associated with BS } m, \\ 1 & \text{if user } k \text{ is associated with BS } m. \end{cases} \quad (1)$$

Let us assume that the user is associated with the BS with the highest marginal utility [31]. When user k is associated with BS m on subcarrier $n(n \in \mathcal{N} = \{1, \ldots, N\})$, the spectrum status $A^n_{m,k}$ can be decided using the following rules

$$A^n_{m,k} = \begin{cases} 0 & \text{if subcarrier } n \text{ is not assigned to user } k \text{ in BS } m, \\ 1 & \text{if subcarrier } n \text{ is assigned to user } k \text{ in BS } m. \end{cases} \quad (2)$$

As described, the users in different cells can be assigned the same subcarrier, such as, the ICI of user k served by BS m on subcarrier n can be expressed by

$$I^n_{m,k} = \sum_{m'=1, m' \neq m}^{M} \sum_{k'=1}^{K} d_{m',k'} A^n_{m',k'} p^n_{m',k'} g^n_{m',k}, \quad (3)$$

where $p^n_{m',k'}$ represents the transmit power of BS m' to user k' on subcarrier n, $g^n_{m',k}$ is the square of channel gain from BS m' to user k on subcarrier n. When $A^n_{m,k} = 1$, the signal-to-interference-plus-noise ratio (SINR) of user k served by BS m on subcarrier n is given by

$$SINR^n_{m,k} = \frac{p^n_{m,k} g^n_{m,k}}{\sigma^2_{m,k} + I^n_{m,k}} = \frac{p^n_{m,k} g^n_{m,k}}{\sigma^2_{m,k} + \sum\limits_{m'=1, m' \neq m}^{M} \sum\limits_{k'=1}^{K} d_{m',k'} A^n_{m',k'} p^n_{m',k'} g^n_{m',k}}, \quad (4)$$

where $\sigma^2_{m,k}$ is the power of Additive White Gaussian Noise (AWGN) from BS m to user k. Let us assume that all the channels experience independent Rayleigh fading. Note that, when user k of BS m and user k' of BS $m'(m' \neq m)$ are both allocated subcarrier n simultaneously, $p^n_{m',k'}$ will interfere with user k of BS m.

B. Problem Formulation

To achieve the tradeoff among the SE, EE and fairness in UDNs, the multi-objective optimization problem can be described as

$$(\mathcal{P}_1) \max_{\{A^n_{m,k}, p^n_{m,k}\}} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{k=1}^{K} \log_2(1 + d_{m,k} A^n_{m,k} SINR^n_{m,k}) \quad (5)$$

$$\max_{\{A^n_{m,k}, p^n_{m,k}\}} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{B^n_{m,k} \log_2(1 + d_{m,k} A^n_{m,k} SINR^n_{m,k})}{p^n_{m,k}} \quad (6)$$

$$\min_{\{A^n_{m,k}, p^n_{m,k}\}} \mathrm{D}\Big\{ \Big(\sum_{m=1}^{M} \sum_{n=1}^{N} B^n_{m,k} \log_2(1 + d_{m,k} A^n_{m,k} SINR^n_{m,k}), \forall k \in \mathcal{K} \Big) \Big\} \quad (7)$$

subject to

$$(C_1) \quad \sum_{n=1}^{N} \sum_{k=1}^{K} d_{m,k} A^n_{m,k} p^n_{m,k} \leq P^{max}_m, \forall m \in \mathcal{M}. \quad (8)$$

Above, in the optimization problem $(\mathcal{P}_1)$, the objective functions (5) and (6) are for SE (bps/HZ) and EE (bps/W) of the UDN considered, respectively. Note that, in the objective function (7), it utilizes the variance of the obtained throughput ((bit/s)^2) between the associated mobile users to quantify fairness [32]. In particular, the smaller the variance, the better fairness among users can be achieved by our UDN. The parameter $B^n_{m,k}$, given by the objective functions (6) and (7), is the bandwidth of subcarrier n from BS m to user k. Furthermore, we should guarantee that the total transmitted power of every small-cell BS is under its power limit denoted as P^{max}_m, given by the power constraint (C_1).

The multi-objective optimization problem in (5), (6), (7) and (8) is an NP-hard nonconvex problem. To solve the problem, it is common practice to transfer the NP-hard nonconvex problem into a sequence of convex programs and develop specific efficient algorithms to solve each convex program separately [16, 33]. However, it needs to run the algorithms by a relatively high number of iterations to get a satisfactory result which still might be far away from the optimal one. Furthermore, it is also

a challenge to estimate CSI and other parameters, especially for the UDN scenarios. Therefore, in this paper, we first decompose the multi-objective optimization problem into two parts, one is the objective function (5) and the constraint (C_1), the other is the objective functions (6) and (7). In Section III, we design a DNN-based optimization framework by solving the objective function (5) and the constraint (C_1). The optimization framework takes CSI as the unknown weights and gets the resource allocation results corresponding to the maximum SE. Then, in Section IV, we propose a novel resource allocation algorithm named CIAQ, which makes the objective functions (6) and (7) as the rewards to train the DNN-based optimization framework, and achieves a promising tradeoff among the SE, EE, and fairness by adjusting the discount factor of the designed loss function. In this way, we solve the multi-objective optimization problem well with partial CSI.

III. General Theory of Resource Allocation

In this section, firstly, we describe the general theory of deep RL based resource allocation for UDNs. Specifically, we design a DNN-based optimization framework which iteratively operates the so-called Alternating Direction Method of Multipliers (ADMM) schemes.

A. General Theory of Deep RL based Resource Allocation

Unlike supervised learning, the RL agent is employed to make actions to influence their environment based on the current state of the network, and to find out which action yields the biggest reward by traversing all possible actions [34]. In particular, RL adopts a trial-and-error search method to interact with the network environment, and obtains the resource allocation policy without huge labeling data at each of a sequence of discrete time steps. Note that, the learning process is a finite Markov decision process (MDP), which is defined by state set $\mathcal{S}$, action set $\mathcal{A}$, reward function r, and policy π as follows.

State Set: $\mathcal{S}$ consists of $t+1$ states $\{s_0, s_1, \ldots, s_t\}$. At each time, the state observed by the agent for characterizing the network environment consists of two parts: user association information d_t and interference power I_t. Therefore, the state can be expressed as $s_t = [d_t, I_t]$. In the UDN considered, the agent only needs to obtain these two types of information for resource allocation.

Action Set: According to the current state, the agent can take an action $a_t \in \mathcal{A}$ based on the decision policy π. The action includes selecting a subcarrier and corresponding transmission power. Therefore, the action can be expressed as $a_t = [A_t, p_t]$. Once a decision is made, each authorized mobile user switches to the assigned subcarrier and adjusts the transmission power.

Reward: After taking the action a_t, the agent can calculate the environment's rewards r_t. The agent's sole objective is to maximize total rewards. Because the action a_t has a direct effect on rewards r_t, the rewards sent to the agent define what are the good and bad actions for the agent. In that case, the rewards in our system model can be expressed as

$$r = \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{B_{m,k}^{n}\log_2(1+d_{m,k}A_{m,k}^{n}SINR_{m,k}^{n})}{p_{m,k}^{n}} - \mathrm{D}(\sum_{m=1}^{M}\sum_{n=1}^{N}B_{m,k}^{n}\log_2(1+d_{m,k}A_{m,k}^{n}SINR_{m,k}^{n}), \forall k \in \mathcal{K}). \tag{9}$$

The reward function consists of two parts, the first part denotes the network EE, and the second part means the variance of the obtained throughput between authorized users, which represents the fairness of service quality. In (9), the agent's objective is to maximize the total rewards it receives, which aims to maximize EE and to minimize the variance of the obtained throughput between the authorized mobile users.

Policy: The goal of the agent is to learn a policy π: $s_t \in \mathcal{S} \rightarrow a_t \in \mathcal{A}$ for choosing the next action $\{A_{m,k}^{n}, p_{m,k}^{n}\}$ based on its current state s_t that produces the greatest possible expected cumulative rewards. Since the state-action space of our resource allocation problem is relatively large, it is impossible to use a look-up table to accomplish the update rule. In this context, we design a DNN-based optimization framework combining Q-learning to generate policy π. The input of the DNN-based optimization framework is the observed state set $\mathcal{S}$, and the output of the DNN-based optimization framework is defined as all the executable actions in action set $\mathcal{A}$. Further, each state-action pair has a corresponding Q-value $Q(s_t, a_t)$. In particular, each step selects the action that gets the maximum Q-value at each state, described as

$$a_t = \arg\ \max_{a\in\mathcal{A}} Q(s_t, a). \tag{10}$$

Before introducing the proposed deep RL framework, let us analyze the optimization problem ($\mathcal{P}_1$).

B. Problem Analysis

In order to tackle Problem ($\mathcal{P}_1$), let us analyze the general SE optimization problem alone, which can be transformed as

$$(\mathcal{P}_2)\ \min_{\{A_{m,k}^{n}, p_{m,k}^{n}\}}\ -\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K}\log_2(1+d_{m,k}A_{m,k}^{n}SINR_{m,k}^{n}) \tag{11}$$

subject to

$$\sum_{n=1}^{N}\sum_{k=1}^{K}d_{m,k}A_{m,k}^{n}p_{m,k}^{n} \le P_m^{max}, \forall m \in \mathcal{M}. \tag{12}$$

Based on the definition of convex optimization problem in [35], we can prove that the above SE optimization problem is a convex optimization problem. Upon considering ($\mathcal{P}_2$), we propose to apply the ADMM to solve the problem ($\mathcal{P}_2$), since the ADMM approach is in general more numerically stable and faster in convergence than conventional decomposition methods [36], When considering problem ($\mathcal{P}_2$), suggested by

[35], the corresponding augmented Lagrangian function is

$$L(A^n_{m,k}, p^n_{m,k}, \alpha_m, \mu) = -\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{K}\log_2(1+d_{m,k}A^n_{m,k} SINR^n_{m,k}) + \frac{1}{2\mu}\sum_{m=1}^{M}\left\{\left[max\left(0, \alpha_m - \mu\left(P^{max}_m - \sum_{n=1}^{N}\sum_{k=1}^{K} d_{m,k}A^n_{m,k}p^n_{m,k}\right)\right)\right]^2 - \alpha_m^2\right\}, \tag{13}$$

where $\alpha = \{\alpha_m, \forall m \in \mathcal{M}\}$ are Lagrangian multipliers and μ is penalty parameter. In this case, the unconstrained optimization problem can be expressed as

$$(\mathcal{P}_3) \qquad \min \;\; L(A^n_{m,k}, p^n_{m,k}, \alpha_m, \mu). \tag{14}$$

In order to get the optimal solution, the ADMM algorithm aims to find the optimal solution of $\{A^n_{m,k}, p^n_{m,k}\}$ by tackling the following two equations

$$\begin{cases} \nabla_{A^n_{m,k}} L(A^n_{m,k}, p^n_{m,k}, \alpha_m, \mu) = 0 \\ \nabla_{p^n_{m,k}} L(A^n_{m,k}, p^n_{m,k}, \alpha_m, \mu) = 0 \end{cases}. \tag{15}$$

For simplicity, let $\rho^n_{m,k}$ denote the sum of noise and interference from BS m to user k on subcarrier n, described as

$$\rho^n_{m,k} = \sigma^2_{m,k} + I^n_{m,k}. \tag{16}$$

In addition, we define

$$\omega^n_{m,k} = \sum_{n'=1}^{N}\sum_{k'=1}^{K} d_{m,k'}A^{n'}_{m,k'}p^{n'}_{m,k'} - d_{m,k}A^n_{m,k}p^n_{m,k}, \tag{17}$$

$$\psi^n_{m,k} = \alpha_m - \mu P^{max}_m + \mu\omega^n_{m,k}, \tag{18}$$

$$\upsilon^n_{m,k} = \mu\rho^n_{m,k} + \psi^n_{m,k}g^n_{m,k}. \tag{19}$$

Upon substituting (4), (16), (17), (18) and (19) into (15), we can solve (15) and get the unconstrained minimum point, given by

$$\begin{cases} A^{n(l+1)}_{m,k} = \dfrac{\sqrt{(\upsilon^{n(l)}_{m,k})^2 - 4\mu g^n_{m,k}(\rho^n_{m,k}\psi^{n(l)}_{m,k} - \frac{g^n_{m,k}}{\ln 2})} - \upsilon^{n(l)}_{m,k}}{2\mu d_{m,k}p^{n(l)}_{m,k}g^n_{m,k}} \\ p^{n(l+1)}_{m,k} = \dfrac{\sqrt{(\upsilon^{n(l)}_{m,k})^2 - 4\mu g^n_{m,k}(\rho^n_{m,k}\psi^{n(l)}_{m,k} - \frac{g^n_{m,k}}{\ln 2})} - \upsilon^{n(l)}_{m,k}}{2\mu d_{m,k}A^{n(l)}_{m,k}g^n_{m,k}} \end{cases}, \tag{20}$$

where $l \in \mathcal{L} = \{1, \ldots, L\}$ denotes the l-th iteration. Meanwhile, we set penalty parameter μ large enough, but does not have to be infinite. Furthermore, the Lagrangian multiplier α is updated as follows

$$\alpha^{(l+1)}_m = \max\{0, \alpha^{(l)}_m - \mu(P^{max}_m - \sum_{n=1}^{N}\sum_{k=1}^{K} d_{m,k}A^{n(l)}_{m,k}p^{n(l)}_{m,k})\}. \tag{21}$$

In (20) and (21), it commonly needs to run the ADMM algorithm in dozens of iterations to get a satisfactory result. Especially, the channel gain and noise power must be known in advance for calculating the optimal values. To overcome these challenges, we design a DNN-based optimization framework for the ADMM algorithm, which considers CSI as the unknown weights.

C. DNN-based Optimization Framework

The designed DNN-based optimization framework is derived from the ADMM iterative procedures, which aims to optimize SE alone. The DNN-based optimization framework comprises of neurons corresponding to different operations in the ADMM iterative procedures, and directed edges corresponding to the data flow between operations. Hence, the l-th layer of the DNN-based optimization framework corresponds to the l-th iteration of ADMM procedures. After entering the DNN-based optimization framework, the input data flow over multiple repetitive layers, which correspond to successive iterations in ADMM. When satisfying the convergence condition, the DNN-based optimization framework generates the resource allocation results.

As shown in Fig. 2, the DNN-based optimization framework constructs four types of network layers: data input layer, strategy generation layer, multiplier update layer, and decision output layer.

Data input layer: This layer provides current observational states from the network environment. There is no computation performed in any of the input neurons, which just pass on the input data to the following hidden nodes. There are two types of neurons in this layer, collected in the set of

$$\mathcal{D} = \{d_{m,k}, I^n_{m,k}, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}\}, \tag{22}$$

where $d_{m,k}$ given as (1) represents the associated relationship between BS m and user k, and $I^n_{m,k}$ given as (3) denotes the interference power from BS m to user k on subcarrier n under the current spectrum allocation strategy. The neuron $d_{m,k}$ contains $M \times K$ inputs, and the neuron $I^n_{m,k}$ contains $M \times K \times N$ inputs.

Next, the hidden layer performs computation and transfers information from the data input layer to the decision output layer. Specifically, the hidden layer contains two parts: strategy generation layer and multiplier update layer. There are L strategy generation layers, and also L multiplier update layers.

Strategy generation layer: This layer computes the resource allocation strategy. There are two types of neurons in this layer, i.e. the spectrum status neuron $A^n_{m,k}$, and the transmit power neuron $p^n_{m,k}$. The neuron $A^n_{m,k}$ contains $M \times K \times N$ data. According to (20), for instance, in the *(l+1)*-th layer, the non-linear transform of neuron $A^n_{m,k}$, i.e. activation function, is defined as

$$\varphi^{(l+1)}_{A^n_{m,k}} = \frac{\sqrt{(\upsilon^{n(l)}_{m,k})^2 - 4\mu g^n_{m,k}(\rho^n_{m,k}\psi^{n(l)}_{m,k} - \frac{g^n_{m,k}}{\ln 2})} - \upsilon^{n(l)}_{m,k}}{2\mu d_{m,k}p^{n(l)}_{m,k}g^n_{m,k}}. \tag{23}$$

The neuron $p^n_{m,k}$ contains $M \times K \times N$ data, similarly, in the *l+1*-th layer, the non-linear transform of neuron $p^n_{m,k}$ is defined as

$$\varphi^{(l+1)}_{p_{m,n}} = \frac{\sqrt{(\upsilon^{n(l)}_{m,k})^2 - 4\mu g^n_{m,k}(\rho^n_{m,k}\psi^{n(l)}_{m,k} - \frac{g^n_{m,k}}{\ln 2})} - \upsilon^{n(l)}_{m,k}}{2\mu d_{m,k}A^{n(l)}_{m,k}g^n_{m,k}}. \tag{24}$$

In particular, the square of channel gain $g^n_{m,k}$ and the noise power $\sigma^2_{m,k}$ are the parameters to be learned. Note that, in the first stage, we need to initialize $A^{n(0)}_{m,k}$, $p^{n(0)}_{m,k}$, $g^{n(0)}_{m,k}$ and $\sigma^{2(0)}_{m,k}$.

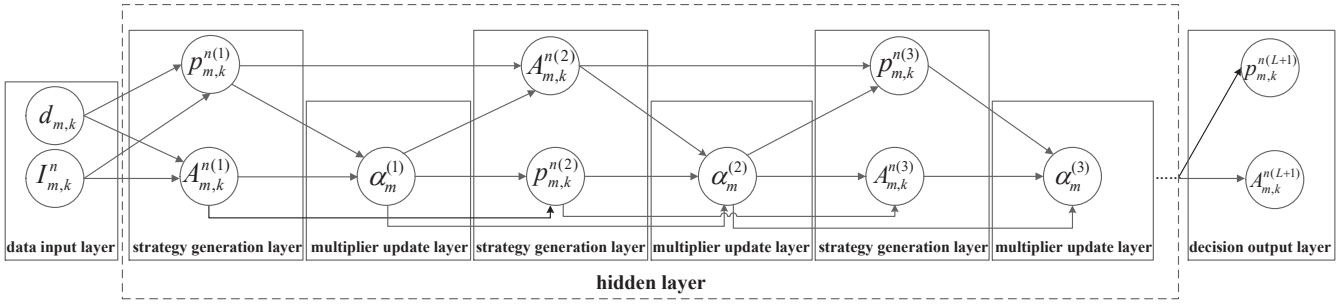


Fig. 2. The structure of the DNN-based optimization framework. The optimization framework not only contains a data input layer and a decision output layer, but also contains multiple repetitive strategy generation layers and multiplier update layers, which belong to the hidden layer in the typical neural network.

Multiplier update layer: This layer describes the updated states of Lagrangian multipliers. The inputs of this layer are $A^n_{m,k}$ and $p^n_{m,k}$ which are the outputs from the last strategy generation layer. Similar to the strategy generation layer, $\alpha^{(0)}_m$ also needs to be initialized in the first stage. The neuron α_m contains M data. Furthermore, in the *l+1*-th layer, the non-linear transform of neuron α_m is defined as

$$\varphi^{(l+1)}_{\alpha_m} = \max\{0, \alpha^{(l)}_m - \mu(P^{max}_m - \sum_{n=1}^{N}\sum_{k=1}^{K} d_{m,k}A^{n(l)}_{m,k}p^{n(l)}_{m,k})\}. \tag{25}$$

Decision Output Layer: This layer generates the final spectrum allocation strategy. Note that, it terminates, if $\left|A^{n(L+1)}_{m,k} - A^{n(L)}_{m,k}\right| \to 0$ and $\left|p^{n(L+1)}_{m,k} - p^{n(L)}_{m,k}\right| \to 0$, or the maximum iteration number of the ADMM algorithm exceeds the pre-defined threshold. Let us denote the final spectrum allocation results as $A^{n(L+1)}_{m,k}$ and $p^{n(L+1)}_{m,k}$. The neuron $A^{n(L+1)}_{m,k}$ contains $M \times K \times N$ outputs, and the neuron $p^{n(L+1)}_{m,k}$ contains $M \times K \times N$ outputs.

The designed DNN-based optimization framework can be employed to optimize the SEs of wireless networks. Moreover, the construction method of the DNN-based optimization framework can be used to solve the similar convex optimization problems which can be solved by ADMM algorithm. In Section IV, in order to optimize EE and fairness simultaneously, a novel deep RL heuristic algorithm is proposed based on the DNN-based optimization framework, where the objective functions (6) and (7) are defined as the rewards for training.

IV. Proposed Channel Information Absent Q-learning Resource Allocation (CIAQ) Algorithm

In this section, we propose a novel channel information absent Q-learning resource allocation algorithm, namely CIAQ, which can obtain the near-optimal resource allocation results. Based on the designed DNN-based optimization framework in Section III, the CIAQ algorithm consists of network learning stage and network training stage. The design objectives and principles of the CIAQ algorithm are presented in the next.

A. Design objective of CIAQ algorithm

The CIAQ algorithm is designed based on the proposed DNN-based optimization framework, which aims to maximize the SE during the network learning stage. Then, during the network training stage, the network EE and fairness are used as the rewards to construct a loss function and to train our DNN-based optimization framework. As shown in Fig. 3, the black dashed arrows indicate the network learning stage, and the black solid arrows indicate the network training stage. The environment contains the quintuple of events $\{D_t, A^n_{m,k}, p^n_{m,k}, r_{t+1}, D_{t+1}\}$, where D_t , $A^n_{m,k}$ and $p^n_{m,k}$ denote the observed values, the spectrum allocation status, and the transmit power allocation strategy at time t, respectively. Further, D_{t+1} and r_{t+1} denote the new observed values and the obtained rewards after taking the resource allocation results of $\{A^n_{m,k}, p^n_{m,k}\}$, respectively. Moreover, in Fig. 3, the learned network weights in the designed DNN-based optimization framework are denoted by $\{g^n_{m,k}, \sigma^2_{m,k}\}$. In addition, the CIAQ algorithm can be set as a discrete-time event. When a new user arrives, the algorithm is triggered to rerun. The red solid arrow represents the trigger information to run the algorithm.

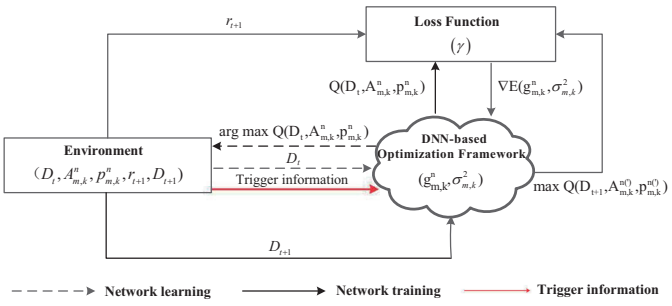


Fig. 3. The structure of CIAQ algorithm.

B. Principles of CIAQ algorithm

1) Network learning stage of CIAQ algorithm: This stage is a forward propagation process, and its principles can be summarized in Algorithm 1. In detail, the current observational state $D_t = \{d_{m,k}, I^n_{m,k}\}$ of the UDN is input into the DNN-based optimization framework. When the convergence condition is satisfied, the resource allocation result, denoted by $\{A^n_{m,k}, p^n_{m,k}\} = \text{argmax}\, Q(D_t, A^n_{m,k}, p^n_{m,k})$, is obtained along the data flow of the DNN-based optimization framework. Since the channel gain and noise power are unknown, the network weight set $\theta = \{g^{n(l)}_{m,k}, \sigma^{2(l)}_{m,k}\}^{L+1}_{l=1}$ of the DNN-based optimization framework needs to be initialized. Furthermore, $A^{n(0)}_{m,k}$, $p^{n(0)}_{m,k}$, and $\alpha^{(0)}_m$ also need to be initialized. Note that, as shown in Fig. 2, in the *l*-th iteration, the learned parameters are $g^n_{m,k}$ and $\sigma^2_{m,k}$.

Specifically, the network parameters can be initialized by two methods, which are model-based initialization method and random initialization method, respectively.

Model-based initialization: We initialize the network parameters based on the characteristics of resource allocation in UDNs. The channel gain $h_{m,k}^{n}$ is initialized as random values with Rayleigh distribution, and then the noise power $\sigma_{m,k}^{2}$ is initialized as white Gaussian noise. Meanwhile, according to the definitions of parameters, we set $A_{m,k}^{n(0)}$ and $\alpha_{m}^{(0)}$ as random values in the range $[0, 1]$, $p_{m,k}^{n(0)}$ as random value under the power limit P_{m}^{max}. The non-linear transforms of neurons in each layer are defined by (23), (24), (25).

Random initialization: In current deep learning literature, the weights in deep neural networks are generally initialized by random values. In this way, $h_{m,k}^{n}$, $\sigma_{m,k}^{2}$, $A_{m,k}^{n(0)}$, $p_{m,k}^{n(0)}$ and $\alpha_{m}^{(0)}$ are initialized as random values following a Gaussian distribution with zero mean and well-chosen variance level. Furthermore, the non-linear transforms of neurons in every layer are defined by rectified linear unit (ReLU) function, as described in [37].

	Algorithm 1: Network Learning Stage of CIAQ Algorithm
1:	Update current observational state $D_t = \{d_{m,k}, I_{m,k}^{n}\}$;
2:	Initialize $\theta = \{g_{m,k}^{n(l)}, \sigma_{m,k}^{2(l)}\}_{l=1}^{L+1}$, $A_{m,k}^{n(0)}$, $p_{m,k}^{n(0)}$, and $\alpha_{m}^{(0)}$;
3:	Set threshold ξ_A, ξ_p, maximum iteration number L, P_{m}^{max} and penalty parameter μ;
4:	**For** $l = 0, \ldots, L$
5:	Calculate $A_{m,k}^{n(l+1)}$, $p_{m,k}^{n(l+1)}$ and $\alpha_{m}^{(l+1)}$ along the DNN-based optimization framework in Fig. 2;
6:	**If** $\left\|A_{m,k}^{n(l+1)} - A_{m,k}^{n(l)}\right\| < \xi_A$ & $\left\|p_{m,k}^{n(l+1)} - p_{m,k}^{n(l)}\right\| < \xi_p$
7:	Output the resource allocation result $\{A_{m,k}^{n(l+1)}, p_{m,k}^{n(l+1)}\}$;
8:	**End If**
9:	**End For**

2) Network training stage of CIAQ algorithm: According to the Q-learning algorithm, the Q-value $Q(s_t, a_t)$ is updated via the following equation [34]

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha_k \cdot [r_{t+1} + \gamma \cdot \max_{a' \in \mathcal{A}} Q_k(s_{t+1}, a') - Q_k(s_t, a_t)], \quad (26)$$

where α_k and γ are learning rate and discount factor, respectively. Above, s_{t+1} and r_{t+1} represent the following state and the obtained reward after taking an action a_t in state s_t, a' denotes the executable action in state s_{t+1}, and $\mathcal{A}$ is the executable action set. In addition, $Q_k(s_t, a_t)$ denotes the Q-value in state s_t, and $\max_{a' \in \mathcal{A}} Q_k(s_{t+1}, a')$ represents the maximum Q-value among the executable action set $\mathcal{A}$ in state s_{t+1}. In (26), when $r_{t+1} + \gamma \cdot \max_{a' \in \mathcal{A}} Q_k(s_{t+1}, a') - Q_k(s_t, a_t) \to 0$, $Q_k(s_t, a_t)$ is similar to $Q_{k+1}(s_t, a_t)$. Therefore, the loss function can be expressed as

$$E = \frac{1}{2}[r_{t+1} + \gamma \cdot \max_{a' \in \mathcal{A}} Q_k(s_{t+1}, a') - Q_k(s_t, a_t)]^2. \quad (27)$$

Above, the discount factor γ determines the agent's horizon, and $0 \leq \gamma \leq 1$. When $\gamma = 0$, the agent is concerned only with rewards, known as EE and fairness. As γ approaches 1, the objective based on the current Q-value becomes SE oriented.

Next, let us compute the gradients of the loss function and utilize gradient backpropagation to train network weights $\theta = \{g_{m,k}^{n(l)}, \sigma_{m,k}^{2(l)}\}_{l=1}^{L+1}$ along the inverse direction of the data flow of the designed DNN-based optimization framework. In the *t*-th repetitive stage as shown by Fig. 4, there are five types of nodes for the data flow. Each node has multiple inputs and outputs. The data flow in forward propagation is indicated by the solid arrow, and gradient backpropagation direction is indicated by the dashed arrow. In the next, we briefly introduce the gradient computation for each node in a typical stage.

Decision output layer ($p_{m,k}^{n(L+1)}$, $A_{m,k}^{n(L+1)}$): As shown in Fig. 4 (a) and (b), this layer has two nodes, each of them has two inputs and one output. The inputs of the node $p_{m,k}^{n(L+1)}$ include $\alpha_{m}^{(L)}$ and $A_{m,k}^{n(L)}$, and its output contains $p_{m,k}^{n(L+1)}$. Similarly, the inputs of the node $A_{m,k}^{n(L+1)}$ are $\alpha_{m}^{(L)}$ and $p_{m,k}^{n(L)}$, and its output is $A_{m,k}^{n(L+1)}$. Moreover, the learned weights in this layer are $g_{m,k}^{n(L+1)}$ and $\sigma_{m,k}^{2(L+1)}$.

In the node $p_{m,k}^{n(L+1)}$, the gradient of loss function with regards to $g_{m,k}^{n(L+1)}$ and $\sigma_{m,k}^{2(L+1)}$ can be computed as

$$\begin{cases} \frac{\partial E}{\partial g_{m,k}^{n(L+1)}} = \frac{\partial E}{\partial p_{m,k}^{n(L+1)}} \frac{\partial p_{m,k}^{n(L+1)}}{\partial g_{m,k}^{n(L+1)}} \\ \frac{\partial E}{\partial \sigma_{m,k}^{2(L+1)}} = \frac{\partial E}{\partial p_{m,k}^{n(L+1)}} \frac{\partial p_{m,k}^{n(L+1)}}{\partial \sigma_{m,k}^{2(L+1)}} \end{cases}, \quad (28)$$

where $\frac{\partial E}{\partial p_{m,k}^{n(L+1)}}$ can be obtained by (9) and (27). Meanwhile, $\frac{\partial p_{m,k}^{n(L+1)}}{\partial g_{m,k}^{n(L+1)}}$ and $\frac{\partial p_{m,k}^{n(L+1)}}{\partial \sigma_{m,k}^{2(L+1)}}$ can be calculated by (24).

In the node $A_{m,k}^{n(L+1)}$, the gradient of loss function can be computed as

$$\begin{cases} \frac{\partial E}{\partial g_{m,k}^{n(L+1)}} = \frac{\partial E}{\partial A_{m,k}^{n(L+1)}} \frac{\partial A_{m,k}^{n(L+1)}}{\partial g_{m,k}^{n(L+1)}} \\ \frac{\partial E}{\partial \sigma_{m,k}^{2(L+1)}} = \frac{\partial E}{\partial A_{m,k}^{n(L+1)}} \frac{\partial A_{m,k}^{n(L+1)}}{\partial \sigma_{m,k}^{2(L+1)}} \end{cases}, \quad (29)$$

where $\frac{\partial E}{\partial A_{m,k}^{n(L+1)}}$ can be obtained by (9) and (27), $\frac{\partial A_{m,k}^{n(L+1)}}{\partial g_{m,k}^{n(L+1)}}$ and $\frac{\partial A_{m,k}^{n(L+1)}}{\partial \sigma_{m,k}^{2(L+1)}}$ can be derived by using (23).

In order to minimize the loss function, the weights can be updated along the inverse direction of the gradient, described as

$$\begin{cases} g_{m,k}^{n(L+1)} \leftarrow g_{m,k}^{n(L+1)} - \lambda_g \frac{\partial E}{\partial g_{m,k}^{n(L+1)}} \\ \sigma_{m,k}^{2(L+1)} \leftarrow \sigma_{m,k}^{2(L+1)} - \lambda_{\sigma^2} \frac{\partial E}{\partial \sigma_{m,k}^{2(L+1)}} \end{cases}, \quad (30)$$

where λ_g and λ_{σ^2} are both learning rates. The learning rates λ_g and λ_{σ^2} are updated according to the obtained loss function. For example, the learning rate λ_g in the *t+1*-th network training stage can be obtained by:

$$\lambda_g(t+1) = \begin{cases} 1.05\lambda_g(t) & E(t) < E(t-1), \\ 0.7\lambda_g(t) & E(t) > 1.04E(t-1), \\ \lambda_g(t) & \text{otherwise.} \end{cases} \quad (31)$$

Multiplier update layer ($\alpha_{m}^{(l)}$): As shown in Fig. 4 (c), this layer has one node, which has three inputs $A_{m,k}^{n(l)}$, $p_{m,k}^{n(l)}$ and $\alpha_{m}^{(l-1)}$. Meanwhile, its output $\alpha_{m}^{(l)}$ becomes the input for computing $A_{m,k}^{n(l+1)}$, $p_{m,k}^{n(l+1)}$ and $\alpha_{m}^{(l+1)}$. Note that, there is no learned weight in this layer.

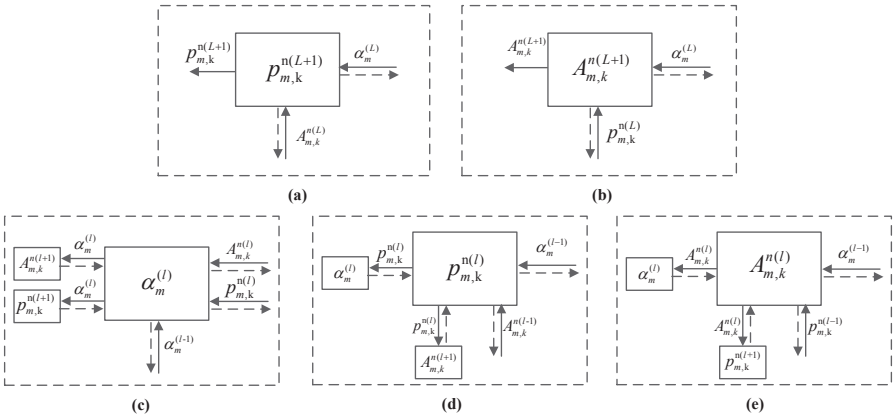


Fig. 4. The data flow and gradient backpropagation direction over each node.

Strategy generation layer ($p_{m,k}^{n(l)}$, $A_{m,k}^{n(l)}$): As shown in Fig. 4 (d), the inputs of the node $p_{m,k}^{n(l)}$ are $A_{m,k}^{n(l-1)}$ and $\alpha_m^{(l-1)}$, and its output is $p_{m,k}^{n(l)}$, which is the input for computing $A_{m,k}^{n(l+1)}$ and $\alpha_m^{(l)}$. The gradient of loss function with regards to $g_{m,k}^{n(l)}$ and $\sigma_{m,k}^{2(l)}$ can be computed by

$$\begin{cases} \frac{\partial E}{\partial g_{m,k}^{n(l)}} = \frac{\partial E}{\partial p_{m,k}^{n(l)}} \frac{\partial p_{m,k}^{n(l)}}{\partial g_{m,k}^{n(l)}} \\ \frac{\partial E}{\partial \sigma_{m,k}^{2(l)}} = \frac{\partial E}{\partial p_{m,k}^{n(l)}} \frac{\partial p_{m,k}^{n(l)}}{\partial \sigma_{m,k}^{2(l)}} \end{cases}, \tag{32}$$

where $\frac{\partial p_{m,k}^{n(l)}}{\partial g_{m,k}^{n(l)}}$ and $\frac{\partial p_{m,k}^{n(l)}}{\partial \sigma_{m,k}^{2(l)}}$ can be computed by (24). Further, $\frac{\partial E}{\partial p_{m,k}^{n(l)}}$ is the sum of gradients along the two input dashed arrows in Fig. 4 (d). Therefore, $\frac{\partial E}{\partial p_{m,k}^{n(l)}}$ can be expressed as

$$\frac{\partial E}{\partial p_{m,k}^{n(l)}} = \frac{\partial E}{\partial \alpha_m^{(l)}} \frac{\partial \alpha_m^{(l)}}{\partial p_{m,k}^{n(l)}} + \frac{\partial E}{\partial A_{m,k}^{n(l+1)}} \frac{\partial A_{m,k}^{n(l+1)}}{\partial p_{m,k}^{n(l)}}. \tag{33}$$

As shown in Fig. 4 (e), the inputs of the node $A_{m,k}^{n(l)}$ are $p_{m,k}^{n(l-1)}$ and $\alpha_m^{(l-1)}$, and its output is $A_{m,k}^{n(l)}$, which is the input to compute $p_{m,k}^{n(l+1)}$ and $\alpha_m^{(l)}$. The gradient of loss function with regards to $g_{m,k}^{n(l)}$ and $\sigma_{m,k}^{2(l)}$ can be computed by

$$\begin{cases} \frac{\partial E}{\partial g_{m,k}^{n(l)}} = \frac{\partial E}{\partial A_{m,k}^{n(l)}} \frac{\partial A_{m,k}^{n(l)}}{\partial g_{m,k}^{n(l)}} \\ \frac{\partial E}{\partial \sigma_{m,k}^{2(l)}} = \frac{\partial E}{\partial A_{m,k}^{n(l)}} \frac{\partial A_{m,k}^{n(l)}}{\partial \sigma_{m,k}^{2(l)}} \end{cases}, \tag{34}$$

where $\frac{\partial A_{m,k}^{n(l)}}{\partial g_{m,k}^{n(l)}}$ and $\frac{\partial A_{m,k}^{n(l)}}{\partial \sigma_{m,k}^{2(l)}}$ can be computed by (23). $\frac{\partial E}{\partial A_{m,k}^{n(l)}}$ is the sum of gradients along the two input dashed arrows in Fig. 4 (e). Therefore, $\frac{\partial E}{\partial A_{m,k}^{n(l)}}$ can be described as

$$\frac{\partial E}{\partial A_{m,k}^{n(l)}} = \frac{\partial E}{\partial \alpha_m^{(l)}} \frac{\partial \alpha_m^{(l)}}{\partial A_{m,k}^{n(l)}} + \frac{\partial E}{\partial p_{m,k}^{n(l+1)}} \frac{\partial p_{m,k}^{n(l+1)}}{\partial A_{m,k}^{n(l)}}. \tag{35}$$

In this layer, the method to update the learned weights $g_{m,k}^{n(l)}$ and $\sigma_{m,k}^{2(l)}$ is the same as (30).

The network training stage is a backward propagation process, and its process is summarized in Algorithm 2. In particular, it first derives executed action $\{A_{m,k}^n, p_{m,k}^n\}$. After executing the action, we get the reward r_{t+1} and current loss function value E. If the loss function value E is bigger than the threshold ζ, the network weights $\theta = \{g_{m,k}^{n(l)}, \sigma_{m,k}^{2(l)}\}_{l=1}^{L+1}$ are updated along the gradient descent direction of loss function, and then the network learning stage repeats. When the loss function value E is smaller than the threshold ζ or the iteration number reaches the pre-defined maximum iteration number T, the CIAQ algorithm terminates, thereby resulting in the final resource allocation results.

Algorithm 2: Network Training Stage of CIAQ Algorithm

1: Set discount factor γ, threshold ζ and maximum iteration number T;
2: **For** $t = 0, \ldots, T$
3: Execute the action $\{A_{m,k}^n, p_{m,k}^n\}$ generated in Algorithm 1;
4: Update the new state D_{t+1} and the obtained reward r_{t+1};
5: Calculate loss function $E = \frac{1}{2}[r_{t+1} + \gamma \cdot \max Q_t(D_{t+1}, A_{m,k}^{n(')}, p_{m,k}^{n(')}) - Q_t(D_t, A_{m,k}^n, p_{m,k}^n)]^2$, where $Q_t(D_t, A_{m,k}^n, p_{m,k}^n)$ and $\max Q_t(D_{t+1}, A_{m,k}^{n(')}, p_{m,k}^{n(')})$ are estimated by the designed DNN-based optimization framework;
6: **If** $E \geq \zeta$
7: Update the weights of the DNN-based optimization framework $\theta = \{g_{m,k}^{n(l)}, \sigma_{m,k}^{2(l)}\}_{l=1}^{L+1}$ along the gradient descent direction of loss function in (28), (29), (32),(33), (34) and (35), and then go into Algorithm 1;
8: **Else**
9: Output current $\{A_{m,k}^n, p_{m,k}^n\}$ as the resource allocation result;
10: **End If**
11: **End For**

As suggested by [27], in our paper the complexity involved can be reflected by the total number of optimization variables required to be calculated. Specifically, in the network learning stage shown in Algorithm 1, the spectrum status neuron $A_{m,k}^n$, the transmit power neuron $p_{m,k}^n$ and the multiplier neuron α_m are calculated along the data flow of the DNN-based optimization framework. If we assume that the data flow has l iterations, both $A_{m,k}^n$ and $p_{m,k}^n$ need to calculate $l*M*K*N$ times, while α_m needs to calculate $l*M$ times. Therefore, the computational complexity of the network learning stage can be expressed as $\mathcal{O}(l * M * K * N)$. After obtaining the resource allocation result, the loss function described in (27) is calculated. Then, it can be confirmed whether the network training stage and a new network learning stage should be executed according to the value of loss function. In the network training stage shown in Algorithm 2, the network weights $g_{m,k}^n$ and $\sigma_{m,k}^2$ in each strategy generation layer and the decision output layer are updated by adopting the gradient backpropagation of the loss

function. Because both $A^n_{m,k}$ and $p^n_{m,k}$ contain $g^n_{m,k}$ and $\sigma^2_{m,k}$, $g^n_{m,k}$ needs to update $2*l*M*K*N$ times, while $\sigma^2_{m,k}$ needs to update $2*l*M*K$ times. Hence, the computational complexity of the network training stage can be also described as $\mathcal{O}(l*M*K*N)$. When assuming cyclic network learning and network training stages need t times, the computational complexity of the proposed CIAQ algorithm is given by $\mathcal{O}(t*l*M*K*N)$.

Further, we propose a distributed multi-agent resource allocation (DMARA) scheme as an enhanced solution in the scenario with the existence of hundreds and even thousands of users. In the proposed DMARA scheme, the small cells are first classified into clusters according to network size and latency constraints. In particular, each cluster-head is the agent which tries to learn the near-optimal strategy by adopting the CIAQ algorithm. Since each cluster makes its decisions only with limited amount of CSI within the management scope, which therefore incurs very small amount of transmission overhead, and is more autonomous and robust. In addition, as shown in Fig. 3, in consideration of the network dynamic, each agent is modeled as a discrete-time event system, driven by the new user arrival events. The operation time line is divided into W time instants. The agent collects information and makes decisions at the beginning of each time instant. If the network environment changes quickly, the cluster size and time instant can be set smaller, so that each agent manages fewer users and the algorithm updates faster. Recently, there are several prior works that adopt multi-agent Q-learning based resource allocation approach to maximize the system performance, such as the self-organizing resource allocation strategy [38]. How to exploit these new technologies to further enhance dynamic and scalability of the proposed algorithm will be a future direction.

Above, we design and train the DNN-based optimization framework by the theory of the model-driven deep reinforcement learning. With the knowledge of a very limited amount of CSI, the proposed CIAQ algorithm solves the multi-objective optimization problem self-adaptively. Each agent can dynamically adjust the management scope and operation time instant according to the dynamic of network environment, and can adopt the trial-and-error mechanism to dynamic real-time interact with the environment to autonomously train the results. In Section V, let us evaluate and analyze the performance of the designed CIAQ algorithm in detail.

V. Simulation Results

In this section, we evaluate the performance of the proposed CIAQ algorithm from various simulation perspectives. We assume there are three different setups for the UDNs: a) Small UDN with 3 small-cell BSs, each of which randomly has 4 users supported by 5 orthogonal subcarriers; b) Medium-size UDN with 9 small-cell BSs, each of which randomly has 10 users supported by 15 orthogonal subcarriers; c) Large UDN with 18 small-cell BSs, each of which randomly has 20 users supported by 25 orthogonal subcarriers. Suppose that, each small-cell BS is located at the center of a round-shape cell with a radius of 200 m, and the total power limit P^{max}_m of BS m on all frequency bands is constrained to be 38 dBm. In addition, let us set penalty parameter $\mu = 2$, the threshold $\xi_A = \xi_p = 0.01$ and $\zeta = 0.001$.

A. Effects of discount factor

In this subsection, we analyze how the discount factor γ affects the network performance in the context of the medium-size UDN. Fig. 5 investigates the SE, EE and fairness of the UDN under different discount factors, which vary from 0 to 1. In the figure, it clearly observes that when γ approaches 0, the values of the EE and fairness increase, while the value of the SE gets smaller. This observation can perfectly validate the analytical results of the loss function given in (27), where the discount factor γ determines the extent of optimization objective. Further, after normalizing the EE, SE and fairness, the curve of the SE intersects with the curves of the EE and fairness, respectively. This is because that, when increasing the discount factor γ, the normalization value of SE increases monotonically, while the normalization values of EE and fairness decrease monotonically, which reveals that the intersection points indicate the optimization objectives represented by the cross curve have the same weight to characterize the resource allocation result. From the above observations, we imply that the discount factor γ should be set to an appropriate value according to the optimization goal, specifically, the values of γ corresponding to the intersection points can be used to well characterize the trade-off between the SE, EE and fairness. We can determine the extent of optimization objective by adjusting the discount factor in order to effectively solve the multi-objective optimization problem.

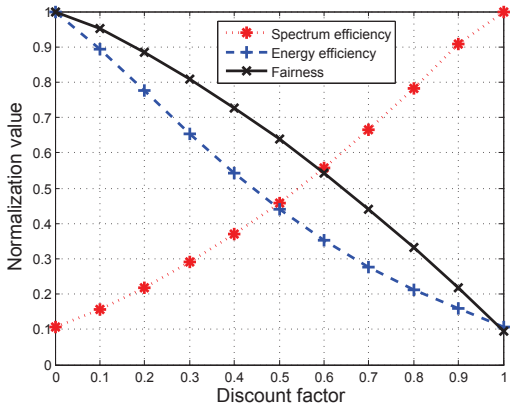


Fig. 5. The effects of discount factor γ on SE, EE and fairness. SE, EE and fairness are respectively normalized. Note that, to facilitate the analysis of intersection points, the normalization value of fairness is calculated as following: $fairness_{normalization} = 1 - \frac{fairness_{current} - fairness_{min}}{fairness_{max} - fairness_{min}}$, where $fairness_{current}$, $fairness_{max}$ and $fairness_{min}$ are the current fairness, the maximum value, and the minimum value of the obtained fairness, respectively.

B. Convergence and computational complexity analysis

In this subsection, we first evaluate the convergence performance of the CIAQ algorithm by varying the DNN-based optimization framework depth (i.e., the number of multiple repetitive iterations in the data flow) and the algorithm iterative times (i.e., the number of cyclic network learning and training stages), and then we compare the computational time of the proposed algorithm with that of the traditional data-driven deep reinforcement learning (DDRL) algorithm [20]. The traditional DDRL algorithm makes the neural network as a black box and adopts random initialization method to initialize the network parameters.

Fig. 6 shows how the depth of the DNN-based optimization framework affects the algorithm performance in terms of the spectrum status $A^n_{m,k}$ and the transmit power $p^n_{m,k}$. Note that, it evaluates the performance of three different UDNs by setting the discount factor $\gamma = 0.6$ and adopts the model-based initialization method. Here, when the difference values $\left|A^{n(L+1)}_{m,k} - A^{n(L)}_{m,k}\right|$ and $\left|p^{n(L+1)}_{m,k} - p^{n(L)}_{m,k}\right|$ are both below the pre-defined thresholds, the DNN-based optimization framework outputs the resource allocation results. In Fig. 6, it shows that the data flow of the optimization framework converges very quickly, in particular, within three or four iterations. This is because our optimization framework consists of a series of ADMM iterative procedures, which features a fast convergence rate. Further, the data flow needs more number of iterations to converge as the size of UDNs increases. However, the number of iterations in large UDN is 12. Therefore, the above observations imply that, the DNN-based optimization framework depth is within our acceptable range, which can significantly enhance the performance of our algorithm.

Fig. 7 evaluates the convergence performance of CIAQ algorithm in terms of the loss function value. In this figure, it considers three different UDNs with the model-based initialization method and sets the discount factor $\gamma = 0.6$. Observed from the figure, the proposed CIAQ algorithm converges within the relatively small number of iterations for the UDNs under different setups. Specially, in the beginning stages, the loss function value of the algorithm presents dramatical decreasing. The reason behind the above observation lies in the fact that, the gradient descent approach is employed by the CIAQ algorithm to train the DNN-based optimization framework, thereby the loss function converges faster in the beginning while becoming flat varying near the minimum point. Further, the figure shows that, when the size of UDN increases, the algorithm will demand more number of iterations to converge. This is simply because the implementation complexity increases as the network size gets larger. In general, the above observations indicate that the proposed CIAQ algorithm has good convergence performance, which will stimulate its practical application.

Fig. 8 compares the average computational time of the CIAQ algorithm with that of the traditional DDRL algorithm running on a GPU (the Nvidia Titan X Pascal) or a CPU (Intel i7-7700HQ, 2.80G Hz, RAM 16.0G Bytes) when varying the size of UDN. Observed from the figure, the computational time of the CIAQ algorithm is obviously less than that of the traditional DDRL algorithm. This is simply because that, the designed DNN-based optimization framework has theoretical explanation, and the network parameters are initialized based on the characteristics of resource allocation in UDNs, which greatly reduces the network training time. Further, the logarithm value of the computational time of the GPU-based algorithm is more than 2 smaller than that of the CPU-based one. This indicates that the GPU-based algorithm runs more than 100 times faster than the CPU-based one. In addition, in the context of the large-size UDN, the computational time of the CPU-based CIAQ algorithm is about 46 seconds, while that of the GPU-based one is only 0.08 second, which is acceptable according to the constraint on the UDNs in practical systems. The observations also imply that, we need to achieve a tradeoff between the computation capacity and the computation time.

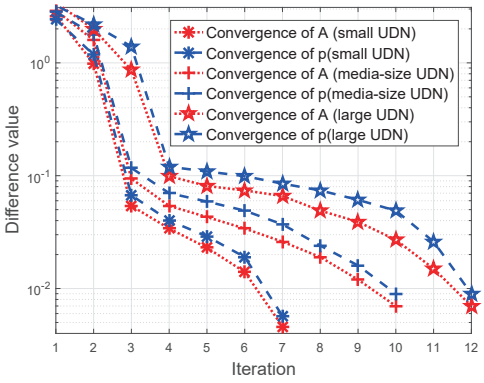


Fig. 6. The effects of different DNN-based optimization framework depth (l) on the algorithm performance.

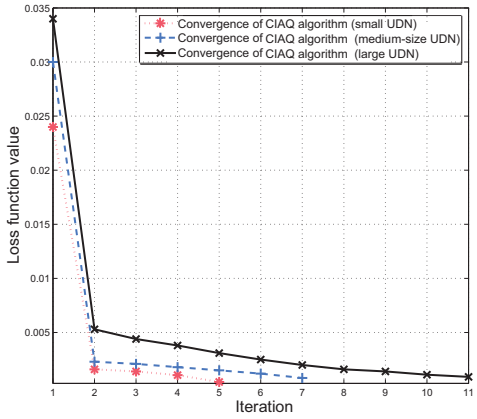


Fig. 7. The effects of different algorithm iterative times (t) on the algorithm performance.

C. Effects of Different Network Initialization

Table I compares the performance of model-based initialization and random initialization when considering the medium-size UDN. Their cases are considered corresponding to the discount factor being 0.1, 0.5 and 0.9, respectively. In general, we can observe that, the model-based initialization can always obtain better performance than the random initialization approach in terms of the SE, EE and fairness. Especially, the computational rate of model-based initialization is much faster than that of random initialization. This is because the model-based initialization method initializes the weights based on the characteristics of resource allocation in UDNs, which improves the optimization performance and computational rate. For the first case of $\gamma = 0.1$, it finds that, the EE and fairness are the dominant factors of resource allocation. Whereas, for the third case of $\gamma = 0.9$, our resource allocation is more concerned with SE. The above observations clearly validate the results of the loss function given in (27) and the simulation in Fig. 5. Furthermore, the results also imply that, regardless of the model-based initialization or the random initialization employed, a proper value of the discount factor γ should be chosen, which may significantly affect the resource allocation performance.

TABLE I
THE RESULT COMPARISONS OF DIFFERENT INITIALIZATIONS.

Optimal Objective	Model-based $\gamma = 0.1$	Random $\gamma = 0.1$	Model-based $\gamma = 0.5$	Random $\gamma = 0.5$	Model-based $\gamma = 0.9$	Random $\gamma = 0.9$
SE(bit/s/Hz)	5.1277	5.0664	18.3093	17.1906	31.4501	29.5775
EE(Mbit/s/W)	6.2825	5.2038	4.8056	3.5127	2.9791	2.0647
Fairness((Mbit/s)^2)	0.00017	0.00032	0.00233	0.00248	0.00474	0.00487
Computational time(s)	0.0123	74.445	0.0127	78.124	0.0121	71.226

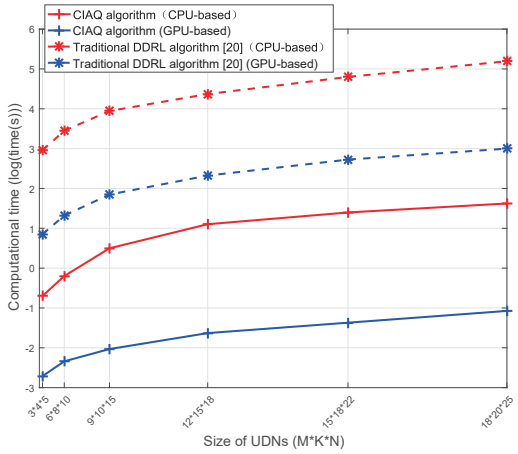


Fig. 8. The computational time versus size of UDNs on the chosen CPU and GPU-based CIAQ algorithm or traditional DDRL algorithm.

Furthermore, Fig. 9 compares the effects of model-based initialization and random initialization on DNN-based optimization framework depth when considering the large-size UDN. The discount factor γ is set to 0.5. We simulate the DNN-based optimization framework depth in terms of the difference values $\left|A_{m,k}^{n(L+1)} - A_{m,k}^{n(L)}\right|$ and $\left|p_{m,k}^{n(L+1)} - p_{m,k}^{n(L)}\right|$. We have the following observations when comparing Fig. 9 with Fig. 6. First, when employing model-based initialization, the DNN-based optimization framework depth in Fig. 9 is a little deeper than that in Fig. 6. This is because as the size of the UDN gets bigger, the CIAQ algorithm demands more complexity to implement and requires more iterations to converge. Second, in Fig. 9, it is worth noting that the DNN-based optimization framework depth with model-based initialization is obviously shorter than that with random initialization. From the above observation, we conclude that the performance of the DNN-based optimization framework obtained by the model-based initialization is better than that obtained by the random initialization.

Fig. 10 investigates the convergence performance of the CIAQ algorithm when employing the model-based initialization and random initialization approaches, where the three different UDNs setups are considered. The discount factor γ is set to 0.7. When using the model-based initialization, the observations in Fig. 10 are similar to those in Fig. 7, and the similar conclusions can be derived. In addition, it is also seen that, in the first iteration, the loss function values with model-based initialization in three different UDNs are all obviously smaller than those with random initialization. For all the UDN setups, the algorithm with model-based initialization converges within a relatively small iteration, such that $\leq$12. By contrast, the random initialization approach demands much higher number of iterations to converge, which is bigger than 20. According to the observations in this subsection, we conclude that, the performance of the algorithm obtained by the model initialization approach is obviously better than that of the random initialization approach, which efficiently improves the performance of the designed DNN-based optimization framework and the CIAQ algorithm.

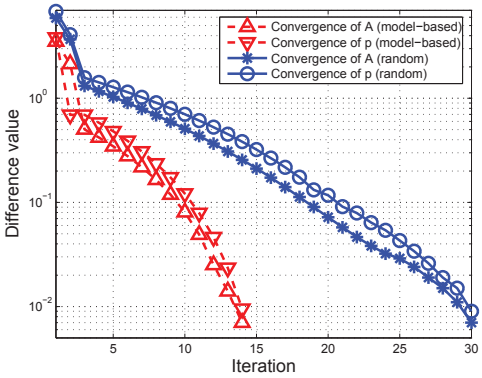


Fig. 9. The effects of different initializations on DNN-based optimization framework depth.

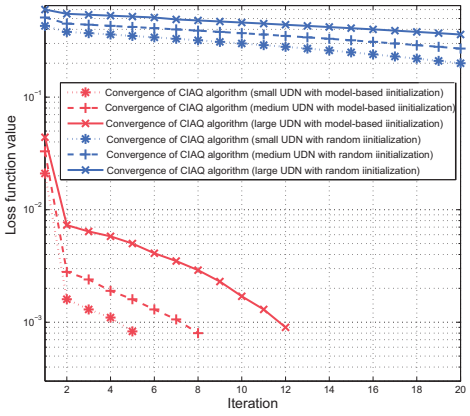
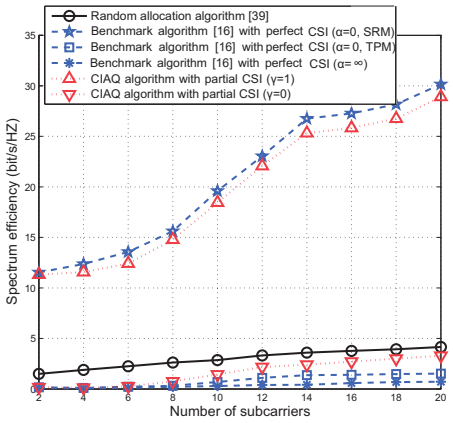


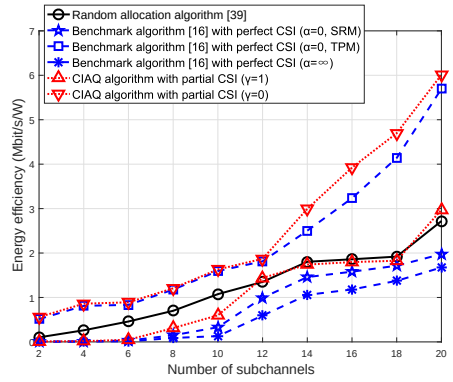
Fig. 10. The effects of different initializations on algorithm iterative times.

D. Comparison

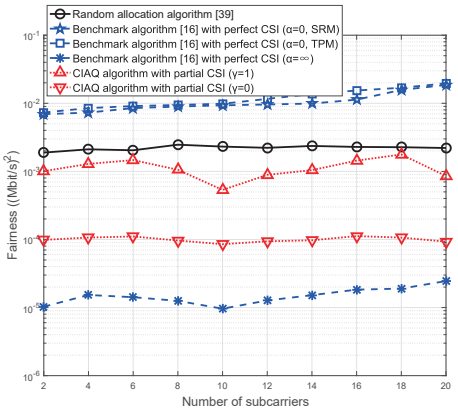
Finally, we provide the numerical results to compare the SE, EE and fairness performance of our proposed CIAQ algorithm, with those of the existing algorithms including the random allocation algorithm [39] and the benchmark algorithm [16]. Random allocation algorithm adopts the random variables with normal distribution to represent the probability of resource allocation. The benchmark algorithm transforms the NP-hard nonconvex optimization problem into five cases of the fairness index α. We simulate the weighted sum rate maximization (SRM) problem or the weighted total power minimization (TPM) problem when $\alpha = 0$, and the fairness maximization problem



(a) Spectrum efficiency comparison

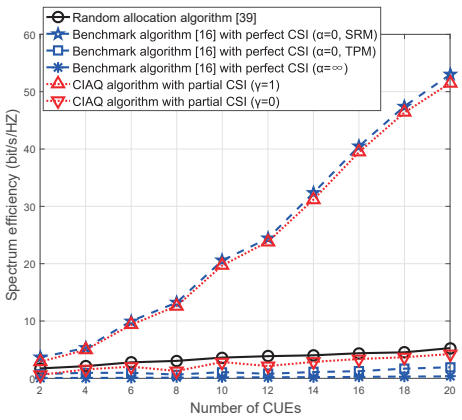


(b) Energy efficiency comparison

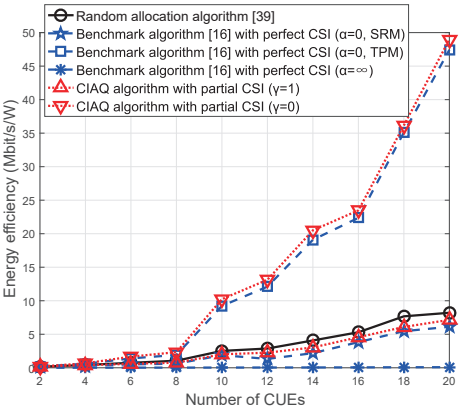


(c) Fairness comparison

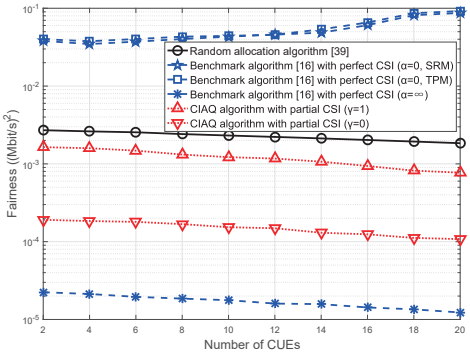
Fig. 11. Algorithm performance comparison versus the number of subcarriers.



(a) Spectrum efficiency comparison



(b) Energy efficiency comparison



(c) Fairness comparison

Fig. 12. Algorithm performance comparison versus the number of cellular users.

when $\alpha = \infty$. Note that, the benchmark algorithm needs to have perfect CSI available at the central controller, while the proposed CIAQ algorithm only demands a very limited amount of CSI.

Fig. 11 compares the performance of CIAQ algorithm with the other existing algorithms when varying the number of subcarriers in the context of the medium-size UDN. In Fig. 11(a), it clearly observes that the benchmark algorithm with SRM under $\alpha = 0$ achieves the highest SE results, which is slightly better than the CIAQ algorithm under $\gamma = 1$. However, this SE performance gain is obtained by sacrificing big losses in the EE and fairness. Further, we should note that, for this case, the CIAQ algorithm devotes a big weight on the SE. In addition, different from the benchmark algorithm, our CIAQ algorithm is able to take care of the EE and fairness while maximizing the SE performance. By contrast, in Fig. 11(b) and Fig. 11(c), the EE and fairness of the CIAQ algorithm under $\gamma = 1$ are obviously better than those achieved by the benchmark algorithm with SRM under $\alpha = 0$. In particular, in Fig. 11(c), the fairness of the CIAQ algorithm under $\gamma = 1$ is better when the number of subcarriers is about 10. This observation shows that the best fairness performance is derived when each user is allocated similar number of subcarriers. For the case when $\gamma = 0$, the CIAQ algorithm yields higher SE, EE and fairness than the benchmark algorithm with TPM under $\alpha = 0$.

This is reasonable since the CIAQ algorithm under $\gamma = 0$ is concerned with rewards known as EE and fairness, while the objective of the benchmark algorithm with TPM under $\alpha = 0$ is to minimize the global power consumption. Moreover, in the benchmark algorithm under $\alpha = \infty$, the highest fairness is sacrified by the lowest SE and EE. Meanwhile, both the SE and fairness of the CIAQ algorithm under $\gamma = 1$ are higher than those of the random algorithm by sacrificing acceptable amount of EE. Further, although the CIAQ algorithm under $\gamma = 0$ achieves a little smaller SE than the random algorithm, the CIAQ algorithm can achieve much bigger performance gains in terms of the EE and fairness over the random algorithm. From the above observations, we can conclude that the CIAQ algorithm not only solves the resource allocation problem with partial CSI, but also is significantly superior to the random algorithm and the benchmark algorithm in term of the tradeoff among the SE, EE and fairness performance of the UDN.

Further, Fig. 12 compares the performance of CIAQ algorithm, with that of the other two algorithms when varying the number of cellular users (CUEs) in the context of the large-size UDN. Seen from Fig. 12, for the case of $\gamma = 1$, the SE of CIAQ algorithm is close to that of the benchmark algorithm with SRM under $\alpha = 0$, while both the EE and fairness of CIAQ algorithm are better than those of the benchmark algorithm. Meanwhile, the SE, EE and fairness of the CIAQ algorithm under $\gamma = 0$ are obviously superior to the benchmark algorithm with TPM under $\alpha = 0$. In addition, the highest fairness of the benchmark algorithm under $\gamma = \infty$ is achieved by minimizing the SE and EE. Further, although the EE of the CIAQ algorithm under $\gamma = 1$ is sightly lower than that of the random algorithm, both the SE and fairness derived by the CIAQ algorithm are better than those achieved by the random algorithm. Moreover, when $\gamma = 0$, our CIAQ algorithm yields higher EE and fairness than the random algorithm by sacrificing acceptable amount of SE performance. Therefore, the above observations demonstrate that the CIAQ algorithm outperforms the other two existing resource allocation algorithms when solving multi-objective optimization problem.

VI. Conclusions

We have proposed and developed a novel model-driven deep reinforcement learning assisted resource allocation for future UDNs in the presence of very limited amount of CSI. The analytical scheme for resource allocation in UDNs has been derived by solving the multi-objective optimization problem that achieves the tradeoff among the SE, EE and fairness. To solve this NP-hard nonconvex problem, we have designed a DNN-based optimization framework consisting of the ADMM iterative procedures. Then the CIAQ algorithm has been proposed, which takes CSI as the unknown weights and trains the designed DNN-based optimization framework without massive labeling data. Our simulation results have shown that, the proposed CIAQ algorithm with rapid convergence ability not only well characterizes the extent of optimization objective by adjusting discount factor, but also significantly outperforms the current random initialization method of neural network and the other existing resource allocation algorithms. Furthermore, according to the performance evaluation, we conclude that our proposed model-driven deep reinforcement learning assisted resource allocation is a promising candidate for future UDNs.

References

[1] M. Jaber, F. J. Lopez-Martinez, M. A. Imran, A. Sutton, A. Tukmanov, and R. Tafazolli, "Wireless backhaul: Performance modeling and impact on user association for 5g," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3095–3110, May. 2018.

[2] P. Patcharamaneepakorn, S. Wu, C. Wang, e. M. Aggoune, M. M. Alwakeel, X. Ge, and M. D. Renzo, "Spectral, energy, and economic efficiency of 5g multicell massive mimo systems with generalized spatial modulation," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9715–9731, Dec. 2016.

[3] C. Pan, H. Ren, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Weighted sum-rate maximization for the ultra-dense user-centric tdd c-ran downlink relying on imperfect csi," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1182–1198, Feb. 2019.

[4] G. Zhang, H. Zhang, Z. Han, and G. K. Karagiannidis, "Spectrum allocation and power control in full-duplex ultra-dense heterogeneous networks," *IEEE Transactions on Communications*, early access, 2019.

[5] S. Feng, R. Zhang, W. Xu, and L. Hanzo, "Multiple access design for ultra-dense vlc networks: Orthogonal vs non-orthogonal," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2218–2232, Mar. 2019.

[6] J. Tang, D. K. C. So, E. Alsusa, K. A. Hamdi, A. Shojaeifard, and K. Wong, "Energy-efficient heterogeneous cellular networks with spectrum underlay and overlay access," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2439–2453, Mar. 2018.

[7] H. Zhang, H. Liu, J. Cheng, and V. C. M. Leung, "Downlink energy efficiency of power allocation and wireless backhaul bandwidth allocation in heterogeneous small cell networks," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1705–1716, Apr. 2018.

[8] N. Mokari, F. Alavi, S. Parsaeefard, and T. Le-Ngoc, "Limited-feedback resource allocation in heterogeneous cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2509–2521, Apr. 2016.

[9] H. Tang and Z. Ding, "Mixed mode transmission and resource allocation for d2d communication," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 162–175, Jan. 2016.

[10] H. Zhang, L. Song, and Y. J. Zhang, "Load balancing for 5g ultra-dense networks using device-to-device communications," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4039–4050, Jun. 2018.

[11] J. Peng, J. Zeng, X. Su, B. Liu, and H. Zhao, "A qos-based cross-tier cooperation resource allocation scheme over ultra-dense hetnets," *IEEE Access*, vol. 7, pp. 27 086–27 096, 2019.

[12] N. Trabelsi, C. S. Chen, R. E. Azouzi, L. Roullet, and E. Altman, "User association and resource allocation optimization in lte cellular networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 429–440, Jun. 2017.

[13] T. Yang, R. Zhang, X. Cheng, and L. Yang, "Graph coloring based resource sharing (gcrs) scheme for d2d communications underlaying full-duplex cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7506–7517, Aug. 2017.

[14] L. Liang, W. Wang, Y. Jia, and S. Fu, "A cluster-based energy-efficient resource management scheme for ultra-dense networks," *IEEE Access*, vol. 4, pp. 6823–6832, 2016.

[15] Y. Lin, R. Zhang, C. Li, L. Yang, and L. Hanzo, "Graph-based joint user-centric overlapped clustering and resource allocation in ultradense networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4440–4453, May. 2018.

[16] Q. Pham and W. Hwang, "Fairness-aware spectral and energy efficiency in spectrum-sharing wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 207–10 219, Nov. 2017.

[17] L. Su, C. Yang, and C. I, "Energy and spectral efficient frequency reuse of ultra dense networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 8, pp. 5384–5398, Aug. 2016.

[18] W. Zhao and S. Wang, "Resource sharing scheme for device-to-device communication underlaying cellular networks," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4838–4848, Dec. 2015.

[19] H. Takshi, G. Doğan, and H. Arslan, "Joint optimization of device to device resource and power allocation based on genetic algorithm," *IEEE Access*, vol. 6, pp. 21 173–21 183, 2018.

[20] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, April 2019.

[21] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric iot," *IEEE Transactions on Emerging Topics in Computing*, early access, 2019.

[22] S. Liu, X. Hu, and W. Wang, "Deep reinforcement learning based dynamic channel allocation algorithm in multibeam satellite systems," *IEEE Access*, vol. 6, pp. 15 733–15 742, 2018.

[23] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 60–69, Nov. 2017.

[24] H. Li, H. Gao, T. Lv, and Y. Lu, "Deep q-learning based dynamic resource allocation for self-powered ultra-dense networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May. 2018, pp. 1–6.

[25] Y. Zhou, Z. M. Fadlullah, B. Mao, and N. Kato, "A deep-learning-based radio resource assignment technique for 5g ultra dense networks," *IEEE Network*, vol. 32, no. 6, pp. 28–34, November 2018.

[26] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y. Liang, "Intelligent resource scheduling for 5g radio access network slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7691–7703, Aug 2019.

[27] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, Nov 2017.

[28] Z. Xu and J. Sun, "Model-driven deep-learning," *National Science Review*, vol. 5, no. 1, pp. 22–24, 2018. [Online]. Available: http://dx.doi.org/10.1093/nsr/nwx099

[29] Y. Yang, J. Sun, H. LI, and Z. Xu, "Admm-csnet: A deep learning approach for image compressive sensing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, early access, 2018.

[30] C. Z. Fa-long Luo, *Signal Processing for 5G: Algorithms and Implementations*, 1st ed. USA: Wiley-IEEE Press, 2016.

[31] W. C. Ao and K. Psounis, "Approximation algorithms for online user association in multi-tier multi-cell mobile networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2361–2374, Aug. 2017.

[32] H. SHI, R. V. Prasad, E. Onur, and I. G. M. M. Niemegeers, "Fairness in wireless networks:issues, measures and challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 5–24, First Quarter 2014.

[33] W. Zeng, Y. R. Zheng, and R. Schober, "Online resource allocation for energy harvesting downlink multiuser systems: Precoding with modulation, coding rate, and subchannel selection," *IEEE Transactions on Wireless Communications*, vol. 14, no. 10, pp. 5780–5794, Oct. 2015.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2017.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*, seven ed. UK: Cambridge University Press, 2009.

[36] H. Tabrizi, B. Peleato, G. Farhadi, J. M. Cioffi, and G. Aldabbagh, "Spatial reuse in dense wireless areas: A cross-layer optimization approach via admm," *IEEE Transactions on Wireless Communications*, vol. 14, no. 12, pp. 7083–7095, Dec. 2015.

[37] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural Network Design*, 2nd ed. USA: Martin Hagan, 2014.

[38] M. Chen, Y. Hua, X. Gu, S. Nie, and Z. Fan, "A self-organizing resource allocation strategy based on q-learning approach in ultra-dense networks," in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Sep. 2016, pp. 155–160.

[39] Z. Kbah and A. Abdelhadi, "Resource allocation in cellular systems for applications with random parameters," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2016, pp. 1–5.