

Computation Offloading and Resource Allocation in C-RAN Supporting Wireless Charging

Ruijing Li¹, Lan Tang¹, *Member, IEEE*, Yechao Bai¹, *Member, IEEE*, Mengting Lou¹, and Xinggan Zhang¹

Abstract—In this article, we consider a cloud-radio access network (C-RAN) with multiple enhanced remote radio heads (RRHs), which support for wireless charging, computation offloading and information transmission. The employment of enhanced RRHs requires a redesign for efficient resource allocation. Our goal is to minimize the weighted sum of energy transfer time and computation time by jointly optimizing energy transmission time, energy beamforming vectors, offloading factors, information transmission time, and combining vectors, while satisfying the communication performance requirements of terminal nodes. The problem is formulated as a mixed-integer nonlinear programming, which can be solved by alternating optimization (AO) utilizing semidefinite relaxation (SDR) and nonlinear transformation. To reduce the online computation delay of AO algorithm, we propose an optimization method based on the deep learning framework, which can determine an optimal solution with a trained neural network in real-time application. To improve the performance of the learning algorithm, we use two methods to set labels for training deep neural network to obtain the optimal offloading decision: 1) preoptimization and 2) self-evolution. Simulation results show that with the trained neural network, the algorithms based on deep learning can achieve a similar performance to the AO algorithm with less computation time.

Index Terms—Alternating optimization (AO), computation offloading, deep neural network, resource allocation, semidefinite relaxation (SDR).

I. INTRODUCTION

IN RECENT years, the advancement of Internet of Things (IoT) has motivated various new applications, such as drones, smart home, autonomous driving, and so on [1]. Some IoT applications are latency-sensitive so that they need the real-time communication and computation of massive wireless terminals. In an IoT network, massive number of wireless terminals capable of computation and communication are deployed. Whereas these terminals always suffer from finite device lifetime and low computing capability. For example, most sensors are powered by batteries and they will not work when batteries run out.

Manuscript received 19 August 2022; revised 24 November 2022; accepted 23 December 2022. Date of publication 28 December 2022; date of current version 9 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62072229, Grant U1936201, Grant 62071220, and Grant 61976113; and in part by the Joint Project of China Mobile Research Institute & X-NET. (*Corresponding author: Lan Tang.*)

Ruijing Li, Lan Tang, Yechao Bai, and Xinggan Zhang are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China (e-mail: tanglan@nju.edu.cn).

Mengting Lou is with the Future Research Laboratory, China Mobile Research Institute, Beijing 100053, China.

Digital Object Identifier 10.1109/JIOT.2022.3232629

Hence, how to resolve the two crucial performance limitations is a critical yet challenging task.

To resolve the aforementioned finite device lifetime issue, wireless energy transfer (WET) has been introduced to IoT networks to charge low-power terminals. In the wireless-powered IoT (WP-IoT) networks, terminals are powered by radio-frequency (RF) energy broadcasted by communication base stations (BSs) or dedicated power beacons. In WP-IoT with one hybrid access point (H-AP) and multiple terminals, terminals harvest energy from downlink RF signals and then the energy is utilized for the uplink information transmission or local computation. Therefore, the H-AP needs to coordinate wireless energy and information transmission by appropriate resource allocation. To maximize the sum-throughput, Ju and Zhang [2] proposed to jointly optimize the downlink energy transfer time and uplink information transmission time. In [3], total transmission time including energy transfer time and data transmission time was minimized under the uplink information transmission constraint. By assuming that terminals can harvest energy from RF signals transmitted by BS or other terminals, Yang et al. [4] maximized the network utility related to sum rate of uplink and downlink via time and power allocation. Wu et al. [5] considered a wireless-powered communication network with separately deployed power station and information receiving station, in which the energy efficiency was maximized by optimizing time allocation and transmit power. To maximize the average throughput of the wireless-powered communication network in fading channels, Hadzi-Velkov et al. [6] proposed two opportunistic scheduling policies and corresponding resource allocation schemes. In [7], the uplink sum rate was maximized by optimizing the energy and time allocation under the energy storage constraint and energy causality constraint. To reduce the control signaling overhead in centralized scheduling, Bi et al. [8] proposed a distributed scheduling protocol in which AP transferred energy when it received energy request buzz, otherwise wireless devices contended to transmit data using Carrier Sensing Multiple Access.

Limited energy supply and computing power make it difficult for low-power nodes to deal with complicated computing tasks. Mobile-edge computing (MEC) can assist in computing by offloading some computing tasks of nodes to nearby servers. For example, MEC can satisfy the requirements of throughput, and, especially, latency, in vehicle-to-everything (V2X) scenarios. By deploying MEC servers within the radio access network (RAN), agile services associated with the vehicles can be provided. Reinforcement learning recently has

been applied to MEC to solve the task offloading problem allowing for both the task execution time and energy consumption [9]. In [10], a deep reinforcement-learning-based online offloading (DROO) framework was proposed to maximize the weighted sum of the computation rates of all nodes by optimizing the offloading decision. The integration of WPT and MEC technologies provides a promising way to prolong the lifetime and improve the computational efficiency of low-power terminals. Bi and Zhang [11] maximized the (weighted) sum computation rate of all terminals in the wireless-powered network by jointly optimizing the individual computing mode selection (i.e., local computing or offloading) and the system transmission time allocation. In wireless-powered MEC, a collaborative offloading mechanism was proposed in [12] to eliminate double-near-far effect with the relay protocol. You et al. [13] considered a single-user wireless-powered MEC with binary offloading, where the probability of successful computation is maximized under the latency constraint.

In this article, we consider a cloud-RAN (C-RAN) supporting wireless charging in which multiple multi-antenna remote radio heads (RRHs) are deployed. We mainly concern several issues: energy transmission between RRHs and terminals, data transmission from terminals to central processor (CP) via multiple RRHs, and mode selection for computation offloading (local computing or offloading to RRH), so as to minimize the overall latency. The main contributions of this article are summarized as follows.

- 1) To minimize the weighted sum of energy transfer time and computation time, we propose an alternating optimization (AO) method based on SDR and nonlinear transformation to jointly optimize energy transmission time, energy beamforming vectors, offloading factors, information transmission time, and combining vectors under the constraints of consumed energy and bit transmission requirements of terminals. For given offloading factors, we obtain the optimal solution of other variables by semidefinite relaxation (SDR) and prove that the optimal solution is rank one solution. For a given solution of other variables, offloading factors are obtained by transforming the problem into an integer linear programming (LP) problem by nonlinear transformation.
- 2) In order to establish a mapping between network states (including channel states and battery powers) and offloading factors to reduce the complexity and computation latency in real-time operations, we propose an optimization method that combines deep learning and convex optimization. A neural network is used to represent the mapping relationship between network states and computation task offloading factors, and convex optimization is used to solve the other transmit parameters. To improve the learning speed, we use results of the AO algorithm as the training labels. Besides, a self-evolution method is proposed to obtain labels from historical optimization results. In practical systems, we can utilize the trained neural network to quickly determine the offloading factors and get the optimal allocation results.

- 3) Finally, we simulate and verify the above AO method and two deep-learning-based optimization algorithms, respectively, and compared the performance of the overall latency. With the offline-trained deep neural network, two deep-learning-based algorithms achieve similar performance to the AO algorithm, which means that the DNN-based algorithms have good generalization. We also analyze the influence of energy consumption constraints, information bit transmission constraints, computation delay constraints, transmission power, the number of RRHs, and the number of task computation terminals on the optimization results. The analysis results reveal that since too many terminals offload computing tasks to RRHs will reduce the computing efficiency of RRHs, there should be an optimal offloading ratio to ensure the shortest total computation latency. To decrease the energy harvesting time, terminals with low residual power have greater priority to offload computing tasks. Furthermore, to satisfy the delay requirement of some time-sensitive terminals, the total time consumption would increase.

The remainder of this article is organized as follows. Section II reviews the related works. Section III includes the system model and the problem formulation. In Section IV, an AO algorithm is proposed to solve the resource allocation and computation offloading problem. In Section V, we propose two deep-learning-based optimization methods to reduce the complexity and delay of real-time optimization. Section VI presents some numerical results to evaluate the system performance. Finally, the primary conclusion is drawn in Section VII.

II. RELATED WORK

In IoT networks, different types of terminals have different Quality-of-Service (QoS) requirements. For example, some sensors need to send data back to the CP for data analysis and fusion and some sensors need energy to implement local computation or task offloading. C-RAN with distributed access and centralized processing is the appropriate network architecture to meet these needs. In wireless-powered C-RAN, terminals harvest energy from RF signals transmitted by RRHs. The distributed multi-RRHs structure can improve the charging efficiency by decreasing the average distance between RRHs and terminals when the locations of terminals are scattered. Moreover, by optimizing the energy beamforming in centralized way, the energy transfer efficiency can be further improved. Some existing work has investigated how to maximize the network utility by optimizing transmit parameters in wireless-powered C-RAN. A general system utility optimization framework was proposed in [14] for the design of coordinated beamforming and power splitting algorithm based on the Lyapunov optimization theory in the energy harvesting powered C-RAN system, which adopted an online algorithm to maximize the system utility subject to energy sustainable constraints at the RRHs, signal-to-interference-plus-noise (SINR) and energy harvesting requirements of mobile devices. Li et al. [15] considered a green C-RAN with

simultaneous wireless information and power transfer ability and a semidefinite programming (SDP) problem was solved under the constraints of energy and information transmission requirements. In [16], it designed the beamforming vectors for data transmission and energy transfer, under finite fronthaul capacity of C-RAN.

In wireless-powered C-RAN, terminals that need to perform computing tasks can use the harvested energy to implement local computation or offload some tasks to appropriate RRHs. A wireless-powered sensors network assisted by edge computing was studied in [17], where BSs equipped with multiple antennas jointly served all sensors and the offloading strategy was optimized via a CP. Considering the independent variation of the wireless channel conditions and computing tasks, Teng et al. [18] proposed a mixed-timescale joint computational offloading and wireless resource allocation algorithm for latency-critical applications, aiming at minimizing the total energy consumption. In [19], it formulated the distributed computation offloading among mobile users as a multiuser computation offloading game, where cloud completed centralized processing by using the game theory to achieve a Nash equilibrium.

When there are different types of terminals in the network, resource needs to be properly allocated based on the overall network state to meet the needs of different terminals. Therefore, different from above work, in this article, we investigated how to satisfy Qos requirements of different terminals under the constraints of energy and computing power of RRHs in a C-RAN system supporting wireless charging.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

As shown in Fig. 1, we consider a C-RAN supporting wireless charging with single CP, M multiantenna RRHs, and K low-power and stationary sensor nodes. It is supposed that each RRH is equipped with N antennas, and each terminal has single antenna. We assume that each RRH is connected to the CP through optic fiber. The small-scale fading coefficients between terminals and RRHs obey Rayleigh distribution and the path loss in each link is a power falloff proportional to the squared distance. We consider the time-division duplexing (TDD) system, so the uplink and downlink are reciprocal. We define $\mathbf{g}_{k,m} = \alpha_{k,m}\bar{\mathbf{g}}_{k,m}$ as channel vector between RRH m and terminal k , where $\alpha_{k,m}$ accounts for large-scale path loss and $\bar{\mathbf{g}}_{k,m}$ denotes the small-scale Rayleigh fading.

We suppose that all of terminals have no fixed power supply and need to harvest energy from the received RF signals sent by RRHs. An energy storage device in the form of rechargeable battery or super-capacitor is equipped at each terminal to store the harvested energy for future use. The terminal nodes use the harvested energy to transmit information bits or finish the computation tasks.

Terminals are divided into two groups: ones in $\mathcal{K}_t$ transmit information bits to CP via RRHs and others in $\mathcal{K}_o$ will complete computation tasks. Due to the limited computational capacities, terminals in $\mathcal{K}_o$ can choose to offload computational tasks to some RRHs. The computational task of terminal k is

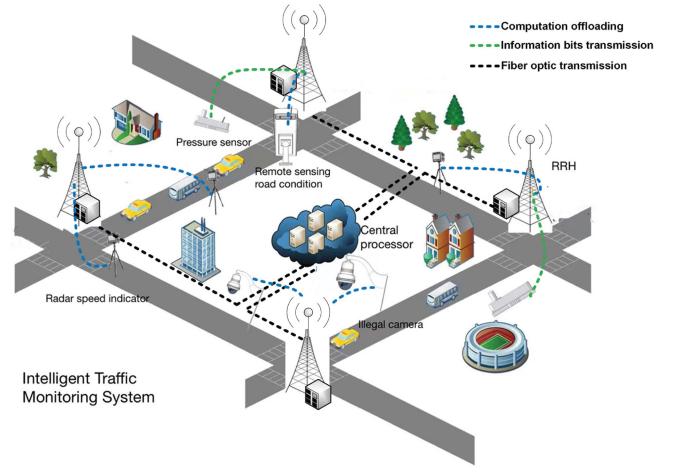


Fig. 1. C-RAN supporting wireless charging for information transmission and computation offloading.

represented by two parameters $\{R1_k, R2_k\}$, where $R1_k$ denotes the size of the task, and $R2_k$ denotes the computational power required for this task quantified by CPU cycles.

The whole transmission process includes two phases: energy transfer and data transmission.

1) *Energy Transfer*: During the energy transfer period, to improve the energy transfer efficiency, RRHs transfer energy to terminals by energy beamforming. The energy signal received by terminal k is

$$y_k = \sum_{m=1}^M \mathbf{g}_{k,m}^H \mathbf{v}_m s_m^{dl} \quad (1)$$

where s_m^{dl} with $E(|s_m^{dl}|^2) = 1$ and $\mathbf{v}_m$ are energy signals of RRH m and the energy beamforming vector, respectively. Assuming that transmitted energy signals transmitted by different RRHs are uncorrelated, the harvested energy of terminal k can be calculated as

$$E_k = \eta \tau_0 E[|y_k|^2] = \eta \tau_0 \sum_{m=1}^M |\mathbf{g}_{k,m}^H \mathbf{v}_m|^2 \quad (2)$$

where $0 < \eta < 1$ denotes the energy harvesting efficiency of the terminal and τ_0 is the energy transfer time.

2) *Data Transmission*: During the data transmission period between terminals and RRHs, terminals in $\mathcal{K}_t$ transmit information symbols to all RRHs simultaneously and some terminals in $\mathcal{K}_o$ offload computational tasks to RRHs via frequency-division multiple access (FDMA). To avoid interference, we assume that terminals in $\mathcal{K}_t$ and ones in $\mathcal{K}_o$ use different frequency bands. The signal received by the RRH m from terminals in $\mathcal{K}_t$ is

$$\mathbf{y}_m = \sum_{k=1}^{|\mathcal{K}_t|} \mathbf{g}_{k,m} s_k^{ul} + \mathbf{z}_m \quad (3)$$

where s_k^{ul} is the transmit signal of terminal $k \in \mathcal{K}_t$, and $|\mathcal{K}_t|$ denotes the number of terminals in $\mathcal{K}_t$ and $\mathbf{z}_m \sim \mathcal{CN}(\mathbf{0}, \phi^2 \mathbf{I})$ is the additive white Gaussian noise (AWGN) at RRH m . RRHs transmit the received information symbols to CP via optic fibers.

Assuming no loss in line transmission, the signals received by the CP can be expressed as

$$\mathbf{y} = [\mathbf{y}_1^T, \mathbf{y}_2^T, \dots, \mathbf{y}_M^T]^T = \sum_{k=1}^{|\mathcal{K}_t|} \mathbf{g}_k s_k^{ul} + \mathbf{z} \quad (4)$$

where $\mathbf{g}_k = [\mathbf{g}_{k,1}^T \dots \mathbf{g}_{k,M}^T]^T$ and $\mathbf{z} = [\mathbf{z}_1^T \dots \mathbf{z}_M^T]^T$. To decode s_k^{ul} , $\mathbf{y}$ is passed through a linear receiver to get

$$\begin{aligned} \hat{s}_k^{ul} &= \boldsymbol{\omega}_k^H \mathbf{y} \\ &= \boldsymbol{\omega}_k^H \mathbf{g}_k s_k^{ul} + \sum_{j \neq k} \boldsymbol{\omega}_k^H \mathbf{g}_j s_j^{ul} + \boldsymbol{\omega}_k^H \mathbf{z} \end{aligned} \quad (5)$$

where $\boldsymbol{\omega}_k \in \mathbb{C}^{MN \times 1}$ is the receive combining vector for estimating s_k^{ul} . According to (4), the SINR for s_k^{ul} is

$$\gamma_k = \frac{p_k^t |\boldsymbol{\omega}_k^H \mathbf{g}_k|^2}{\sum_{j \neq k} p_j^t |\boldsymbol{\omega}_k^H \mathbf{g}_j|^2 + \phi^2 \|\boldsymbol{\omega}_k^H\|^2} \quad (6)$$

where $p_k^t = E(|s_k^{ul}|^2)$. Assuming that the transmission time of terminals in $\mathcal{K}_t$ is τ_1 , the number of information bits transmitted by the terminal $k \in \mathcal{K}_t$ is

$$B_k = \tau_1 w_t \log(1 + \gamma_k) \quad (7)$$

where w_t is the bandwidth allocated to terminals in $\mathcal{K}_t$.

Terminals in $\mathcal{K}_o$ can choose to complete the computing task locally or upload the task to RRHs. The task offloading factor is denoted as $x_{k,m} \in \{0, 1\}$ ($0 \leq m \leq M$), where $x_{k,m} = 1$ ($1 \leq m \leq M$) means that the computing task of terminal k is offloaded to RRH server m , and $x_{k,0} = 1$ means that the task is computed locally. We assume that the computing task of terminal k can only be offloaded to one RRH, i.e., $\sum_{m=0}^M x_{k,m} = 1$. If the terminal $k \in \mathcal{K}_o$ chooses to upload the computation task to RRH m , it will use the harvested energy to transmit $R1_k$ bits to the RRH m . The transmission time is

$$t_{k,m}^{\text{off}} = \frac{R1_k \sum_{k'=1}^{|\mathcal{K}_o|} (1 - x_{k',0})}{w_o \log\left(1 + \frac{\|\mathbf{g}_{k,m}\|^2 p_k^o}{\phi^2}\right)} \quad (8)$$

where w_o is the channel bandwidth allocated to $\mathcal{K}_o$, and p_k^o is the transmit power of the terminal k in $\mathcal{K}_o$. The consumed energy for the transmission is

$$E_{k,m}^{\text{off}} = p_k^o t_{k,m}^{\text{off}} = \frac{p_k^o R1_k \sum_{k'=1}^{|\mathcal{K}_o|} (1 - x_{k',0})}{w_o \log\left(1 + \frac{\|\mathbf{g}_{k,m}\|^2 p_k^o}{\phi^2}\right)}. \quad (9)$$

Here, the transmission cost for returning the computation results to the terminal is ignored, because the amount of output data is usually much less than that of the input data.

Let $f_{k,m}^{\text{mec}}$ denote the computing resources of RRH m allocated to terminals in $\mathcal{K}_o$. It is determined by the number of terminals who offload computing tasks to this RRH, i.e., $f_{k,m}^{\text{mec}} = (F_m) / (\sum_{k=1}^{|\mathcal{K}_o|} x_{k,m})$, where F_m is the computational power of the RRH m . Thus, the computing time $t_{k,m}^{\text{mec}}$ for the task of terminal k offloaded to the RRH m is

$$t_{k,m}^{\text{mec}} = \frac{R2_k \sum_{k=1}^{|\mathcal{K}_o|} x_{k,m}}{F_m}. \quad (10)$$

The energy consumption for computation can be calculated as

$$E_{k,m}^{\text{mec}} = p_m^{\text{mec}} t_{k,m}^{\text{mec}} = \frac{p_m^{\text{mec}} R2_k \sum_{k=1}^{|\mathcal{K}_o|} x_{k,m}}{F_m} \quad (11)$$

where p_m^{mec} denotes the power consumed for computation in RRH m .

Denoting f_k^l and p_k^l as the local computing ability and power consumption of terminal k , respectively, the time and energy consumptions for local computation are $t_k^l = R2_k / f_k^l$ and $E_k^l = p_k^l R2_k / f_k^l$, respectively.

Based on the above analysis, no matter what decision the terminal in $\mathcal{K}_o$ makes, the total time required to complete the computing task is

$$\tau_{2,k} = \sum_{m=1}^M \left[x_{k,m} (t_{k,m}^{\text{off}} + t_{k,m}^{\text{mec}}) + (1 - x_{k,m}) t_k^l \right]. \quad (12)$$

The harvested energy E_k is stored in the battery of each terminal k . In the i th time slot, the battery energy changes according to

$$E_{bk}(i) = \min([0, E_k(i-1) + E_{bk}(i-1) - E_{kc}(i-1)]^+, E_{\max}) \quad (13)$$

where $[0, x]^+ = \max(x, 0)$, $E_{bk}(i)$ and $E_k(i)$ denote the battery energy and harvested energy in the i th time slot for each terminal k , respectively, $E_{kc}(i)$ is the energy consumed in the i th time slot, and $E_{\max}$ is the battery storage capacity. When $k \in \mathcal{K}_t$, $E_{kc}(i) = p_k^t \tau_1(i)$ and when $k \in \mathcal{K}_o$, $E_{kc}(i) = \sum_{m=1}^M x_{k,m}(i) E_{k,m}^{\text{off}}(i) + x_{k,0}(i) E_k^l$.

B. Problem Formulation

In this article, we aim to minimize the weighted sum of energy transfer time and computation time in each time slot by optimizing energy transmission time τ_0 , energy beam-forming vectors $\{\mathbf{v}_m\}_{m=1}^M$, offloading factors $\{x_{k,m}\}_{k=1, m=1}^{k=|\mathcal{K}_o|, m=M}$, information transmission time τ_1 , and combining vectors $\{\boldsymbol{\omega}_k\}_{k=1}^{|\mathcal{K}_t|}$ under the constraints of consumed energy and bit transmission requirements of terminals in $\mathcal{K}_t$. The problem is formulated as

$$\begin{aligned} \text{Variables : } & \tau_0, \tau_1, \{x_{k,m}\}_{k=1, m=1}^{k=|\mathcal{K}_o|, m=M}, \{\mathbf{v}_m\}_{m=1}^M, \{\boldsymbol{\omega}_k\}_{k=1}^{|\mathcal{K}_t|} \\ \text{minimize } & \left\{ \tau_0 + \sum_{k=1}^{|\mathcal{K}_o|} w_k \tau_{2,k} \right\} \end{aligned} \quad (14)$$

$$\text{s. t. } p_k^t \tau_1 \leq E_{bk} + E_k \quad \forall k \in \mathcal{K}_t \quad (14a)$$

$$\sum_{m=1}^M x_{k,m} E_{k,m}^{\text{off}} + x_{k,0} E_k^l \leq E_{bk} + E_k \quad \forall k \in \mathcal{K}_o \quad (14b)$$

$$E_{bk} + E_k \leq E_{\max} \quad \forall k \in \mathcal{K} \quad (14c)$$

$$\|\mathbf{v}_m\|^2 \leq P_m, m = 1, \dots, M \quad (14d)$$

$$\tau_0 \|\mathbf{v}_m\|^2 + \sum_{k=1}^{|\mathcal{K}_o|} x_{k,m} E_{k,m}^{\text{mec}} \leq E_m, m = 1, \dots, M \quad (14e)$$

$$x_{k,m} \in \{0, 1\} \quad \forall k \in \mathcal{K}_o, 0 \leq m \leq M \quad (14f)$$

$$\sum_{m=0}^M x_{k,m} = 1 \quad \forall k \in \mathcal{K}_o \quad (14g)$$

$$B_k \geq b_k \quad \forall k \in \mathcal{K}_t \quad (14h)$$

$$\tau_{2,k} \leq T_{kmax} \quad \forall k \in \tilde{\mathcal{K}}_o \quad (14i)$$

where $\{w_k\}$ s are the weighted values representing the priority of terminals in $\mathcal{K}_o$, P_m , E_m , b_k , and T_{kmax} denote transmit power constraint, total energy consumption constraint of RRH m , information bits transmission constraint and the computation delay constraint of terminals in $\tilde{\mathcal{K}}_o$, respectively, and $\tilde{\mathcal{K}}_o \in \mathcal{K}_o$. We assume that terminals in $\tilde{\mathcal{K}}_o$ have strict computation delay requirements. (14a) and (14b) denote the causal constraints of terminal energy consumption at current time, (14c) denotes the battery capacity constraint of each terminal, (14d) denotes the transmit power constraint of each RRH, (14e) is the energy consumption constraint of each RRH, (14h) is the bits transmission requirement of terminals in $\mathcal{K}_t$, and (14i) indicates the latency requirement of terminals in $\tilde{\mathcal{K}}_o$.

IV. ALTERNATING OPTIMIZATION

The optimization problem in (14) is a nonconvex nonlinear mixed integer programming problem, and it is difficult to obtain the optimal solution. In this section, we solve (14) with AO algorithm, which solves two subproblems in (15) and (25) alternately in N_{iter} iterations. The AO algorithm guarantees the output of a locally optimal solution.

A. Optimal Solution To (14) for Given $\{x_{k,m}\}$

For given $\{x_{k,m}\}$, problem (14) reduces to the following nonconvex problem:

$$\begin{aligned} \text{Variables : } & \tau_0, \tau_1, \{\mathbf{v}_m\}_{m=1}^M, \{\boldsymbol{\omega}_k\}_{k=1}^{|\mathcal{K}_t|} \\ \text{minimize } & \left\{ \tau_0 + \sum_{k=1}^{|\mathcal{K}_o|} w_k \tau_{2,k} \right\} \\ \text{s. t. } & (14a) - (14e), (14h). \end{aligned} \quad (15)$$

1) *Optimal Solution of $\{\boldsymbol{\omega}_k\}$ and τ_1 :* We can see that variables $\{\boldsymbol{\omega}_k\}$ s are only related to the constraint (14h). Therefore, the optimal $\{\boldsymbol{\omega}_k\}$ s can be determined by maximizing γ_k in (6). γ_k can be re-expressed as

$$\gamma_k = \frac{p_k^t \boldsymbol{\omega}_k^H \mathbf{g}_k \mathbf{g}_k^H \boldsymbol{\omega}_k}{\boldsymbol{\omega}_k^H \mathbf{S}_{ub} \boldsymbol{\omega}_k} \quad (16)$$

where $\mathbf{S}_{ub} = \sum_{j \neq k} p_j^t \mathbf{g}_j \mathbf{g}_j^H + \phi^2 \mathbf{I}$. Defining $\tilde{\boldsymbol{\omega}}_k = \mathbf{S}_{ub}^{(1/2)} \boldsymbol{\omega}_k$, we have

$$\gamma_k = \frac{p_k^t \tilde{\boldsymbol{\omega}}_k^H \mathbf{S}_{ub}^{-\frac{H}{2}} \mathbf{g}_k \mathbf{g}_k^H \mathbf{S}_{ub}^{-\frac{1}{2}} \tilde{\boldsymbol{\omega}}_k}{\tilde{\boldsymbol{\omega}}_k^H \tilde{\boldsymbol{\omega}}_k}. \quad (17)$$

According to the Rayleigh–Ritz theorem, when $\tilde{\mathbf{w}}_k$ is the eigenvector of $\mathbf{S}_{ub}^{-(H/2)} \mathbf{g}_k \mathbf{g}_k^H \mathbf{S}_{ub}^{-(1/2)}$ corresponding to the maximum eigenvalue $\lambda_{\max}(\mathbf{S}_{ub}^{-(H/2)} \mathbf{g}_k \mathbf{g}_k^H \mathbf{S}_{ub}^{-(1/2)})$, γ_k reaches the maximum

$$\gamma_k^* = \lambda_{\max} \left(\mathbf{S}_{ub}^{-\frac{H}{2}} \mathbf{g}_k \mathbf{g}_k^H \mathbf{S}_{ub}^{-\frac{1}{2}} \right) p_k^t. \quad (18)$$

On the other hand, the variable τ_1 is only related to constraints (14a) and (14h). If problem (15) is feasible, the optimal solution τ_1^* must satisfy

$$\frac{b_k}{w_t \log(1 + \gamma_k^*)} \leq \tau_1^* \leq \frac{E_{bk} + E_k^*}{p_k^t} \quad \forall k \in \mathcal{K}_t \quad (19)$$

where E_k^* is determined by the optimal solution to (15). τ_1^* can take any value in the interval denoted by (19) without effecting the value of objective function. Since the lower boundary point is only related to the previously determined $\{\boldsymbol{\omega}_k\}$ s, the optimal solution of τ_1 is chosen as

$$\tau_1^* = \max_{k \in \mathcal{K}_t} \frac{b_k}{w_t \log(1 + \gamma_k^*)}. \quad (20)$$

2) *Optimal Solution of $\{\mathbf{v}_m\}$:* In this section, SDR is applied to get the optimal $\{\mathbf{v}_m\}$. By introducing a new variable $\mathbf{O}_m = \mathbf{v}_m \mathbf{v}_m^H$, E_k in (2) can be expressed as

$$E_k = \eta \tau_0 \sum_{m=1}^M \text{Tr}(\mathbf{g}_{k,m} \mathbf{g}_{k,m}^H \mathbf{O}_m). \quad (21)$$

Meanwhile, the constraints (14d) and (14e) can be transformed into

$$\text{Tr}(\mathbf{O}_m) \leq P_m \quad \forall m \in M \quad (22)$$

and

$$\tau_0 \text{Tr}(\mathbf{O}_m) + \sum_{k=1}^{|\mathcal{K}_o|} x_{k,m} E_{k,m}^{\text{mec}} \leq E_m \quad \forall m \in M. \quad (23)$$

Then, problem (15) can be expressed as

$$\begin{aligned} \text{Variables : } & \tau_0, \mathbf{O}_m \\ \text{minimize } & \left\{ \tau_0 + \sum_{k=1}^{|\mathcal{K}_o|} w_k \tau_{2,k} \right\} \end{aligned} \quad (24)$$

$$\text{s. t. } p_k^t \tau_1 \leq E_{bk} + E_k \quad \forall k \in \mathcal{K}_t \quad (24a)$$

$$\sum_{m=1}^M x_{k,m} E_{k,m}^{\text{off}} + x_{k,0} E_k^l \leq E_{bk} + E_k \quad \forall k \in \mathcal{K}_o \quad (24b)$$

$$E_{bk} + E_k \leq E_{\max} \quad \forall k \in \mathcal{K} \quad (24c)$$

$$\mathbf{O}_m \geq 0 \quad \forall m \in M \quad (24d)$$

$$\text{Tr}(\mathbf{O}_m) \leq P_m \quad \forall m \in M \quad (24e)$$

$$\tau_0 \text{Tr}(\mathbf{O}_m) + \sum_{k=1}^{|\mathcal{K}_o|} x_{k,m} E_{k,m}^{\text{mec}} \leq E_m \quad \forall m \in M \quad (24f)$$

$$\text{rank}(\mathbf{O}_m) = 1, 1 \leq m \leq M. \quad (24g)$$

To transform (24) into a convex problem, we first relax the rank constraint in (24g) utilizing the SDR technique and then search for the minimum τ_0 by solving a convex feasible problem repeatedly. Specifically, for a given τ_0 , the optimal $\mathbf{O}_m$ without the rank constraint in (24g) can be obtained by solving an LP problem. Therefore, the minimum τ_0 which makes the problem feasible can be found by a bisection method [20]. Moreover, after applying SDR, we have the following conclusion about the optimality of the solution.

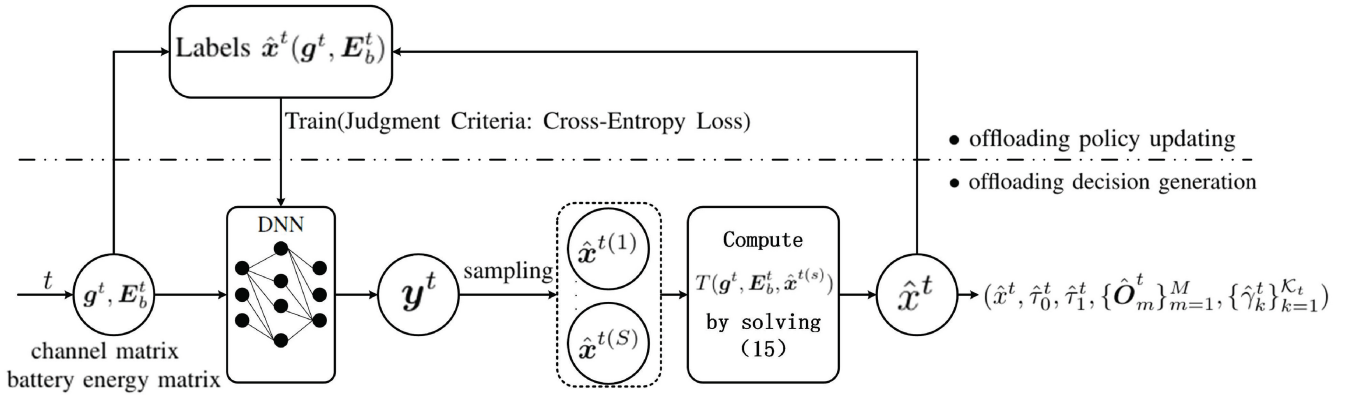


Fig. 2. Structure of the DNN-based self-evolution optimization algorithm.

Lemma: For a given τ_0 , the optimal solution $\mathbf{O}_m^*$ of problem (24) without the rank constraint is a rank one matrix.

Proof: Refer to the Appendix. ■

B. Optimal Solution To $\{x_{k,m}\}$

For given $\{\tau_0, \tau_1, \{\mathbf{v}_m\}_{m=1}^M, \{\omega_k\}_{k=1}^{|\mathcal{K}_t|}\}$, we solve the following problem:

$$\begin{aligned} \text{Variables : } & \{x_{k,m}\}_{k=1, m=1}^{k=|\mathcal{K}_o|, m=M} \\ \text{minimize } & \left\{ \tau_0 + \sum_{k=1}^{|\mathcal{K}_o|} w_k \tau_{2,k} \right\} \\ \text{s. t. } & (14b), (14e) - (14g), (14i). \end{aligned} \quad (25)$$

By substituting (8) and (10) into (12), we have

$$\begin{aligned} \tau_{2,k} = & \sum_{m=1}^M \sum_{m'=1}^M \sum_{k'=1}^{|\mathcal{K}_o|} \frac{R1_{k,x_{k,m}x_{k',m'}}}{w_o \log\left(1 + \frac{\|\mathbf{g}_{k,m}\|^2 p_k^o}{\phi^2}\right)} \\ & + \sum_{m=1}^M \sum_{k'=1}^{|\mathcal{K}_o|} \frac{R2_{k,x_{k,m}x_{k',m}}}{F_m} + x_{k,0} t_k^l. \end{aligned} \quad (26)$$

Similarly, by substituting (9) into (14b) and substituting (11) into (14e), (14b) and (14e) can be re-expressed as

$$\begin{aligned} \sum_{k'=1}^{|\mathcal{K}_o|} \sum_{m=1}^M \sum_{m'=1}^M \frac{R1_{k,x_{k,m}x_{k',m'}}}{w_o \log\left(1 + \frac{\|\mathbf{g}_{k,m}\|^2 p_k^o}{\phi^2}\right)} + x_{k,0} E_k^l & \leq E_{bk} + E_k \\ \forall k \in \mathcal{K}_o \\ \frac{p_m^{\text{mec}} R2_k \sum_{k=1}^{|\mathcal{K}_o|} \sum_{k'=1}^{|\mathcal{K}_o|} x_{k,m} x_{k',m}}{F_m} + \tau_0 \text{Tr}(\mathbf{O}_m) & \leq E_m \quad \forall m \in M. \end{aligned}$$

The above equations contain quadratic terms of integer variables, which make it difficult to solve (25) with existing integer programming tools. As $\{x_{k,m}\}$ are binary variables, we can transform $x_{k,m}x_{k',m'}$ to a linear term, i.e., $2x_{k,m}x_{k',m'} = (x_{k,m} + x_{k',m'} - 1) + |x_{k,m} + x_{k',m'} - 1|$. Through such transformation, problem (25) becomes a linear integer programming problem, which can be solved by the YALMIP.

The complexity of this AO algorithm is analyzed as follows. The algorithm divides the problem into subproblems (15) and (25) for AO. The SDP problems are commonly solved by the interior point method (IPM) [21], which obtains an

ϵ -optimal solution after a sequence of iterations with the given ϵ . In problem (24), the constraints are linear matrix inequality (LMI). In each iteration, the computation complexity for solving SDP can be given by $\mathcal{O}(\sqrt{2} \ln(1/\epsilon)(2|\mathcal{K}| + 3M)^{3.5} N^{6.5})$ by reserving the highest order term. Given the number of bisection search N_f in the convex feasibility solution for τ_0 , the computation complexity of problem (15) is $\mathcal{O}(\sqrt{2} N_f \ln(1/\epsilon)(2|\mathcal{K}| + 3M)^{3.5} N^{6.5})$. The linear integer programming problem can be solved in polynomial time using the YALMIP. Given the $\mathbf{x}$ matrix of size $M \times |\mathcal{K}_o|$, the computation complexity can be expressed as $\mathcal{O}(M^2 |\mathcal{K}_o|^2)$. In N_{iter} iteration times, the overall complexity of AO algorithm is $\mathcal{O}(N_{\text{iter}}(\sqrt{2} N_f \ln(1/\epsilon)(2|\mathcal{K}| + 3M)^{3.5} N^{6.5} + M^2 |\mathcal{K}_o|^2))$.

V. DEEP-LEARNING-BASED OPTIMIZATION METHOD

In the dynamic network, CP needs to solve (14) in real time with changing channel information $\mathbf{g} = [\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{|\mathcal{K}_o|}]$ and battery power $\mathbf{E}_b = [E_{b1}, E_{b2}, \dots, E_{b|\mathcal{K}_o|}]$, which leads to heavy computing burden. To reduce the real-time computing burden and delay, we further propose optimization methods based on DNN. Different from the AO algorithm, this method use the deep learning approach to obtain the offloading matrix $\mathbf{x}$. It is noted that we do not train DNN based on real-time network states, but train DNN offline in advance with a large number of samples of network states (including channel states and battery power) to obtain the optimal resource allocation scheme in an average sense. With the trained DNN, optimization results can be obtained quickly in a real-time system.

As shown in Fig. 2, we adopt DNN to represent the non-linear relationship between the network state $(\mathbf{g}, \mathbf{E}_b)$ and the computation offloading factors. The DNN consists of three layers: 1) input; 2) hidden; and 3) output. Given the network state $(\mathbf{g}, \mathbf{E}_b)$, we adopt a rectified linear unit (ReLU) as the activation function in the input layer and hidden layer. In the output layer, we use the softmax function as the output function to map the output value $\mathbf{y}^t = [(\mathbf{y}_1^t)^T, (\mathbf{y}_2^t)^T, \dots, (\mathbf{y}_{|\mathcal{K}_o|}^t)^T]^T$, $\mathbf{y}_k^t = [y_{k,0}^t, y_{k,1}^t, \dots, y_{k,M}^t]$ to $[0, 1]$ and make $\mathbf{y}_k^t \mathbf{1}^T = 1$, where $\mathbf{1}$ is a row vector with all 1. The relationship between the input and output is

$$\mathbf{y}^t = f_{\theta_t}(\mathbf{g}^t, \mathbf{E}_b^t) \quad (27)$$

where t is the t th time frame. The parameters θ_t of the DNN are randomly initialized following a zero-mean normal distribution when $t = 1$. The loss function of DNN is the average cross-entropy loss

$$L(\theta_t) = -\frac{1}{D} \sum_{d \in D} \left[\left(\hat{x}^d \right)^T \log f_{\theta_t}(\mathbf{g}^d, \mathbf{E}_b^d) + \left(1 - \hat{x}^d \right)^T \log \left(1 - f_{\theta_t}(\mathbf{g}^d, \mathbf{E}_b^d) \right) \right] \quad (28)$$

where D denotes the sample size for updating parameters, and $\hat{x}^d$ denotes the action label. The parameters θ_t of the DNN are updated by applying the Adam algorithm [22].

In the learning process of computation offloading decisions, the most important aspect is the generation of labels $\hat{x}^d(\mathbf{g}^d, \mathbf{E}_b^d)$. We propose two methods to obtain labels: preoptimization and self-evolution. In the preoptimization method, we first select a batch of network states as samples, compute the optimal $\hat{x}^d$ for each given $\mathbf{g}^d$ and $\mathbf{E}_b^d$ using the AO algorithm, and then the optimization results are stored in the Label buffer as fixed labels for training the DNN. Parameters of DNN θ_t are updated to minimize the average cross-entropy loss function in (28), so that the output of DNN converges to the label value as the number of iterations increases. By combining AO algorithm and deep learning algorithm, it speeds up the training process of the DNN. With the trained neural network, we sample the output $\mathbf{y}^t$ of the neural network and select the offloading matrix that satisfies (14i) and maximizes the objective function of (15) from the sample matrices as the optimal offloading matrix.

If action labels cannot be obtained by solving conventional optimization method, we can obtain action labels by learning with self-evolution selection, as shown in Fig. 2. Denoting the output of DNN $y_{k,m}^t$ as the probability that terminal k selects RRH m , the random number generated according to the discrete probability distribution $\mathbf{y}_k^t$ determines RRH selected by the terminal k . Based on $\mathbf{y}^t$, we can get S sets of sample matrices $[\hat{\mathbf{x}}^{t(1)}, \hat{\mathbf{x}}^{t(2)}, \dots, \hat{\mathbf{x}}^{t(S)}]$, where $\hat{\mathbf{x}}^{t(s)}$ is the matrix composed of $|\mathcal{K}_o| \times M$ offloading factors. We choose S' offloading matrices satisfying (14i) from S sample matrices. For each sample matrix, we solve (15) to obtain the optimal objective function $T(\mathbf{g}^t, \mathbf{E}_b^t, \hat{\mathbf{x}}^{t(s)})$. The optimal state-action value $(\mathbf{g}^t, \mathbf{E}_b^t) - \hat{\mathbf{x}}^t$ where $\hat{\mathbf{x}}^t = \underset{\mathbf{x}^{t(s)}, s \in S'}{\operatorname{argmin}} T(\mathbf{g}^t, \mathbf{E}_b^t, \hat{\mathbf{x}}^{t(s)})$ is selected and stored in

the Label buffer for training DNN. It is noted that S must be large enough to ensure that $S' > 0$. We update parameters of DNN every $\mathcal{T}$ time frames by selecting D labels randomly from the buffer. According to [10], such experience replay technique can reduce the variance of θ_t by utilizing historical data and accelerate the convergence by reducing the correlation of training samples. In general, DNN continuously draws the better labels from the Label buffer for updating the network parameters and outputs better offloading decisions over time. The pseudocode of the DNN-based self-evolution optimization algorithm is shown in Algorithm 1.

The complexity of the deep-learning-based optimization method is analyzed as follows. Note that the complexity of neural networks is generally expressed in terms of floating-point operations (FLOPs). Given the input-layer dimension

Algorithm 1 DNN-Based Self-Evolution Optimization Algorithm

Initialize the DNN with θ_0 and empty Label buffer;
 Set iteration number N_{iter} and the training interval $\mathcal{T}$;

- 1: **for** $t = 1, 2, \dots, N_{iter}$ **do**
- 2: DNN outputs a softmax relaxed action $\mathbf{y}^t = f_{\theta_t}(\mathbf{g}^t, \mathbf{E}_b^t)$;
- 3: Generate S sets of sample matrices $[\mathbf{x}^{t(1)}, \mathbf{x}^{t(2)}, \dots, \mathbf{x}^{t(S)}]$ based on $\mathbf{y}^t$, choose S' offloading matrices satisfying (14i) and select the sample according to $\hat{\mathbf{x}}^t = \underset{\mathbf{x}^{t(s)}, s \in S'}{\operatorname{argmin}} T(\mathbf{g}^t, \mathbf{E}_b^t, \mathbf{x}^{t(s)})$;
- 4: Update the Label buffer by adding new state-action pair $(\mathbf{g}^t, \mathbf{E}_b^t) - \hat{\mathbf{x}}^t$;
- 5: **if** $t \bmod \mathcal{T} = 0$ **then**
- 6: Randomly select D pairs of $(\mathbf{g}^t, \mathbf{E}_b^t) - \hat{\mathbf{x}}^t$ from Label buffer to update θ_t by using the Adam Algorithm;
- 7: **end if**
- 8: **end for**

$(MN + 1)|\mathcal{K}_o|$, the hidden-layer dimension D_h and the output layer dimension $|\mathcal{K}_o|M$, its FLOPs of one forward propagation can be expressed as $(2(MN + 1)|\mathcal{K}_o| - 1)D_h + (2D_h - 1)|\mathcal{K}_o|M$. In the training process, for the preoptimization method, with given action labels, the complexity of it is $\mathcal{O}(N_{iter}((2(MN + 1)|\mathcal{K}_o| - 1)D_h + (2D_h - 1)|\mathcal{K}_o|M))$. In the self-evolution method, it is essential to get the optimal solution by solving problem (15) to update the Label Buffer. Combined with the complexity of solving (15), the overall complexity of the self-evolution method is $\mathcal{O}(N_{iter}((2(MN + 1)|\mathcal{K}_o| - 1)D_h + (2D_h - 1)|\mathcal{K}_o|M + \sqrt{2}S'N_f \ln(1/\epsilon)(2|\mathcal{K}| + 3M)^{3.5}N^{6.5}))$. With the trained DNN, the complexity of two methods in real-time operation is $\mathcal{O}((2(MN + 1)|\mathcal{K}_o| - 1)D_h + (2D_h - 1)|\mathcal{K}_o|M + \sqrt{2}S'N_f \ln(1/\epsilon)(2|\mathcal{K}| + 3M)^{3.5}N^{6.5})$.

VI. NUMERICAL RESULTS

In this section, we provide some numerical results to evaluate the performances of the proposed AO algorithm and deep-learning-based algorithm. We focus on comparing the performance of different algorithms and demonstrating the impact of different parameters on system performance.

A. Parameter Setting

In the network configuration, M RRHs with N antennas are located on a square, and K terminals are distributed within a square area of area $10 \text{ m} \times 10 \text{ m}$. The carrier frequency of wireless transmission is 26 GHz. The channel bandwidth allocated to terminals are 200 MHz, among which 20 and 180 MHz are allocated to terminals in $\mathcal{K}_t$ and $\mathcal{K}_o$, respectively. For each terminal, the energy harvesting efficiency is set to $\eta = 0.5$. We assume that terminals in $\mathcal{K}_o$ have same priority so that all w_k in (15) are set to $(1/|\mathcal{K}_o|)$. In Figs. 3–6 and Fig. 8, we set $|\tilde{\mathcal{K}}_o| = 0$ and in Fig. 7, we set $|\tilde{\mathcal{K}}_o| = 2$. Unless specified otherwise, the system parameters are summarized in Table I.

TABLE I
PARTIAL PARAMETERS

Parameter	Value
learning rate	$lr = 0.01$
Label buffer size	$L = 1024$
training batch size	$D = 128$
training interval	$\mathcal{T} = 10$
communication task	$R1_k = 600$ bits
computation task	$R2_k = 1000$ cycles
computation rate of RRH m	$F_m \in (20000, 60000)$ cycles/s
computation rate of terminal k	$f_{k,l} = 5000$ cycles/s
computation power of RRH m	$p_m^{rec} = 50$ W
computation power of terminal k	$p_k^t = 10$ W
number of RRHs	$M = 2$
number of antennas	$N = 5$
number of computation terminals	$ \mathcal{K}_o = 20$
number of communication terminals	$ \mathcal{K}_t = 2$
energy constraint of RRH m	$E_m = 200$ J
bits transmission constraint of terminal k	$b_k = 10$ Mbits
battery capacity constraint of terminal k	$E_{max} = 4.4$ Wh
transmit power of the terminal k	$p_k^t = p_k^o = -1$ dB
distance between RRH m and terminal k	$d_{k,m} \in (5, 10)$ m
receiver noise power	$\phi^2 = -113$ dBm

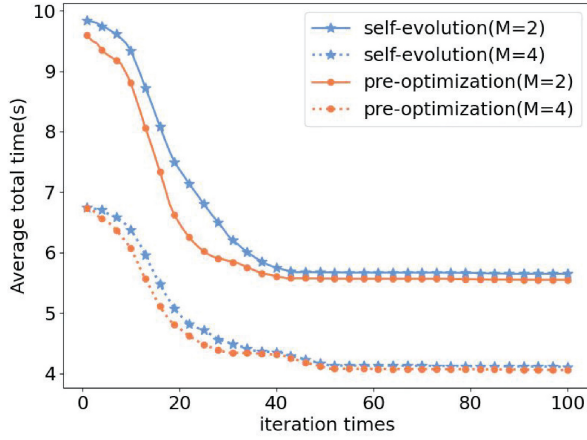


Fig. 3. Learning curves of average total time consumption for two deep-learning-based optimization methods: preoptimization and self-evolution.

B. Numerical Results

In this section, performance metric in each Figure is obtained by averaging 5000 channel realizations. In different time slots, battery energy changes according to (13). In Fig. 3, the learning curves of two deep-learning-based optimization methods are exhibited. We use the update times of θ_t as the abscissa, and the average value of optimal objective function $T(\mathbf{g}, \mathbf{E}_b, \hat{\mathbf{x}})$ in the test set as the ordinate. We train the DNN with time-varying channel matrix $\mathbf{g}^t$, and adopt a fixed set of channel matrices $[\mathbf{g}^1, \mathbf{g}^2, \dots, \mathbf{g}^{T_n}]$ as the test set, where T_n is the size of the test set. For each channel realization, $\mathbf{E}_b^t$ change according to (13) with an initial value $\mathbf{E}_b^0 = 0$. Once the network parameters θ_t are updated, we calculate the average objective function with channel matrices in the test set to observe the learning property of the DNN. From Fig. 3, we can see that the preoptimization method achieves better steady-state performance than the self-evolution method because a sufficient amount of AO optimization results are given as labels for training. Fig. 3 also reveals that performance gap between the self-evolution method and the preoptimization

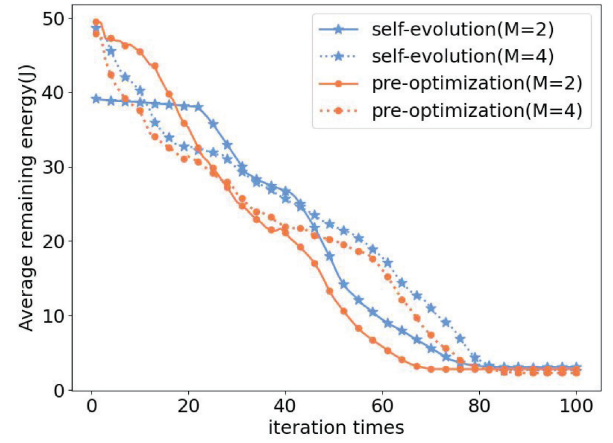


Fig. 4. Learning curves of average remaining energy for two deep-learning-based optimization methods: preoptimization and self-evolution.

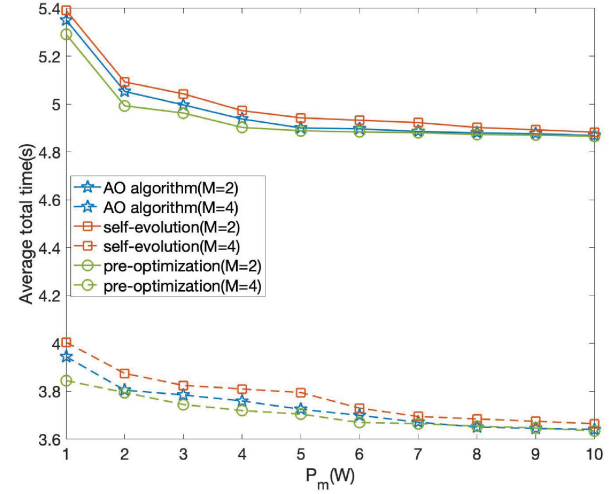


Fig. 5. Performance comparison of different algorithms.

method is small. When the number of RRHs M decreases, the dimensions of input $\mathbf{g}^t$ and output $\hat{\mathbf{x}}^t$ are reduced, which accelerates the convergence of DNN.

Fig. 4 presents the learning curves of the average remaining energy of two deep-learning-based optimization methods. We set ten computation tasks in each channel realization for each terminal in $\mathcal{K}_o$ and the average remaining energy after completing the ten tasks is exhibited. In order to improve the utilization rate of the harvested energy and reduce the energy transfer time τ_0 , the offloading policy tends to allow terminals with more remaining battery power to adopt local computing. As shown in Fig. 4, the average residual energies of terminals decrease with the increase of iteration times.

In Fig. 5, we demonstrate the relationship between the average weighted time and the RRH transmit power P_m with varying M in three different algorithms. The AO algorithm outputs a local optimal solution based on the real-time channel and battery power states, which can be regarded as a benchmark algorithm for evaluating the generalization performance of DNN-based methods. It can be seen that the optimization

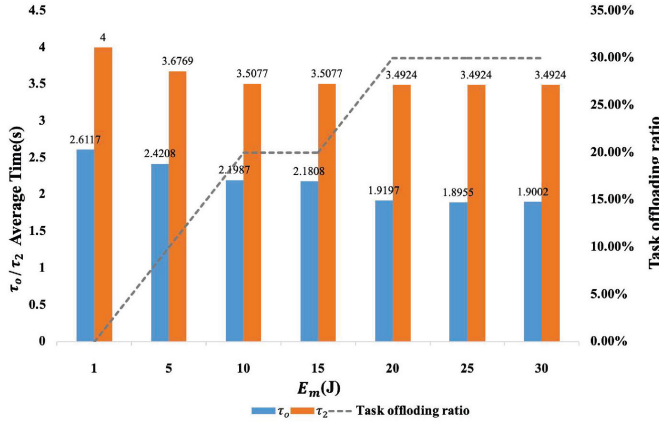


Fig. 6. Effects of energy constraint E_m of RRH m on the time consumption and the offloading ratio.

results of the two DNN-based algorithms can achieve similar performance as the AO-based method, indicating that the optimization results obtained by DNN-based methods have good generalization on test channels. The AO algorithm optimizes transmit parameters based on real-time network state, which needs multiple iterations for each input to get the optimal result. Since the offloading factor x is mainly affected by the number of RRHs M and the number of computation terminals $|\mathcal{K}_o|$, taking $M = 2$, $|\mathcal{K}_o| = 20$ as an example, carried out in a computer with the Intel(R) Xeon(R) W-2295 CPU @ 3.00-GHz processor and 256-GB 2934-MHz DDR4, AO algorithm takes 539.6 s to get the optimal solution for a given network state, but with a trained DNN, the algorithm based on deep learning only takes 11.59 s to output the optimization result, which can better meet the real-time requirements. As the RRH transmit power P_m increases, the energy transfer time τ_0 is significantly reduced, so that the average total time consumption decreases. When P_m is close to 10 W, τ_0 is less than 0.03, so the change of the curve tends to be smooth. In addition, as the number of RRHs M decreases, RRHs sharing the offloading task decreases, which leads to an increase in the computation time $\tau_{2,k}$ and, thus, an increase in the average total time.

In Fig. 6, we evaluate the impact of the energy constraint of RRH m on the energy transfer time τ_0 , the total task computation time $\tau_2 = \sum_{k=1}^{|\mathcal{K}_o|} w_k \tau_{2,k}$, and the task offloading ratio. We set $P_m = 1$ W and the other parameter settings are shown in Table I. According to constraints (14a), (14b) and (14e), when E_m is small so that it can only meet the energy transfer requirements of terminals in $\mathcal{K}_t$, all terminals in $\mathcal{K}_o$ have to implement local computation. Due to short distances between RRHs and terminals, the energy for local computation is greater than that for task offloading. Therefore, in this case, both the energy transfer time and computation time of terminals are the longest. As E_m increases, the RRH side gradually has ability to assist in computing, so the proportion of task offloading begins to increase, and the average computing time decreases. Meanwhile, increasing E_m will raise remaining battery power of some terminals. By effectively utilizing the remaining battery power, terminals implementing

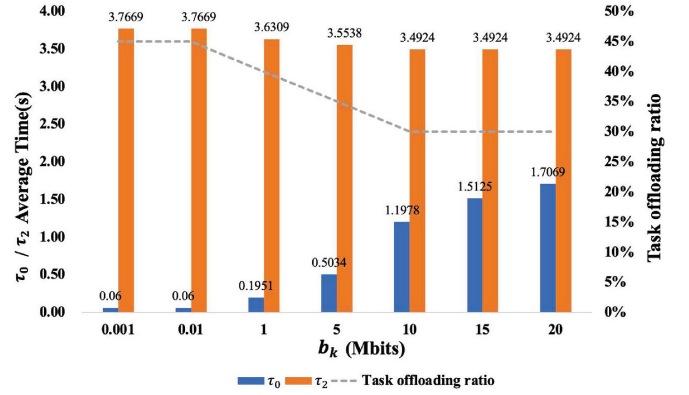


Fig. 7. Influence of the information bit transmission constraint b_k on the time consumption and the offloading ratio.

local computing need to harvest less energy, thus reducing the energy transmission time τ_0 . However, due to limited computing resources of RRH, when too many computing tasks are offloaded to the same RRH, the computing time will increase instead. As a result, when the optimal number of offloading tasks is reached, offloading tasks will no longer continue to increase with the increase of E_m . In our settings, to ensure the least overall average latency, the optimal offloading ratio is 30%. That is, when $E_m \geq 20$ J, each RRH accepts about three offloading tasks, and the remaining terminals implement local computation. It is observed that when the task offloading ratio is fixed, increasing E_m has marginal effect on τ_0 .

Fig. 7 shows the influence of b_k on the energy transfer time τ_0 , the total task computation time τ_2 and the task offloading ratio. We set $P_m = 10$ W and the other parameter settings are the same as those in Fig. 6. From the optimization problem (14), we can infer that the increase of b_k will lead to the increase of τ_1 and τ_0 , which is verified by the results in Fig. 6. To ensure the shortest total latency, when b_k is small, the energy transfer process will guarantee that some terminals complete local computation and some terminals achieve task offloading. As shown in Fig. 7, the shortest average time is 3.8269 s when $b_k = 0.001$ Mbits. When $b_k \leq 0.01$ Mbits, τ_0 and τ_2 will not change with the increasing of b_k . When $b_k > 0.01$ Mbits, τ_0 increases with the increasing of b_k and more terminals in $\mathcal{K}_o$ start to perform local computation due to enough harvested energy, which results in slight decrease in computation time. However, the increase of τ_0 with b_k is greater than the decrease of τ_2 caused by the reduction of offloading tasks, and the average total time gradually increases. When $b_k \geq 10$ Mbits, the task offloading ratio reaches the optimal value, τ_2 no longer changes, and increase of b_k only brings about the increase of τ_0 .

Fig. 8 presents the effect of T_{kmax} in (14i) on the energy transfer time τ_0 , the total task computation time τ_2 and the task offloading ratio. We set $|\mathcal{K}_o| = 2$, $F_m = 150000$ cycles/s, $P_m = 1$ W and the other parameter settings are shown in Table I. In Fig. 8, when the latency requirement in (14i) is removed, all terminals in $\mathcal{K}_o$ offload the tasks to RRHs and the shortest average total time is 2.8458 s. However, with (14i), to ensure that terminals in $\mathcal{K}_o$ satisfy the delay requirements,

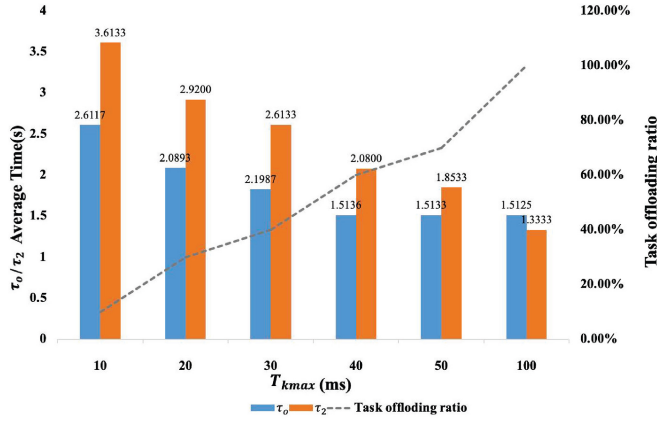


Fig. 8. Effect of the computation delay constraint T_{kmax} in (14i) on the time consumption and the offloading ratio.

other terminals in $\mathcal{K}_o$ would choose local computation, resulting in an increase in the energy transfer time τ_0 and the computation time τ_2 . For example, when $T_{kmax} = 10$ ms, each RRHs computation resources can only assist terminals in $\mathcal{K}_o$ to complete computation task within the specified time, so that other terminals in $\mathcal{K}_o$ have to choose local computation, which increases the average total computation time. By relaxing the latency requirement, the task offloading ratio increases, and, thus, τ_2 decreases. At the meanwhile, more terminals have enough residual energy to accomplish local computation. By effectively utilizing the remaining battery power, terminals implementing local computing need to harvest less energy, which reduces τ_0 .

Fig. 9 illustrates the relationship between time consumption and RRH transmit power P_m with varying parameters: τ_0 (t^{off}) versus P_m with different p_k , τ_0 (τ_2) versus P_m with different $\mathcal{K}_o$ and M . The parameter settings are shown in Table I. From Fig. 9, we can see that increase in P_m only results in a decrease in τ_0 and has no effect on other time consumptions. According to (14a), the increase of p_k will lead to an increase in the energy demand of terminals and, thus, the increase of τ_0 . It can be shown by (8) that the task offloading transmission time t^{off} will decrease with the increasing of p_k , as seen from Fig. 9(a). Fig. 9(b) reveals that both τ_0 and τ_2 will increase with the increase of $|\mathcal{K}_o|$. As shown in Fig. 9(c), increasing M means that RRHs which can transmit energy and provide computation resource increases, so both τ_0 and τ_2 are greatly reduced.

VII. CONCLUSION

This article studied a wireless-powered communication network, including multiple multiantenna RRHs and single-antenna terminals with different requirements. Under a harvest-then-use protocol, terminals harvest energy from RF signals from RRHs and then transmit information bits to RRHs or achieve local computation. At the RRHs side, we adopt energy beamforming to improve the energy harvesting efficiency. To minimize the weighted sum of energy transfer time and computation time while meeting the requirement of data transmission, we propose an AO algorithm, and DNN-based algorithms to optimize the energy transfer

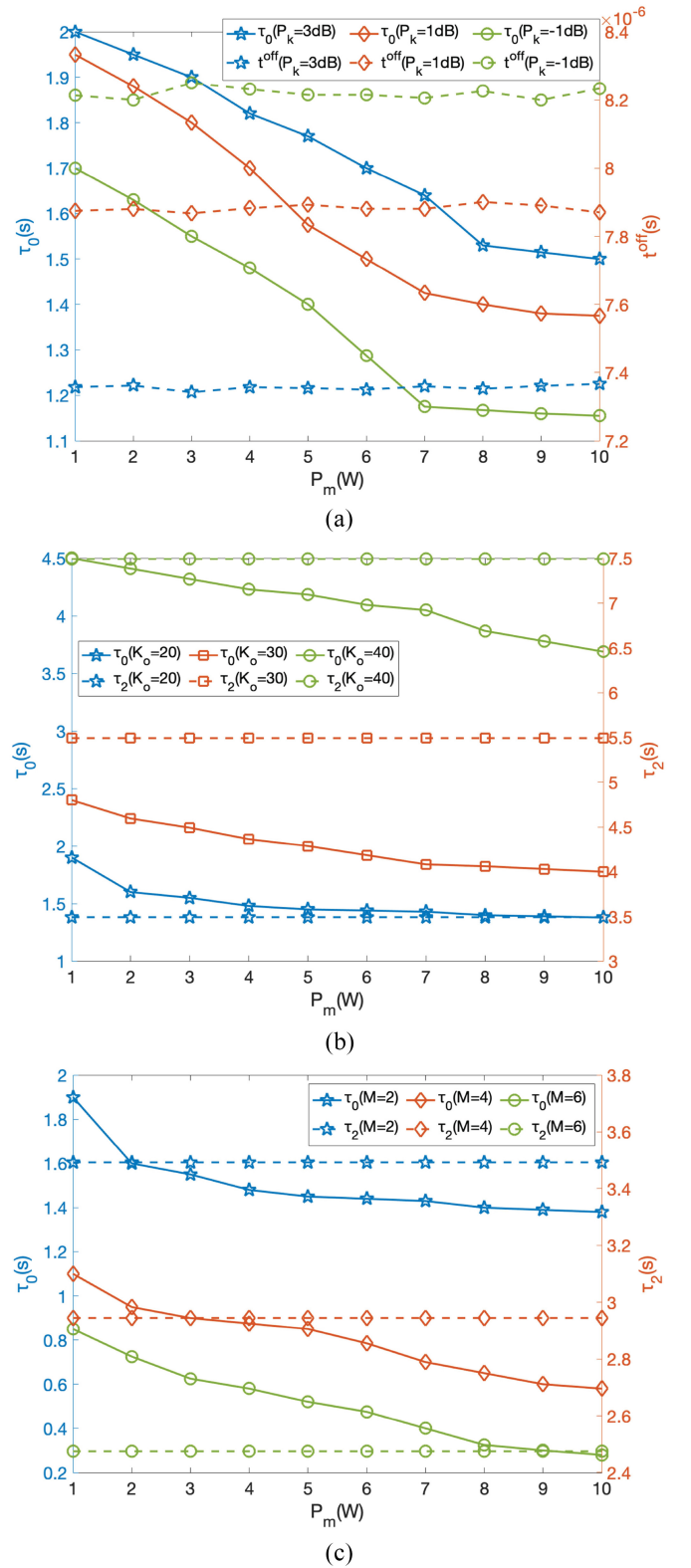


Fig. 9. Relationship between time consumption and RRH transmit power P_m with varying parameters. (a) τ_0 (t^{off}) versus P_m with different p_k . (b) τ_0 (τ_2) versus P_m with different $\mathcal{K}_o$. (c) τ_0 (τ_2) versus P_m with different M .

time, information transmission time, computation offloading allocation, beamforming vectors, and combining vectors, respectively. Simulation results showed that with trained neural

network, methods based on DNN can achieve a similar performance to the AO algorithm with less computation time and can better meet the requirements of real-time applications. Furthermore, the numerical results revealed that the influence of various system parameters on energy transfer time and computation time.

APPENDIX PROOF OF THEOREM 1

For given τ_0 , the Lagrangian of problem (24) is given by

$$\begin{aligned}
\mathcal{L} & \left(\{\lambda_k\}_{k=1}^K, \{\lambda_m^{(3)}\}_{m=1}^M, \{\lambda_m^{(4)}\}_{m=1}^M, \{\mathbf{O}_m\}_{m=1}^M \right) \\
& = \tau_0 + \sum_{m=1}^M \sum_{k \in \mathcal{K}_o} w_k \left[x_{k,m} (t_{k,m}^{\text{off}} + t_{k,m}^{\text{mec}}) + (1 - x_{k,m}) t_k^l \right] \\
& \quad - \sum_{k \in \mathcal{K}_t} \lambda_k \left(\eta \tau_0 \sum_{m=1}^M \text{Tr}(\mathbf{g}_{k,m} \mathbf{g}_{k,m}^H \mathbf{O}_m) - p_k^t \tau_1 \right) \\
& \quad - \sum_{k \in \mathcal{K}_o} \lambda_k \left\{ \eta \tau_0 \sum_{m=1}^M \text{Tr}(\mathbf{g}_{k,m} \mathbf{g}_{k,m}^H \mathbf{O}_m) \right. \\
& \quad \quad \left. - \left[\sum_{m=1}^M x_{k,m} * E_{k,m}^{\text{off}} + \left(1 - \sum_{m=1}^M x_{k,m} \right) * E_k^l \right] \right\} \\
& \quad + \sum_{m=1}^M \lambda_m^{(3)} [\text{Tr}(\mathbf{O}_m) - P_m] \\
& \quad + \sum_{m=1}^M \lambda_m^{(4)} \left[\tau_0 \text{Tr}(\mathbf{O}_m) + \sum_{k \in \mathcal{K}_o} x_{k,m} E_{k,m}^{\text{mec}} - E_m \right] \\
& = \tau_0 + \sum_{m=1}^M \sum_{k \in \mathcal{K}_o} w_k \left[x_{k,m} (t_{k,m}^{\text{off}} + t_{k,m}^{\text{mec}}) + (1 - x_{k,m}) t_k^l \right] \\
& \quad + \sum_{k \in \mathcal{K}_t} \lambda_k p_k^t \tau_1 \\
& \quad + \sum_{k \in \mathcal{K}_o} \lambda_k \left[\sum_{m=1}^M x_{k,m} * E_{k,m}^{\text{off}} + \left(1 - \sum_{m=1}^M x_{k,m} \right) * E_k^l \right] \\
& \quad - \sum_{m=1}^M \left(\lambda_m^{(3)} P_m - \lambda_m^{(4)} \left(\sum_{k \in \mathcal{K}_o} x_{k,m} E_{k,m}^{\text{mec}} - E_m \right) \right) \\
& \quad + \sum_{m=1}^M \text{Tr} \left\{ \left[\lambda_m^{(3)} \mathbf{I} + \lambda_m^{(4)} \tau_0 \mathbf{I} - \sum_{k \in \mathcal{K}} \lambda_k \tau_0 (\mathbf{g}_{k,m} \mathbf{g}_{k,m}^H) \right] \mathbf{O}_m \right\}
\end{aligned} \tag{29}$$

where $\{\lambda_k\}_{k=1}^K$, $\{\lambda_m^{(3)}\}_{m=1}^M$, $\{\lambda_m^{(4)}\}_{m=1}^M$ are the Lagrange multipliers. Denote $\mathbf{A}_m = \lambda_m \mathbf{I} - \sum_{k \in \mathcal{K}} \lambda_k \mathbf{F}_k^m$, where $\lambda_m = \lambda_m^{(3)} + \lambda_m^{(4)} \tau_0$, $\mathbf{F}_k^m = \eta \tau_0 (\mathbf{g}_{k,m} \mathbf{g}_{k,m}^H) \forall m \in M$, and $\mathbf{O}_m^*$ as the optimal matrix which minimizes the $\mathcal{L}(\{\lambda_k\}_{k=1}^K, \{\lambda_m^{(3)}\}_{m=1}^M, \{\lambda_m^{(4)}\}_{m=1}^M, \{\mathbf{O}_m\}_{m=1}^M)$. To make sure the dual function $\inf_{\{\mathbf{O}_m \geq 0\}_{m=1}^M} \mathcal{L}(\cdot)$ bounded below, we have $\mathbf{A}_m \geq 0$ and

$$\text{Tr}(\mathbf{A}_m \mathbf{O}_m^*) = 0 \quad \forall m \in M. \tag{30}$$

Therefore, it can be verified that $\text{rank}(\mathbf{A}_m) + \text{rank}(\mathbf{O}_m^*) = N$.

Note that $\mathbf{A}_m \geq 0$ implies that

$$\lambda_m \geq \lambda_{\max} \left(\sum_{k \in \mathcal{K}} \lambda_k \mathbf{F}_k^m \right) \quad \forall m \in M \tag{31}$$

where $\lambda_{\max}(\cdot)$ represents the largest eigenvalue of the matrix. Due to (30), $\mathbf{A}_m$ must be singular since $\mathbf{O}_m^* \geq 0$ and $\mathbf{O}_m^* \neq 0$. Therefore, we have

$$\lambda_m = \lambda_{\max} \left(\sum_{k \in \mathcal{K}} \lambda_k \mathbf{F}_k^m \right) \quad \forall m \in M. \tag{32}$$

The rank of $\mathbf{A}_m$ depends on the value of λ_k . Obviously, to make the problem feasible, at least one λ_k must be greater than 0. Otherwise, $\text{rank}(\mathbf{A}_m) = N$ and the problem is infeasible.

Without loss of generality, we assume that $\{\lambda_k\}_{k=1}^{K'}$ are larger than zero and other λ_k s are zero. Therefore,

$$\mathbf{A}_m = \lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m - \lambda_1 \mathbf{F}_1^m \quad \forall m \in M. \tag{33}$$

From (33), we can get

$$\lambda_m = \lambda_{\max} \left(\sum_{k=1}^{K'} \lambda_k \mathbf{F}_k^m \right) \quad \forall m \in M. \tag{34}$$

When (34) holds, we have $\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m \geq 0$ according to the Weyl theorem. Next, we will prove by contradiction that $\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m > 0$. If $\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m > 0$ does not hold, then $\lambda_m = \lambda_{\max}(\sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m) \quad \forall m \in M$. Denoting $\mathbf{f}_{\max}$ as the eigenvector of $\sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m$ corresponding to $\lambda_{\max}(\sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m)$, we have

$$\begin{aligned}
& \lambda_m + \mathbf{f}_{\max}^H \lambda_1 \mathbf{F}_1^m \mathbf{f}_{\max} \\
& = \mathbf{f}_{\max}^H \left(\sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m + \lambda_1 \mathbf{F}_1^m \right) \mathbf{f}_{\max} \\
& \leq \lambda_{\max} \left(\sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m + \lambda_1 \mathbf{F}_1^m \right) = \lambda_m
\end{aligned} \tag{35}$$

which results in

$$\mathbf{f}_{\max}^H \lambda_1 \mathbf{F}_1^m \mathbf{f}_{\max} = 0 \Leftrightarrow \mathbf{f}_{\max}^H \mathbf{g}_{1,m} = 0. \tag{36}$$

Based on the definition of $\mathbf{F}_k$, we know that

$$\mathbf{f}_{\max} \in \text{span}\{\mathbf{g}_{2,m}, \dots, \mathbf{g}_{K',m}\}. \tag{37}$$

In the channel model, we assume that channel vectors of different terminals are independent random vectors, so $\Pr(\mathbf{f}_{\max}^H \mathbf{g}_{1,m} = 0) = 0$. Therefore, the assumption that $\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m$ is not full rank leads to contradiction. Hence, $\text{rank}(\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m) = N$. Because $\text{rank}(\mathbf{F}_1^m) = 1$, we have $\text{rank}(\mathbf{A}_m) > \text{rank}(\lambda_m \mathbf{I} - \sum_{k=2}^{K'} \lambda_k \mathbf{F}_k^m) - \text{rank}(\mathbf{F}_1^m) = N - 1$. This implies that $\text{rank}(\mathbf{O}_m^*) = 1$.

REFERENCES

- [1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
- [2] H. Ju and R. Zhang, "Throughput maximization in wireless powered communication networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 1, pp. 418–428, Jan. 2014.
- [3] K. Chi, Y.-H. Zhu, Y. Li, L. Huang, and M. Xia, "Minimization of transmission completion time in wireless powered communication networks," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1671–1683, Oct. 2017.
- [4] Z. Yang, W. Xu, Y. Pan, C. Pan, and M. Chen, "Optimal fairness-aware time and power allocation in wireless powered communication networks," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 3122–3135, Jul. 2018.
- [5] Q. Wu, M. Tao, D. W. K. Ng, W. Chen, and R. Schober, "Energy-efficient resource allocation for wireless powered communication networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 2312–2327, Mar. 2016.
- [6] Z. Hadzi-Velkov, I. Nikoloska, H. Chingoska, and N. Zlatanov, "Opportunistic scheduling in wireless powered communication networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 4106–4119, Jun. 2017.
- [7] H. Lee, K.-J. Lee, H. Kim, B. Clerckx, and I. Lee, "Resource allocation techniques for wireless powered communication networks with energy storage constraint," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2619–2628, Apr. 2016.
- [8] S. Bi, Y. J. Zhang, and R. Zhang, "Distributed scheduling in wireless powered communication network: Protocol design and performance analysis," in *Proc. 15th Int. Symp. Model. Optim. Mobile Ad Hoc Wireless Netw. (WiOpt)*, May 2017, pp. 1–7.
- [9] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
- [10] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
- [11] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.
- [12] L. Ji and S. Guo, "Energy-efficient cooperative resource allocation in wireless powered mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4744–4754, Jun. 2019.
- [13] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.
- [14] S. Mao, S. Leng, J. Hu, and K. Yang, "Utility-optimal resource allocation in energy harvesting powered C-RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jul. 2018, pp. 1–6.
- [15] X. Li, Z. Chen, A. Laha, Z. Chen, Y. Cheng, and L. Cai, "A hybrid approach for efficient wireless information and power transfer in green C-RAN," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jul. 2018, pp. 1–5.
- [16] C. Qin, W. Ni, H. Tian, and R. P. Liu, "Fronthaul load balancing in energy harvesting powered cloud radio access networks," *IEEE Access*, vol. 5, pp. 7762–7775, 2017.
- [17] L. Wang, H. Shao, J. Li, X. Wen, and Z. Lu, "Optimal multi-user computation offloading strategy for wireless powered sensor networks," *IEEE Access*, vol. 8, pp. 35150–35160, 2020.
- [18] Y. Teng, K. Cheng, Y. Zhang, and X. Wang, "Mixed-timescale joint computational offloading and wireless resource allocation strategy in energy harvesting multi-MEC server systems," *IEEE Access*, vol. 7, pp. 74640–74652, 2019.
- [19] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
- [20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2004.
- [21] K.-Y. Wang, A. M.-C. So, T.-H. Chang, W.-K. Ma, and C.-Y. Chi, "Outage constrained robust transmit optimization for multiuser MISO downlinks: Tractable approximations by conic optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 21, pp. 5690–5705, Nov. 2014.
- [22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.



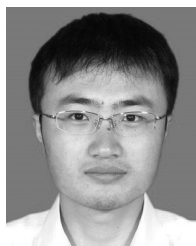
Ruijing Li received the B.S. degree in electronic information engineering from Harbin Engineering University, Harbin, China, in 2017. She is currently pursuing the Ph.D. degree in information and communication engineering with Nanjing University, Nanjing, China.

Her current research interests include image processing, edge computing, wireless communication energy transmission, and reinforcement learning.



Lan Tang (Member, IEEE) received the Ph.D. degree in communication and information science from Southeast University, Nanjing, China, in 2009.

She has been with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, since 2009, where she is currently an Associate Professor. She was a Visiting Researcher with Columbia University, New York, NY, USA, in 2014. Her current research interests include optimization and machine learning in wireless communications, and statistical signal processing applied for radar and communications.



Yechao Bai (Member, IEEE) received the B.S. and Ph.D. degrees in electronic science and engineering from Nanjing University, Nanjing, China, in 2005 and 2010, respectively.

He has been with Nanjing University, since 2010, where he is currently an Associate Professor with the School of Electronic Science and Engineering. His current research interests include array signal processing, target detection, and parameter estimation.



Mengting Lou received the B.S. and M.S. degrees from Beijing Jiaotong University, Beijing, China, in 2015 and 2019, respectively.

She is a member of Technical Staff with the Future Research Laboratory, China Mobile Research Institute, Beijing. Her current research interests include wireless communications, integrated sensing and communication, virtual multiinput–multioutput, sparse array, and coordinated multipoint transmission.



Xinggan Zhang received the B.S. degree in electrical engineering and the S.M. and Ph.D. degrees from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 1982, 1988, and 2001, respectively.

He joined the Department of Electronic Engineering, NUAA, in 1992, where he was an Associate Professor. He joined the Department of Electronic Science and Engineering, Nanjing University, in 1999, where he is currently a Professor. He has presided over and undertaken more than 30 National Natural Science Foundation of China and Jiangsu Province Science and Technology Plan Projects. He has published more than 100 academic papers and applied for/authorized more than 20 national invention patents. His research interests include target recognition and image processing.