Programmable Hierarchical C-RAN: From Task Scheduling to Resource Allocation

Wenchao Xia, *Student Member, IEEE*, Tony Q. S. Quek, *Fellow, IEEE*, Jun Zhang, *Member, IEEE*, Shi Jin, *Member, IEEE*, and Hongbo Zhu

***Abstract*— Traffic delay is a key metric to measure the quality-of-service of next-generation wireless communication networks. In this paper, we consider a cloud radio access network architecture with a hierarchical structure of virtual controllers and multiple clusters of remote radio heads (RRHs). A high-level controller coordinates control plane decisions among local controllers and each local controller is in charge of a cluster of RRHs. Moreover, each local controller is equipped with one server for creating virtual machines (VMs) to execute the users' baseband processing tasks. Then, under the considered architecture, we aim to minimize the average delay consisting of task execution delay and signal transmission delay under total power constraint, by joint optimization of task scheduling and resource allocation, including VM allocation and RRH assignment. Due to the non-deterministic polynomial-time hardness (NP-hardness) of the joint optimization problem, we translate it into a matroid constrained submodular maximization problem and propose heuristic algorithms to find solutions with 0.5-approximation. Besides, both centralized and distributed control schemes are considered. In the centralized control scheme, all decisions about task scheduling, VM allocation, and RRH assignment are made in the high-level controller. But in the distributed control scheme, the high-level controller is only in charge of task scheduling based on graph theory and the local controllers are responsible for their respective VM allocation and RRH assignment. The simulation results show that the proposed algorithms can achieve better performance than the separate optimization of VM allocation and RRH assignment.**

Manuscript received April 9, 2018; revised July 18, 2018, October 12, 2018 and December 27, 2018; accepted February 14, 2019. Date of publication March 5, 2019; date of current version March 11, 2019. The work of W. Xia, J. Zhang, and H. Zhu was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U1805262, Grant 61671251, and Grant 61871446, and in part by the Nanjing University of Posts and Telecommunications Start-Up Foundation under Grant NY215004. The work of T. Q. S. Quek was supported by the SUTD-ZJU Research Collaboration under Grant SUTD-ZJU/RES/01/2016 and Grant SUTD-ZJU/RES/05/2016. The work of S. Jin was supported by NSFC under Grant 61625106 and Grant 61531011. This paper was presented in part at the IEEE/CIC International Conference on Communications in China, Beijing, China, August 2018 [1]. The associate editor coordinating the review of this paper and approving it for publication was X. Zhou. *(Corresponding authors: Hongbo Zhu; Jun Zhang)*

W. Xia, J. Zhang, and H. Zhu are with the Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, and also with the Engineering Research Center of Health Service System Based on Ubiquitous Wireless Networks, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 2015010203@njupt.edu.cn; zhangjun@njupt.edu.cn; hbz@njupt.edu.cn).

T. Q. S. Quek is with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore 487372 (e-mail: tonyquek@sutd.edu.sg).

S. Jin is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: jinshi@seu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2019.2901684

Index Terms— **Cloud radio access networks, software-defined networking, delay, submodular, resource allocation, task scheduling.**

I. Introduction

IT IS envisioned that the exponentially growing data traffic leads to the demand for higher capacity density in radio access networks (RANs) [2], [3]. To meet such demands, one of candidate solutions is to make the network infrastructure dense and heterogeneous [4]. By deploying small cells in areas with a lot of traffic concentrated, dense networks allow for a higher spatial efficiency of radio resource and also a potential that network capacity increases in proportion to the number of base stations [5]. However, the dense deployment of small cells brings new challenges to RANs, such as the increased signaling and management overhead and the more complex interference management. Scalability and energy efficiency are also the key problems that should be concerned.

A novel network architecture named cloud RANs (C-RANs) suggested by mobile operators has attracted wide attention recently. In C-RANs, cloud-computing technique is applied and baseband units (BBUs) of all base stations are aggregated into a centralized BBU pool for signal processing, whereas radio frequency (RF) functions are implemented at low-cost remote radio heads (RRHs). The operations in the physical layer, such as coding, modulation, and precoding for the downlink transmission can be integrated as a task of each user that is executed in the BBU pool. Similarly, the operations such as decoding, demodulation, channel estimation, and signal detection for the uplink transmission can also be treated as a processing task. These users' tasks can even include some operations of upper layers. RRHs are placed closer to users and BBUs are stacked into a hotel, thus transmit power is reduced and energy efficiency is improved significantly.

At the same time, the software defined networking (SDN) concept is introduced in C-RANs to tackle network deployment and management issues [3]–[9]. SDN is a centralized control paradigm enabled by OpenFlow protocol and supports programmability of network protocols [5], [7]. SDN decouples the control plane from the data plane in network elements and centralizes the functions of the control plane in a software-based controller with a stringent control plane. Thus C-RANs can make decisions about resource allocation and interference management from a global view and avoid decisional overhead at data plane elements [8]. In summary, C-RANs based on SDN not only separate the control plane

from the data plane, but also decouple baseband processing from RF transmission in the data plane. However, such benefits come with a cost and one major challenge is scalability of the control plane [9]. As the number of the data plane elements distributed over a designated area is very large, too much overhead for computation and communication will be incurred by the centralized controller. Besides, the inherent control latency from the centralized controller to the data plane elements is high and the control latency problem becomes more severe as the density of networks increases.

To address these issues, inspired by [3], [8], and [9]. we consider a new C-RAN architecture with a hierarchical structure of virtual controllers. In the hierarchical C-RANs, RRHs are divided into several clusters according to their geographic locations and each cluster is assigned a virtual local controller, as well as a BBU pool (It can be a server in practice) for signal processing. A high-level controller in the network core is responsible for coordinating control plane decisions among the local controllers. Such a hierarchical structure of virtual controllers improves flexibility and scalability of C-RANs. Specifically, the high-level controller can make network-wide decisions according to the information collected from the local controllers, but also can offload a part of its control tasks to the local controllers, including those requiring fast response (e.g., real-time applications) and those based on local information (e.g., intra-cluster interference management and cluster-wide handover). Meanwhile, these local controllers can make cluster-wide decisions or follow the commands from the high-level controller. Besides, each RRH handles local decisions that do not affect other neighboring RRHs [4]. Offloading control tasks to local controllers and making local decisions at the network edge can alleviate the control latency and improve responsiveness [9]. Also, scalability and flexibility performances are improved compared to conventional C-RANs.

With the popularity of online videos and mobile games, as well as the development of the Internet of things, traffic delay is considered as a key metric to measure the quality-of-service (QoS). Take the application of online video downloads as an example. The application server prepares the data requested by users and the SDN controller should choose a server with more computation resource available and some RRHs that are closer to users for signal processing and transmission, respectively, because online videos usually have a large amount of data. The SDN controller should take the QoS requests and resource limitations into account and make control decisions about task scheduling and resource allocation, so that users can finish video downloads as early as possible. Although the hierarchical C-RANs can alleviate the latency of the control plane, the delay of the data plane is still a challenge that the hierarchical C-RANs have to deal with. The key to addressing this challenge is to jointly optimize task scheduling and resource allocation including computation and transmission resources, because a successful downlink communication process in a hierarchical C-RAN includes the following steps: Before transmission in the downlink, each user has one task that first is scheduled on a server and then virtual machines (VMs) are created by the server to execute this task. Finally the output data is transmitted using RRHs to users. Actually, transmission resource allocation has been studied in a lot of works [10]–[14]. These works focused on physical layer issues such as power minimization problem and achievable rate maximization problem by joint optimization of precoding design, power allocation, user association, and so on. There have also been many works investigating the joint allocation problem of transmission and computation resources, e.g., [15]–[18], and most of these works used queueing to model data processing and transmission behavior. However, these works did not take task scheduling into consideration because all RRHs were connected to the same BBU pool for signal processing in their system models. In the proposed hierarchical C-RANs, RRHs are divided into several clusters and each cluster has a different server. Task scheduling has an important impact on system performance. For example, if users' tasks are scheduled on servers which have low execution efficiency or are short of computation resource, then the execution delay is large. Similarly, if users are scheduled into clusters whose RRHs have poor channel condition, then the transmission delay is large. Motivated by these facts, this work aims to minimize the system delay by optimizing task scheduling and resource allocation including transmission and computation resources simultaneously. Furthermore, since the resource allocation problems, as well as the scheduling problems, are usually formulated as an integer/mixed-integer programming whose optimal solution is hard to find, we introduce a mathematical tool named submodular function maximization for such problems.

Submodular function maximization is a powerful tool to solve combinatorial problems and it has a lot of applications in network issues [19]–[28]. For example, submodular function maximization was used to find solutions to cache placement problems in [19]–[22]. The works in [23] and [27] aimed to solve the wireless network deployment problems with different optimization goals which could be modeled as a submodular set function. And this tool can also be applied to solve the joint problem of base station operation and user association in cellular networks, which was investigated in [24]–[26]. Besides, the task offloading problem in proximate clouds with submodular function maximization was considered in [28]. Inspired by these works, we exploit submodularity of the joint problem of task scheduling and resource allocation in the hierarchical C-RAN and propose some heuristic algorithms to find suboptimal solutions with guaranteed approximation [29].

The main contributions of this work are listed as follows.

- We consider a C-RAN architecture with a hierarchical structure of virtual controllers. In the hierarchical C-RANs, the RRHs are formed into several clusters according to their geographic locations and each cluster is allocated a server for signal processing. There is also a local controller assigned to each cluster and the high-level controller is responsible for coordinating control decisions among local controllers from a global view.
- Aiming to minimize the average system delay consisting of transmission delay and processing delay, we formulate a joint optimization problem of task scheduling and resource allocation in the proposed hierarchical C-RAN,

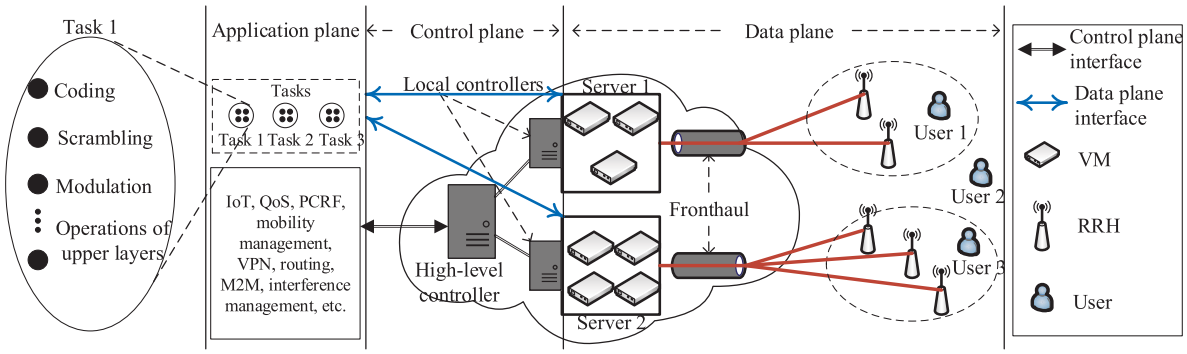


Fig. 1. A typical structure of hierarchical C-RANs in the downlink. In the hierarchical C-RAN, virtual machines (VMs) are created by servers to execute users' tasks and then the output data is transmitted to users by remote radio heads (RRHs). The application plane consists of one or more applications, such as Internet of things (IoT), quality of service (QoS), policy and charging rules function (PCRF), virtual private network (VPN), machine-to-machine (M2M) communication, interference management, etc.

which is an integer programming. In the joint problem, resource for allocation includes not only transmission resource but also computation resource for signal processing.

- By exploiting submodularity, we reformulate the joint problem as a submodular function maximization problem. Then we propose two heuristic algorithms for both centralized and distributed control schemes, which can find solutions with 0.5-approximation [29], [30].
- We develop both centralized and distributed control schemes for the joint optimization problem. In the centralized control scheme, the high-level controller makes all decisions about task scheduling and resource allocation and then deliver the control commands to all local controllers. But in the distributed control scheme, only the decisions about task scheduling are made at the high-level controller using graph theory. After task scheduling is determined, each local controller makes decisions about resource allocation for their respective users. The centralized control scheme can achieve lower average delay than the distributed control scheme, but with higher complexity and more overhead.

The remainder of this paper is organized as follows. Section II introduces the system model and formulates a delay minimization problem in the centralized control scheme. Section III reformulates the delay minimization problem in the centralized control scheme as a matroid-constrained monotone submodular maximization problem and proposes a greedy algorithm. Section IV analyses the delay issue in the distributed control scheme and also applies graph theory to task scheduling. Numerical results are presented in Section V. Finally, conclusion is drawn in Section VI.

II. System Model

Consider a downlink C-RAN which includes a hierarchical structure of virtual controllers in the center and multiple clusters of RRHs with an index set $\mathcal{K} = \{1, 2, \ldots, K\}$, as shown in Fig. 1. A high-level controller coordinates control plane decisions among K local controllers and each local controller $k \in \mathcal{K}$ is in charge of the RRHs in cluster k. In addition, each local controller k is equipped with a server k that is responsible for creating a set $\mathcal{V}_k$ of VMs to execute users' tasks. Each user's task is an integration of some operations in the physical layer such as coding, scrambling, and modulation. These tasks can also include some operations of upper layers. Due to the resource limit, each server can support at most N VMs to run at the same time, i.e., $|\mathcal{V}_k| \leq N, \forall k \in \mathcal{K}$. Note that the operator $|\bullet|$ represents the cardinality of a set. The set $\mathcal{R}_k$ of $L_k = |\mathcal{R}_k|$ RRHs in cluster k is connected to server k via high-speed fronthaul links. There is a set of single-antenna users, indexed by $\mathcal{U} = \{1, 2, \ldots, U\}$, each with a task $u \in \mathcal{U}$ to execute before its output data is transmitted. Here, we assume that all the VMs have the same computation capability as ζ CPU cycles per second and each user's task is dividable such that allocating more VMs to the same task can save more time. Besides, these users are assumed to be static or have low mobility. We describe each task u with a tuple $\langle \rho_u, \Phi_u \rangle$ where ρ_u represents the load of task u in the form of the total number of CPU cycles required to accomplish the computation task u and Φ_u is the amount of output data after task u is finished. Such a task model is used commonly [28], where ρ_u and Φ_u depend on the types of applications and can be obtained by the methods in [31]–[33].

In the following, we introduce more implementation details of the system model working in the centralized control scheme, i.e., the high-level controller makes all decisions about task scheduling and resource allocation and then delivers the control commands to all local controllers.

A. Execution Delay

In the centralized control scheme, before task execution, the high-level controller should specify a target server for each task and decide how many VMs are allocated to them, as well as how many RRHs are used to transmit their output data. Then the local controllers, following the control commands from the high-level controller, manage to allocate VM and RF resource. We define $x_{kv}^u, u \in \mathcal{U}, k \in \mathcal{K}, v \in \mathcal{V}_k$, as the VM allocation plan with $x_{kv}^u = 1$ if VM v created on server k is allocated to task u, and $x_{kv}^u = 0$ otherwise. We also define $\mathcal{S}_u = \{k : x_{kv}^u = 1, k \in \mathcal{K}, v \in \mathcal{V}_k\}$ as the index set of the servers that provide service for task u. Note that each task can be assigned to at most one server due to the separation

of different servers and each VM can be allocated to at most one task in an allocation period, i.e., $|\mathcal{S}_u| \leq 1, \forall u \in \mathcal{U}$ and $\sum_{u\in\mathcal{U}} x_{kv}^u \leq 1, \forall k \in \mathcal{K}, \forall v \in \mathcal{V}_k$. For simplicity, we assume that each task can be divided into sub-tasks proportionally according to the execution efficiency of VMs assigned to itself, so that the execution time of task u is given by

$$\tau_u^{(EX)} = \begin{cases} \dfrac{\rho_u}{\zeta \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} x_{kv}^u \lambda_{kv}^u}, & \text{if } |\mathcal{S}_u| = 1, \\ \tau_0^{(C)}, & \text{otherwise}, \end{cases} \tag{1}$$

where λ_{kv}^u is the efficiency of executing task u with VM v on server k and the constant $\tau_0^{(C)}$ ($\tau_0^{(C)} > \tau_u^{(EX)}, \forall u \in \mathcal{U}$ with $|\mathcal{S}_u| = 1$) represents a larger delay including not only execution time but also waiting time in the case where no VM is allocated to task u in this allocation period. For example, the task may be executed in next VM allocation period. Note that the efficiency of different VMs execute the same task can be different, although these VMs have the same computation capability. This is because other types of resource in different VMs, e.g., block storage, message queues, interfaces, and memory, can be heterogeneous.

B. Transmission Delay

After task execution is finished, the output data will be transmitted to the corresponding users via RRHs. These RRHs work at the same frequency band. We use the binary variable $y_{kr}^u, u \in \mathcal{U}, k \in \mathcal{K}, r \in \mathcal{R}_k$, to denote the connection between the users and RRHs, with $y_{kr}^u = 1$ indicating user u is served by RRH r in cluster k and $y_{kr}^u = 0$ otherwise. Then the achievable transmission rate is given by [34]

$$R_u = B \log(1 + \sum_{k\in\mathcal{K}, r\in\mathcal{R}_k} y_{kr}^u |h_{kr}^u|^2 \gamma_{kr}), \tag{2}$$

where $\gamma_{kr} = \frac{P_{kr}}{\sigma^2 B + \chi}$ denotes the target signal-to-interference-plus-noise ratio at the transmitter side, P_{kr} is the average transmit power at RRH r in cluster k, B is the system bandwidth, σ^2 is the noise power spectral density, χ is the average interference power controlled by interference management schemes[1] [34], [35], and h_{kr}^u denotes the channel coefficient between user u and RRH r in cluster k, which is composed of large-scale fading and small-scale fading where the large-scale fading is caused by path loss and shadow fading. We define $\mathcal{W}_u = \{k : y_{kr}^u = 1, k \in \mathcal{K}, r \in \mathcal{R}_k\}$ as the index set of the clusters whose RRHs serve user u. Note that each user cannot be served by the RRHs across different clusters simultaneously and the computation processing and signal transmission of each task must be in the same cluster, i.e., $|\mathcal{W}_u| \leq 1$ and $\mathcal{W}_u = \mathcal{S}_u$. Then the average transmission time of user u is computed as

$$\tau_u^{(TR)} = \begin{cases} \dfrac{\Phi_u}{\mathbb{E}_{\mathbf{h}}(R_u)}, & \text{if } \mathcal{W}_u = \mathcal{S}_u \neq \emptyset, \\ \tau_1^{(C)}, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathbb{E}_{\mathbf{h}}(\bullet)$ is used to evaluate the ergodic transmission rate under the assumption that the channel coefficients are identically and independently distributed, and the constant $\tau_1^{(C)}$ ($\tau_1^{(C)} > \tau_u^{(TR)}, \forall u \in \mathcal{U}$ with $\mathcal{W}_u = \mathcal{S}_u \neq \emptyset$) represents a larger delay including not only transmission time but also waiting time in the case where no RRH is assigned to user u in the current transmission period. For example, the signal of user u may be transmitted in next transmission period. In this work, we jointly optimize the allocation of VMs and RRHs where the RRH assignment problem is a fast time-scale issue because it depends on small-scale fading which varies in the order of milliseconds. However, the VM allocation problem is a slow time-scale issue since the VM allocation is usually executed much slower than milliseconds. To deal with the time-scale challenge, we evaluate the average transmission time with the ergodic transmission rate instead of the instantaneous transmission rate. Monte Carlo averaging over channels with respect to small-scale fading, denoted as $\mathbb{E}_{\mathbf{h}}(R_u)$, is adopted to calculate the ergodic transmission rate. Then the joint allocation problem becomes a slow time-scale issue which is suboptimal. Note that Monte Carlo averaging over channels with respect to small-scale fading needs a lot of channel samples, which causes much communication overhead and high computational complexity. Furthermore, when each RRH has more than one antenna, the ergodic transmission rate can be approximated with a deterministic result according to large system analysis, which is only dependent on statistical channel information instead of small-scale fading [36]. Therefore, the communication and computation overhead can be reduced significantly.

C. Problem Formulation

In this work, we aim to minimize the average delay cost of all users including execution delay and transmission delay under the power constraint of all running VMs. We assume that each VM has the same power consumption when they process tasks since they have the same computation capability. Therefore, the power constraint of VMs can be simplified as the number constraint of VMs running at the same time. Then the underlaying optimization problem to realize the proposed strategy is formulated as

$$\min_{\mathbf{X},\mathbf{Y}} \ \frac{1}{U} \sum_{u\in\mathcal{U}} \alpha_u \tau_u \tag{4a}$$

$$\text{s.t.} \ \sum_{u\in\mathcal{U}} \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} x_{kv}^u \leq V^{(MAX)}, \tag{4b}$$

$$\sum_{u\in\mathcal{U}} x_{kv}^u \leq 1, \quad \forall k \in \mathcal{K}, \ \forall v \in \mathcal{V}_k, \tag{4c}$$

$$\sum_{u\in\mathcal{U}} y_{kr}^u \leq C_{kr}, \quad \forall k \in \mathcal{K}, \ \forall r \in \mathcal{R}_k, \tag{4d}$$

$$|\mathcal{S}_u| \leq 1, \quad |\mathcal{W}_u| \leq 1, \ \mathcal{S}_u = \mathcal{W}_u, \ \forall u \in \mathcal{U}, \tag{4e}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the collections of x_{kv}^u's and y_{kr}^u's, respectively, $\tau_u = \tau_u^{(EX)} + \tau_u^{(TR)}$ is the total delay of user u, $\alpha_u > 0$ is the positive weight for user u that characterizes the priority of its computation task, and C_{kr} is the maximum number of active connection links that RRH r in cluster k

[1] In the following, we assume that χ is fixed by applying intelligent interference management schemes that are able to adapt according to the number of clusters and user number, no matter of the distribution of RRHs.

can support at the same time. Note that these α_u values are designated by the applications in the application plane and forwarded to the controllers in the control plane. Thus the weight values of users are taken into account when the high-level controller makes decisions. In problem (4), the first constraint (4b) represents the power budget for all the running VMs simultaneously and the last constraint (4e) indicates that the computational processing and signal transmission of each task should be in the same cluster, but also each task can be scheduled on at most one server.

The integer programming (4) is NP-hard, whose global optimal solution with exponential complexity is usually impractical to find. Hence, we first give an equivalent expression of problem (4) as:

$$\max_{\mathbf{X},\mathbf{Y}} \sum_{u\in\mathcal{U}} D_u \tag{5a}$$

$$\text{s.t. (4b)} - \text{(4e)}, \tag{5b}$$

where $D_u = \alpha_u(\tau_0^{(C)} + \tau_1^{(C)} - \tau_u)$ in the objective function aims to maximize the delay cost reduction. Then, we reformulate problem (5) as a submodular function maximization problem in next section since submodular optimization is a powerful tool for solving combinatorial problems [29] and it has a lot of applications in network issues [19]–[27].

III. Submodular Optimization

We first provide the basic definitions of matroid and submodular functions [29], [37].

A. Preliminaries

Definition 1. Let $\mathcal{C}$ be a finite set and $\mathcal{I} \subseteq 2^{\mathcal{C}}$ be a collection of subsets of $\mathcal{C}$. Then the set family $\mathcal{M} = (\mathcal{C}, \mathcal{I})$ is a matroid if the following conditions are satisfied

- $\mathcal{I} \neq \emptyset$;
- If $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{C}$ and $\mathcal{A}' \in \mathcal{I}$, then $\mathcal{A} \in \mathcal{I}$;
- If $\mathcal{A}, \mathcal{A}' \in \mathcal{I}$ and $|\mathcal{A}| < |\mathcal{A}'|$, then there exists $a \in \mathcal{A}' \setminus \mathcal{A}$, such that $\mathcal{A} \cup \{a\} \in \mathcal{I}$.

Definition 2. Define a set function $f : 2^{\mathcal{C}} \to \mathbb{R}, \mathcal{A} \subseteq \mathcal{C}, a \in \mathcal{C}$ and let $\Delta_f(a|\mathcal{A}) = f(\mathcal{A} \cup \{a\}) - f(\mathcal{A})$ be the discrete derivative of f with respect to element a. Then f is submodular if each $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{C}$ and $a \in \mathcal{A}' \setminus \mathcal{A}$ satisfy the following constraint

$$\Delta_f(a|\mathcal{A}) > \Delta_f(a|\mathcal{A}'). \tag{6}$$

Equivalently, f is submodular if for each $\mathcal{A}, \mathcal{A}' \subseteq \mathcal{C}$,

$$f(\mathcal{A}' \cup \mathcal{A}) + f(\mathcal{A}' \cap \mathcal{A}) \leq f(\mathcal{A}') + f(\mathcal{A}). \tag{7}$$

Submodularity has an intuitive interpretation as the property of diminishing returns. The gain resulting from adding a new element to the set will decrease as the set becomes larger. In addition, f is said to be monotone if $f(\mathcal{A}) \leq f(\mathcal{A}'), \forall \mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{C}$.

B. Proposed Algorithm

We first define the VM allocation ground set and the RRH assignment ground set as

$$\mathcal{A} = \{\tilde{x}_{11}^1, \ldots, \tilde{x}_{11}^U, \tilde{x}_{12}^1, \ldots, \tilde{x}_{12}^U, \ldots, \tilde{x}_{KN}^1, \ldots, \tilde{x}_{KN}^U\},$$

and

$$\mathcal{B} = \{\tilde{y}_{11}^1, \ldots, \tilde{y}_{11}^U, \tilde{y}_{12}^1, \ldots, \tilde{y}_{12}^U, \ldots, \tilde{y}_{KL_K}^1, \ldots, \tilde{y}_{KL_K}^U\},$$

respectively, where $\tilde{x}_{kv}^u$ denotes the action that VM v on server k is allocated to task u and $\tilde{y}_{kr}^u$ denotes the action that RRH r in cluster k serves user u. The ground set $\mathcal{A}$ can be partitioned into $N' = KN$ disjoint sets, i.e., $\mathcal{A} = \bigcup_{n=1}^{N'} \mathcal{A}_n$ and $\mathcal{A}_n \bigcap \mathcal{A}_{n'} = \emptyset$ for any $n \neq n'$, where $\mathcal{A}_n = \{\tilde{x}_{kv}^1, \ldots, \tilde{x}_{kv}^U\}$ with $n = (k-1)N + v$. Also, the ground set $\mathcal{B}$ can be divided into $L = \sum_{k\in\mathcal{K}} L_k$ disjoint sets, i.e., $\mathcal{B} = \bigcup_{l=1}^{L} \mathcal{B}_l$ and $\mathcal{B}_l \bigcap \mathcal{B}_{l'} = \emptyset$ for any $l \neq l'$, where $\mathcal{B}_l = \{\tilde{y}_{kr}^1, \ldots, \tilde{y}_{kr}^U\}$ with $l = \sum_{i=1}^{k} L_{i-1} + r$.

Then we also define a pair $\mathcal{M} = (\mathcal{C}, \mathcal{I})$ where $\mathcal{C} = \{\mathcal{A}, \mathcal{B}\}$ is the combinational ground set and $\mathcal{I} \subseteq 2^{\mathcal{C}}$ denoting a collection of independent subsets of $\mathcal{C}$ is defined as

$$\begin{aligned}\mathcal{I} = \Big\{ & \mathcal{X} \subseteq \mathcal{C} : |\mathcal{X} \bigcap \mathcal{A}_n| \leq 1, n = 1, \ldots, N', \\ & |\mathcal{X} \bigcap \mathcal{B}_l| \leq \bar{C}_l, l = 1, \ldots, L, \\ & |\mathcal{X} \bigcap \mathcal{A}| \leq V^{(MAX)}, \mathcal{S}_u(\mathcal{X} \bigcap \mathcal{A}) = \mathcal{W}_u(\mathcal{X} \bigcap \mathcal{B}), \\ & |\mathcal{S}_u(\mathcal{X} \bigcap \mathcal{A})| \leq 1, |\mathcal{W}_u(\mathcal{X} \bigcap \mathcal{B})| \leq 1, u \in \mathcal{U} \Big\},\end{aligned} \tag{8}$$

where $\bar{C}_l = C_{kr}$ with $l = \sum_{i=1}^{k} L_{i-1} + r$, $\mathcal{S}_u(\mathcal{X}) = \{k : \tilde{x}_{kv}^u \in \mathcal{X}, \tilde{x}_{kv}^u = 1\}$, and $\mathcal{W}_u(\mathcal{X}) = \{k : \tilde{y}_{kr}^u \in \mathcal{X}, \tilde{y}_{kr}^u = 1\}$. From (8), we find that the task that occupies VM v on server k is denoted by $\mathcal{X} \bigcap \mathcal{A}_n$ with $n = (k-1)N + v$ and the set of users who are served by RRH r in cluster k is denoted by $\mathcal{X} \bigcap \mathcal{B}_l$ with $l = \sum_{i=1}^{k} L_{i-1} + r$. In what follows we show the set family $\mathcal{M} = (\mathcal{C}, \mathcal{I})$ is a matroid in the following lemma:

Lemma 1: The set family $\mathcal{M} = (\mathcal{C}, \mathcal{I})$ is a matroid.

Proof: See Appendix A for reference. ■

Furthermore, based on **Lemma 1**, we have the following theorem to describe the property of the objective function in problem (5):

Theorem 1: The delay reduction function $D = \sum_{u\in\mathcal{U}} D_u$ is a monotone submodular function over $\mathcal{X} \in \mathcal{I}$.

Proof: See Appendix B for reference. ■

On the basis of **Lemma 1** and **Theorem 1**, by replacing the constraints in problem (5) with the pair $\mathcal{M} = (\mathcal{C}, \mathcal{I})$, then problem (5) can be reformulated as a matroid-constrained monotone submodular maximization problem:

$$\max_{\mathcal{X}} \sum_{u\in\mathcal{U}} D_u \tag{9a}$$

$$\text{s.t. } \mathcal{X} \in \mathcal{I}. \tag{9b}$$

A popular approach towards maximizing a monotone submodular function in the case of matroid constraints is the greedy algorithm [29], [30].

C. Greedy Algorithm for Centralized Control Scheme

The greedy algorithm for the centralized control scheme is shown in **Algorithm 1**. In this algorithm, we first initialize $\underline{\mathcal{A}}_n$'s and $\underline{\mathcal{B}}_l$'s, $n = 1, \ldots, N'$, $l = 1, \ldots, L$ to be the empty set $\emptyset$, and define $\underline{\mathcal{A}} = \bigcup_{n=1}^{N'} \underline{\mathcal{A}}_n$, $\underline{\mathcal{B}} = \bigcup_{l=1}^{L} \underline{\mathcal{B}}_l$, $\mathcal{X} = \{\underline{\mathcal{A}}, \underline{\mathcal{B}}\}$, and $\mathcal{Y} = \mathcal{C}$. Before staring iteration, the high-level controller first choose a couple $(\tilde{x}^u_{kv}, \tilde{y}^u_{kr})$ with the largest value $D_u(\tilde{x}^u_{kv}, \tilde{y}^u_{kr})$ for each user, such that we can achieve a beginning with global advantages. Then in each iteration, a new element $a^\star$ with the highest marginal gain is added into the set $\mathcal{X}$ and removed from the set $\mathcal{Y}$. This new element $a^\star$ can be either $\tilde{x}^u_{kv} \in \mathcal{A}$ or $\tilde{y}^u_{kr} \in \mathcal{B}$. In the case of $a^\star = \tilde{y}^u_{kr} \in \mathcal{B}$, the actions that allocate the VMs on other servers $k' \neq k$ to task u, as well as the actions that assign the RRHs in other clusters $k' \neq k$ to user u should be removed from the candidate action set, i.e.,

$$\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}_{n'}, \quad n' = 1, 2, \ldots, N', \\ n' \neq (k-1)N + 1, \ (k-1)N + 2, \ldots, kN, \quad (10)$$

and

$$\mathcal{Y} = \mathcal{Y} \setminus \mathcal{B}_{l'}, \quad l' = 1, \ldots, L, \\ l' \neq \sum_{i=1}^{k} L_{i-1} + 1, \ \sum_{i=1}^{k} L_{i-1} + 2, \ldots, \sum_{i=1}^{k} L_i. \quad (11)$$

In the case of $a^\star = \tilde{x}^u_{kv} \in \mathcal{A}$, in addition to (10) and (11), the actions that allocate VM v on server k to users should also be removed from the candidate action set, i.e., $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}_n$ with $n = (k-1)N + v$. Such iteration is repeated until the candidate action set is empty or the marginal gain is zero.

Algorithm 1 is guaranteed to produce a suboptimal solution with 0.5-approximation to the optimal value in general [38]. In **Algorithm 1**, given that U is a constant and calculating each marginal value takes $\mathcal{O}(1)$ time, thus the computational complexity of the for loop is estimated as $\mathcal{O}(KN + L)$. Then there are at most $(V^{(MAX)} + \sum_{l=1}^{L} \bar{C}_l - 2U)$ iterations and each iteration involves evaluating the marginal value of at most $(KN + L - 2U)$ elements which have not been included in $\mathcal{X}$. Actually, the number of the elements not included in $\mathcal{X}$ decreases as the iteration time increases. Therefore, the total computational complexity can be estimated as $\mathcal{O}(KN + L)$.

In the centralized control scheme, all the decisions including task scheduling, VM allocation, and RRH assignment are made in the high-level controller before task execution. Therefore, the CSI of RRHs in all clusters, as well as the information about computation resource of all servers, has to be collected and transmitted to the high-level controller, which causes much communication overhead and high computational complexity. To reduce the overhead and complexity, we can decouple the decisions into two parts. Specifically, the high-level controller is only responsible for specifying the target server for each task and then each local controller independently make the decisions about VM allocation and RRH assignment for their respective users. This scheme is referred to as distributed control scheme and we introduce it in the following section.

Algorithm 1 Greedy Algorithm for the Centralized Control Scheme

1: **Initialize:** Set $\underline{\mathcal{A}}_n = \emptyset, n = 1, \ldots, N'$, $\underline{\mathcal{A}} = \bigcup_{n=1}^{N'} \underline{\mathcal{A}}_1$, $\underline{\mathcal{B}}_l = \emptyset, l = 1, \ldots, L$, $\underline{\mathcal{B}} = \bigcup_{l=1}^{L} \underline{\mathcal{B}}_l$, $\mathcal{X} = \{\underline{\mathcal{A}}, \underline{\mathcal{B}}\}$, and $\mathcal{Y} = \mathcal{C}$.
2: **for** $u \in \mathcal{U}$ **do**
3: Choose a couple $(\tilde{x}^u_{kv}, \tilde{y}^u_{kr}) = \arg\max_{\tilde{x}^u_{k_1v'}, \tilde{y}^u_{k_2r'} \in \mathcal{Y}, k_1 = k_2} \Delta_D(\tilde{x}^u_{k_1v'}, \tilde{y}^u_{k_2r'} | \mathcal{X})$.
4: Update $\underline{\mathcal{A}}_n = \underline{\mathcal{A}}_n \cup \{\tilde{x}^u_{kv}\}$ with $n = (k-1)N + v$, $\underline{\mathcal{A}} = \underline{\mathcal{A}} \cup \{\tilde{x}^u_{kv}\}$, $\mathcal{X} = \mathcal{X} \cup \{\tilde{x}^u_{kv}\}$, $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}_n$, $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}_{n'}, n' = 1, 2, \ldots, N', n' \neq (k-1)N + 1, (k-1)N + 2, \ldots, kN$, and $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{B}_l, l = 1, \ldots, L, l \neq \sum_{i=1}^{k} L_{i-1} + 1, \sum_{i=1}^{k} L_{i-1} + 2, \ldots, \sum_{i=1}^{k} L_i$.
5: **if** $|\underline{\mathcal{A}}| = V_{MAX}$, **then** $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}$. **end if**
6: Update $\underline{\mathcal{B}}_l = \underline{\mathcal{B}}_l \cup \{\tilde{y}^u_{kr}\}$ with $l = \sum_{i=1}^{k} L_{i-1} + r$, $\mathcal{X} = \mathcal{X} \cup \{\tilde{y}^u_{kr}\}$, $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{B}_{l'}, l' = 1, \ldots, L, l' \neq \sum_{i=1}^{k} L_{i-1} + 1, \sum_{i=1}^{k} L_{i-1} + 2, \ldots, \sum_{i=1}^{k} L_i$, and $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{A}_{n'}, n' = 1, 2, \ldots, N', n' \neq (k-1)N + 1, (k-1)N + 2, \ldots, kN$.
7: **if** $|\underline{\mathcal{B}}_l| = \bar{C}_l, \forall l = 1, \ldots, L$, **then** $\mathcal{Y} = \mathcal{Y} \setminus \mathcal{B}_l$. **end if**
8: **end for**
9: **repeat:**
10: Choose an element $a^\star = \arg\max_{a \in \mathcal{Y}} \Delta_D(a | \mathcal{X})$.
11: **if** $a^\star = \tilde{x}^u_{kv} \in \mathcal{A}$ **then**
12: Repeat steps 4 and 5.
13: **end if**
14: **if** $a^\star = \tilde{y}^u_{kr} \in \mathcal{B}$ **then**
15: Repeat steps 6 and 7.
16: **end if**
17: **until** $\mathcal{Y} = \emptyset$ or $\Delta_D(a | \mathcal{X}) = 0$, then return $\mathcal{X}$.

IV. Distributed Control Scheme

In the distributed control scheme, the high-level controller does not need to know the CSI. However, some auxiliary information depending on specific methods is necessary to assist the high-level controller to make decisions about task scheduling. For example, in this work, we collect the information about all users' candidate serving clusters as the auxiliary information. We assume each user u has a set $\mathcal{K}_u \subseteq \mathcal{K}$ of candidate serving clusters, in which an element represents a cluster whose respective server and RRHs can provide processing and transmission service for task u. Since one cluster of RRHs is equipped with one server, $\mathcal{K}_u$ can also be regarded as the candidate server set of task u.

The selection of $\mathcal{K}_u$'s is critical since a wrong selection strategy may lead to the degradation of system performance. For example, if the RRHs in $\mathcal{K}_u$ have poor channel condition or the VMs created by servers in $\mathcal{K}_u$ have low execution efficiency, then the transmission delay or the execution delay is large. These $\mathcal{K}_u$'s can be determined according to different strategies based on the execution efficiency of VMs, the capacities of servers and RRHs, CSI, the proximity of users to RRHs, and so on. For simplicity, in this work we assume each $\mathcal{K}_u$ is determined by the weighted sum of average distance between RRHs and users and average execution efficiency, i.e.,

$$\max_{k \in \mathcal{K}} \left(\beta_1 \frac{1}{N} \sum_{v \in \mathcal{V}_k} \lambda^u_{kv} - \beta_2 \frac{1}{L^u_k} \sum_{r \in \mathcal{R}_k} d^u_{kr} \right), \quad (12)$$

where β_1 and β_2 are two weights, and L^u_k denotes the number of candidate RRHs of user u in cluster k constrained by

$0 \leq L_k^u \leq L_k$. Here, we assume a user can access to a candidate RRH if the user in the coverage of the RRH. The goal of (12) is to find a cluster whose RRHs are close to the target user and whose server can execute the target user's task with higher execution efficiency. Then, the cluster with a larger value of (12) has priority to being chosen as one of the candidate serving clusters of the target user.

The size of each $\mathcal{K}_u$ is pre-determined with a constraint K_C, i.e., $|\mathcal{K}_u| \leq K_C$. We assume a user can access to a cluster if the user can access to any RRH in this cluster. The process of determining $\mathcal{K}_u$ is described as follows. Each user u broadcasts some pilots to all its candidate RRHs to help them to achieve distance information. Each local controller calculates the priority value in (12) for the users that can access to its corresponding cluster, then transmit the auxiliary information about priority values to the high-level controller. Finally, the high-level controller chooses the first $|\mathcal{K}_u|$ candidate serving clusters for each user u in descending order of priority value according to the auxiliary information collected from all local controllers and store $\mathcal{K}_u$'s into a table. The updating of the table is triggered by the arrivals of users' tasks and mobility of users.

We define $\bar{\mathcal{K}} = \bigcup_{u\in\mathcal{U}} \mathcal{K}_u \in \mathcal{K}$ as the set of candidate serving clusters (servers) of all users (tasks). The high-level controller is assumed to schedule all tasks on different servers associated with the clusters in $\bar{\mathcal{K}}$ as evenly as possible, which can be represented in mathematical expression as

$$\min_{\{\mathcal{S}_u, u\in\mathcal{U}\}} \max_{k\in\bar{\mathcal{K}}} |\mathcal{U}_k| \tag{13a}$$

$$\text{s.t. } \mathcal{S}_u \subseteq \mathcal{K}_u, \quad |\mathcal{S}_u| = 1, \ \forall u \in \mathcal{U}, \tag{13b}$$

where $\mathcal{U}_k = \{u_k^1, u_k^2, \ldots, u_k^{|\mathcal{U}_k|}\}$ is the set of tasks that scheduled on server k, which can also be considered as the set of users that are served in cluster k. Note that in task scheduling problem (13), for simplicity, we just force the number of tasks scheduled on each server to be as evenly as possible. A better scheme should take task load, output data amount, and execution efficiency of VMs created on servers into account. After the task scheduling plan is determined, each local controller k allocates VM and RF resource to their respective users by solving the following problem:

$$\min_{\mathbf{X}_k, \mathbf{Y}_k} \sum_{u\in\mathcal{U}_k} \alpha_u \tau_u \tag{14a}$$

$$\text{s.t. } \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}_k} x_{kv}^u \leq V_k^{(MAX)}, \quad \forall v \in \mathcal{V}_k, \tag{14b}$$

$$\sum_{u\in\mathcal{U}} y_{kr}^u \leq C_{kr}, \quad \forall r \in \mathcal{R}_k, \tag{14c}$$

where $\mathbf{X}_k$ and $\mathbf{Y}_k$ are the collections of x_{kv}^u's and y_{kr}^u's, respectively, and $V_k^{(MAX)} \leq N$ is the maximal number of VMs that server k can provide in this allocation period.

In the following, we propose algorithms for the task scheduling problem of the high-level controller and the VM and RRH allocation problem of the local controllers, i.e., problem (13) and problem (14), respectively.

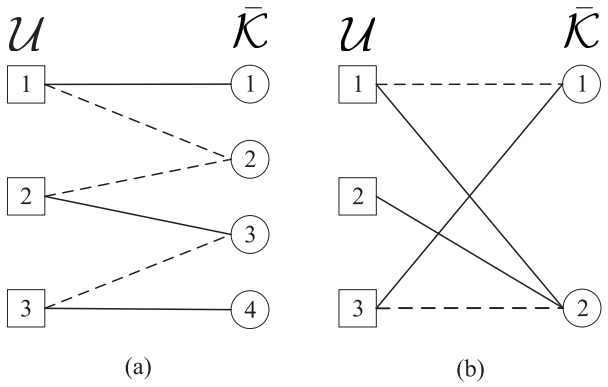


Fig. 2. Two examples of task scheduling are shown in (a) and (b) where $\mathcal{U}$ denotes the set of users and $\bar{\mathcal{K}}$ denotes the set of candidate serving clusters (servers) of all users (tasks). In both examples, an edge (represented by solid or dashed lines) indicates that the right vertex incident to this edge is the candidate serving cluster of the left vertex and the set of solid lines represents one of possible solutions. The solution in (a) is a maximum matching but not in (b).

A. Task Scheduling

We apply the bipartite graph model for problem (13) and define a bipartite graph ψ where the vertex set is composed of two disjoint sets $\mathcal{U}$ and $\bar{\mathcal{K}}$ and each edge joins a task in $\mathcal{U}$ to each of its candidate servers in $\bar{\mathcal{K}}$. For example, as shown in Fig. 2(a), servers 1 and 2 are the candidate servers of task 1. Then, the objective function of problem (13) can be interpreted as finding a set of edges that cover all vertices in $\mathcal{U}$ and make the degree of each vertex in $\bar{\mathcal{K}}$ as small as possible. Ideally, each task is scheduled on one different server and thus problem (13) can be solved by finding a maximum matching for bipartite graph ψ as shown in Fig. 2(a) and the set of solid lines is a maximum matching. However, according to Hall's "marriage" theorem [39], one-to-one scheduling can succeed with a necessary condition that for all $\mathcal{U}' \subseteq \mathcal{U}$, $|\mathbf{N}(\mathcal{U}')| \geq |\mathcal{U}'|$ where $\mathbf{N}(\mathcal{U}')$ denotes the set of vertices having a neighbor in $\mathcal{U}'$. This necessary condition does not usually hold as shown in Fig. 2(b) where tasks 1 and 2 are scheduled on server 2 simultaneously. Based on the above analysis, we propose a task scheduling algorithm based on Hungarian maximum matching method for problem (13) as shown in **Algorithm 2**.

Algorithm 2 includes two main processes: Find a maximum matching between $\mathcal{U}$ and $\bar{\mathcal{K}}$ based on Hungarian method and re-schedule the remaining tasks on servers if tasking scheduling is not completed in the former process. The function $\delta^{-1}(u)$ ($\delta(k)$) is used to achieve the current server (task) matching with task u (server k) and the variable Num is used to denote the current number of tasks that have been assigned. We define a recursive function **path** to find the augmenting path. After Hungarian method is finished, the remaining tasks are collected into $\underline{\mathcal{U}}$ for re-scheduling. If $\underline{\mathcal{U}} = \emptyset$, we directly return $\mathcal{U}_k$'s as the final result. Otherwise, we choose task $u^\star$ with the least candidate servers in $\mathcal{U}$ and then schedule task $u^\star$ to the server with the least assigned tasks. Such process is repeated until $\underline{\mathcal{U}} = \emptyset$.

B. Resource Allocation

After task scheduling is finished, each local controller k should allocate VM and RF resource for task execution and

Algorithm 2 Task Scheduling Algorithm

Initialize: Set $\delta^{-1}(u) = void, \forall u \in \mathcal{U}$; $\delta(k) = void, \forall k \in \bar{\mathcal{K}}$, $Num = 0$;
Finding a maximum matching based on Hungarian method:
for each $u \in \mathcal{U}$ **do**
  Set $\underline{\mathcal{K}} = \bar{\mathcal{K}}$.
  $Num = Num +$ **path**(u) where **path** is a recursive function defined in the following.
end for
path (u) {
for each $k \in \mathcal{K}_u \bigcap \underline{\mathcal{K}}$ **do**
  Update $\underline{\mathcal{K}} = \underline{\mathcal{K}} \setminus \{k\}$.
  if (k is unsaturated $\|$ **path**$(\delta(k))$) **then**
    $\delta^{-1}(u) = k$, $\delta(k) = u$, return 1.
  end if
end for
Return 0. }
$\mathcal{U}_k = \{u : \delta^{-1}(u) = k, u \in \mathcal{U}\}, \forall k \in \mathcal{K}$, $\underline{\mathcal{U}} = \{u : \delta^{-1}(u) = void, u \in \mathcal{U}\}$.
Re-scheduling remaining tasks:
if $\underline{\mathcal{U}} \neq \emptyset$ **then**
  repeat:
  Choose $u^\star = \arg\min_{u \in \underline{\mathcal{U}}} |\mathcal{K}_u|$ and update $\underline{\mathcal{U}} = \underline{\mathcal{U}} \setminus \{u^\star\}$.
  Calculate $\delta^{-1}(u^\star) = \arg\min_{k \in \mathcal{K}_{u^\star}} |\mathcal{U}_k|$.
  until $\underline{\mathcal{U}} = \emptyset$. Then, update and return $\mathcal{U}_k = \{u : \delta^{-1}(u) = k, u \in \mathcal{U}\}, k \in \mathcal{K}$.
else
  Stop and return $\mathcal{U}_k, k \in \mathcal{K}$.
end if

signal transmission to their respective users in $\mathcal{U}_k$. Similar to problem (4), we reformulate problem (14) as

$$\max_{\mathbf{X}_k, \mathbf{Y}_k} \sum_{u \in \mathcal{U}_k} D_u \tag{15a}$$

$$\text{s.t. (14b) and (14c)}, \tag{15b}$$

whose optimal solution is also difficult to find due to its NP-hardness. In the following, we prove that problem (15) can be reformulated as a submodular function maximization problem with a matroid constraint. According to **Theorem 1**, $D_{\mathcal{U}_k} = \sum_{u \in \mathcal{U}_k} D_u$ is a monotone submodular function. Therefore, we only need to prove that constraints (14b) and (14c) can be replaced with a matroid constraint.

We define $\mathcal{E}_k = \bigcup_{v \in \mathcal{V}_k} \mathcal{E}_{kv}$ and $\mathcal{F}_k = \bigcup_{r \in \mathcal{R}_k} \mathcal{F}_{kr}$ as the VM allocation ground set and the RRH assignment ground set of cluster k, respectively, with $\mathcal{E}_{kv} = \{\tilde{x}_{kv}^{u_k^1}, \tilde{x}_{kv}^{u_k^2}, \ldots, \tilde{x}_{kv}^{u_k^{|\mathcal{U}_k|}}\}$ and $\mathcal{F}_{kr} = \{\tilde{y}_{kr}^{u_k^1}, \tilde{y}_{kr}^{u_k^2}, \ldots, \tilde{y}_{kr}^{u_k^{|\mathcal{U}_k|}}\}$, where $\tilde{x}_{kv}^{u_k^i}$ denotes the action that VM v on server k is allocated to task u_k^i and $\tilde{y}_{kr}^{u_k^i}$ denotes the action that RRH r in cluster k serves user u_k^i. We also define a set family $\mathcal{H}_k = (\mathcal{G}_k, \mathcal{T}_k)$ where $\mathcal{G}_k = \{\mathcal{E}_k, \mathcal{F}_k\}$ is a combinational ground set and $\mathcal{T}_k \subseteq 2^{\mathcal{G}_k}$ is a collection of independent subsets of $\mathcal{G}_k$ given by

$$\mathcal{T}_k = \{\mathcal{X}_k \subseteq \mathcal{G}_k : |\mathcal{X}_k \bigcap \mathcal{E}_{kv}| \leq 1, |\mathcal{X}_k \bigcap \mathcal{F}_{kr}| \leq C_{kr}, |\mathcal{X}_k \bigcap \mathcal{E}_k| \leq V_k^{(MAX)}\}. \tag{16}$$

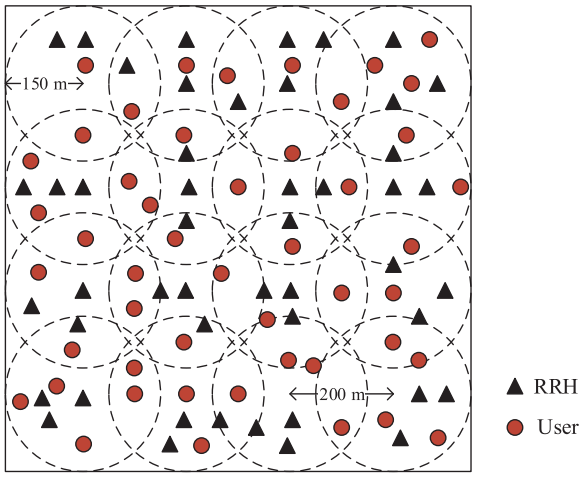


Fig. 3. A random snapshot of the simulation scenario.

We show that $\mathcal{H}_k$ is a matroid in the following lemma.

Lemma 2: For $\forall k \in \mathcal{K}$, the set family $\mathcal{H}_k = (\mathcal{G}_k, \mathcal{T}_k)$ is a matroid.

Proof: See Appendix C for reference. ■

Based on **Lemma 2**, problem (15) can be reformulated as a matroid-constrained monotone submodular maximization problem:

$$\max_{\mathcal{X}_k} \sum_{u \in \mathcal{U}_k} D_u \tag{17a}$$

$$\text{s.t. } \mathcal{X}_k \in \mathcal{T}_k. \tag{17b}$$

Then, problem (17) can be solved with a greedy algorithm for distributed control scheme similar to **Algorithm 1**, as shown in **Algorithm 3**.

Algorithm 3 is a complete algorithm for the distributed scheme including the task scheduling algorithm of the high-level controller and the greedy resource allocation algorithm of the local controllers. After task scheduling is finished at the high-level controller, all the local controllers run the greedy resource allocation algorithm for their respective users independently and parallel. We take local controller k as an example. Local controller k first initializes its $\underline{\mathcal{E}}_{kv}$ and $\underline{\mathcal{F}}_{kr}$ to be empty set and defines $\underline{\mathcal{E}}_k = \bigcup_{v \in \mathcal{V}_k} \underline{\mathcal{E}}_{kv}$, $\underline{\mathcal{F}}_k = \bigcup_{r \in \mathcal{R}_k} \underline{\mathcal{F}}_{kr}$, $\mathcal{X}_k = \{\underline{\mathcal{E}}_k, \underline{\mathcal{F}}_k\}$, and $\mathcal{Y}_k = \mathcal{G}_k$. Then in each iteration, a new element $a^\star$ with the highest marginal gain is added into the set $\mathcal{X}_k$ and removed from the set $\mathcal{Y}_k$. This new element $a^\star$ can be either $\tilde{x}_{kv}^{u_k^i} \in \mathcal{E}_k$ or $\tilde{y}_{kr}^{u_k^i} \in \mathcal{F}_k$. In the case of $a^\star = \tilde{x}_{kv}^{u_k^i} \in \mathcal{E}_k$, the actions that allocate VM v to any task u_k^i should be removed from the candidate action set, i.e., $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{E}_{kv}$. But in the case of $a^\star = \tilde{y}_{kr}^{u_k^i} \in \mathcal{B}$, the actions that assign RRH r to any user u_k^i should be removed from the candidate action set, i.e., $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{F}_{kr}$. Such iteration is repeated until the candidate action set is empty or the marginal gain is zero.

V. Simulation Results

In this section, we carry out Matlab-based simulation experiments to demonstrate the performance of the proposed algorithms. We consider a square area covered by $K = 16$ clusters of RRHs, as shown in Fig. 3. Each cluster has a disc area with a radius of 150m and the distance between

Algorithm 3 Greedy Algorithm for the Distributed Control Scheme

High-level controller's algorithm:
Achieving the task set $\mathcal{U}_k$ of each server $k \in \mathcal{K}$ by executing task scheduling algorithm, i.e., **Algorithm 2**.
Local controllers' algorithm: (Take local controller k as an example.)
Initialize: Set $\underline{\mathcal{E}}_{kv} = \emptyset, v \in \mathcal{V}_k$, $\underline{\mathcal{F}}_{kr} = \emptyset, r \in \mathcal{R}_k$, $\underline{\mathcal{E}}_k = \bigcup_{v\in\mathcal{V}_k} \underline{\mathcal{E}}_{kv}$, $\underline{\mathcal{F}}_k = \bigcup_{r\in\mathcal{R}_k} \underline{\mathcal{F}}_{kr}$, $\mathcal{X}_k = \{\underline{\mathcal{E}}_k, \underline{\mathcal{F}}_k\}$, and $\mathcal{Y}_k = \mathcal{G}_k$.
repeat:
Choose element $a^\star = \arg\max_{a\in\mathcal{Y}} \Delta_{D_{\mathcal{U}_k}}(a|\mathcal{X})$.
if $a^\star = \tilde{x}^u_{kv} \in \mathcal{E}_k$ **then**
  Update $\underline{\mathcal{E}}_{kv} = \underline{\mathcal{E}}_k \cup \{\tilde{x}^u_{kv}\}$, $\underline{\mathcal{E}}_k = \underline{\mathcal{E}}_k \cup \{\tilde{x}^u_{kv}\}$, $\mathcal{X}_k = \mathcal{X}_k \cup \{\tilde{x}^u_{kv}\}$, and $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{E}_{kv}$.
  if $|\underline{\mathcal{E}}_k| = V_k^{(MAX)}$ **then**
    $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{E}_k$.
  end if
end if
if $a^\star = \tilde{y}^u_{kr} \in \mathcal{F}_k$ **then**
  Update $\underline{\mathcal{F}}_{kr} = \underline{\mathcal{F}}_{kr} \cup \{\tilde{y}^u_{kr}\}$, $\underline{\mathcal{F}}_k = \underline{\mathcal{F}}_k \cup \{\tilde{y}^u_{kr}\}$, $\mathcal{X}_k = \mathcal{X}_k \cup \{\tilde{y}^u_{kr}\}$, and $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{F}_{kr}$.
  if $|\underline{\mathcal{F}}_{kr}| = C_{kr}$ **then**
    $\mathcal{Y}_k = \mathcal{Y}_k \setminus \mathcal{F}_{kr}$.
  end if
end if
until $\mathcal{Y}_k = \emptyset$ or $\Delta_{D_{\mathcal{U}_k}}(a|\mathcal{X}) = 0$, then return $\mathcal{X}_k$.

neighboring clusters is 200m. For simplicity, we assume each cluster has the same number of RRHs distributed uniformly and independently in their respective area, i.e., $L_k = 3, \forall k \in \mathcal{K}$. In addition, there are $U = 480$ users uniformly and independently distributed within the whole square area, given these users are static or have low mobility.

It is assumed that the VMs created on the same server have the same execution efficiency when executing the same task, i.e., $\lambda^u_{kv} = \lambda^u_k, \forall v \in \mathcal{V}_k$, and the execution efficiency λ^u_k is uniformly distributed in [0.1,1]. The task load L_u is assumed to be uniformly distributed in $[10^8, 10^9]$. More default parameter values are listed in Table I.

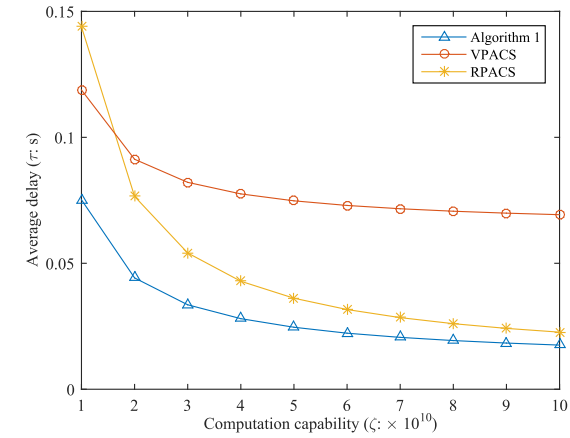
For performance comparison, we introduce two baseline algorithms for the centralized control scheme as follows: VM prioritized allocation algorithm for the centralized control scheme (VPACS) and RRH prioritized assignment algorithm for the centralized control scheme (RPACS). In the VPACS, we first allocate VMs to users' tasks according to greedy algorithms and then choose the nearest RRHs for each user as their serving RRHs in their respective clusters under the constraint of the maximal number of active connection links. But in the RPACS, the serving RRHs for each user are chosen according to greedy algorithms. Then we allocate equal number of VMs to their tasks and the remaining VMs are allocated to the tasks with larger load. Also, a baseline algorithm for the distributed control scheme is adopted, which is called RRH prioritized assignment algorithm for the distributed control scheme (RPADS). In the RPADS, the task scheduling algorithm (**Algorithm 2**) is applied by the high-level controller.

TABLE I
SIMULATION PARAMETERS

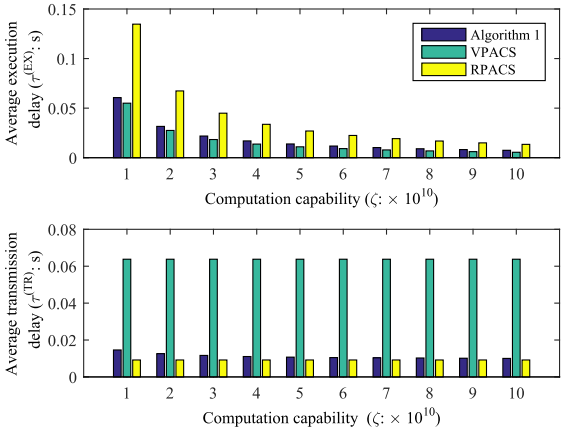
Parameter	Value
Cluster number K	16
User number U	480
Constants $\tau_0^{(C)}$, $\tau_1^{(C)}$	5s, 5s
System bandwidth B	10MHz
Interference constant	10^{-14}
Computing capability ζ	10^{10} cycles/s
Computing capacity N	40
positive weights $\alpha_u, \forall u \in \mathcal{U}$	1
maximum number of active connection links per RRH $C_{kr}, \forall k \in \mathcal{K}, r \in \mathcal{R}_k$	12
RRH number per cluster R	3
Noise power spectral density σ^2	-169 dBm/Hz
Path loss (d in km)	$37.6\log_{10}(d) + 140.7$
Amount of output data $\Phi_u, \forall u \in \mathcal{U}$	10^6
Task load $\rho_u, \forall u \in \mathcal{U}$	$[10^8, 10^9]$
Log-normal shadowing	8 dB
Execution efficiency $\lambda^u_k, \forall u \in \mathcal{U}, \forall k \in \mathcal{K}$	$[0.1, 1]$
Transmit power $P_{kr}, \forall k \in \mathcal{K}, \forall r \in \mathcal{R}_k$	1W
Maximal number of VMs working simultaneously $V^{(MAX)}$	500

We choose the nearest RRHs for each user as their serving RRHs under the constraint of the maximal number of active connection links and then allocate equal number of VMs to their tasks. Also, the remaining VMs are allocated to the tasks with larger load. Note that the three baseline algorithms are based on the separate optimization of VM allocation and RRH assignment.

We first compare the proposed greedy algorithm for the centralized control scheme (**Algorithm 1**) with the VPACS and RPACS. Fig. 4(a) shows the average delay $\tau = \frac{1}{U}\sum_{u\in\mathcal{U}} \tau_u$ with respect to different computation capabilities of VMs and Fig. 4(b) presents two components of τ including the average execution delay $\tau^{(EX)} = \frac{1}{U}\sum_{u\in\mathcal{U}} \tau_u^{(EX)}$ and the average transmission delay $\tau^{(TR)} = \frac{1}{U}\sum_{u\in\mathcal{U}} \tau_u^{(TR)}$ into two sub-figures. In Fig. 4(a), the average delay τ of all three algorithms decreases as the value of computation capability ζ increases. Specifically, the average execution delay $\tau^{(EX)}$ in the upper sub-figure of Fig. 4(b) is reduced with more powerful computation capability but the average transmission delay $\tau^{(TR)}$ in the bottom sub-figure of Fig. 4(b) is almost unchanged. This is because more powerful computation capability can save more computation time. We can also find that the proposed greedy algorithm for the centralized control scheme achieves better performance than the other two algorithms. In the VPACS, the average execution delay $\tau^{(EX)}$ is the smallest but the average transmission delay $\tau^{(TR)}$ is the largest because the VPACS algorithm gives priority to allocating VMs according to greedy algorithms and then the serving RRHs are chosen according to the proximity of RRHs to users. In contrast to the VPACS, the RPACS has the smallest average transmission delay $\tau^{(TR)}$ and the largest average execution delay $\tau^{(EX)}$ because its RRH assignment according to greedy algorithms takes priority over VM allocation. This fact suggests that when the computation capability is large enough, the transmission delay becomes dominant and thus the performance of the RPACS is close to the proposed greedy algorithm for the centralized control scheme.

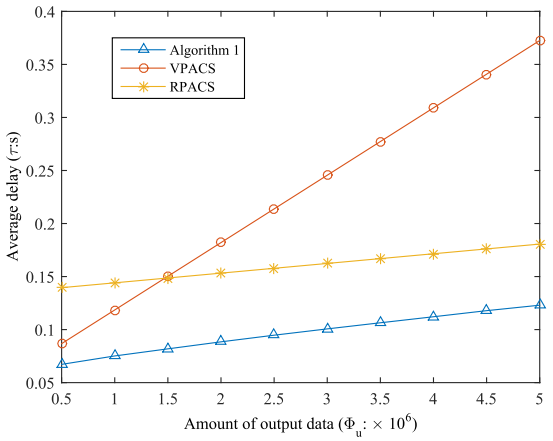


(a)

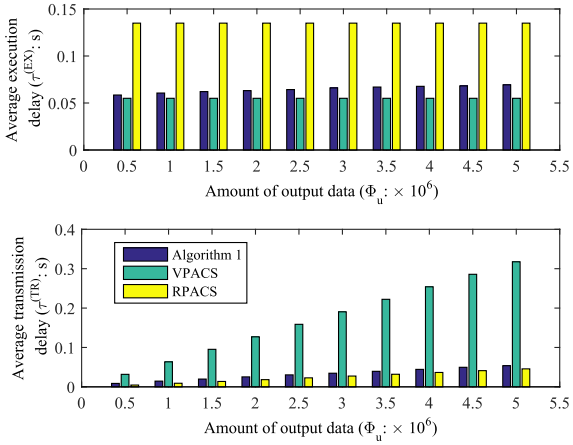


(b)

Fig. 4. Comparison of the proposed greedy algorithm for the centralized control scheme and two baseline ones, the VPACS and the RPACS: (a) is the average delay τ versus different computation capabilities of VMs and (b) presents two parts of τ including the average execution delay $\tau^{(EX)}$ and the average transmission delay $\tau^{(TR)}$.



(a)

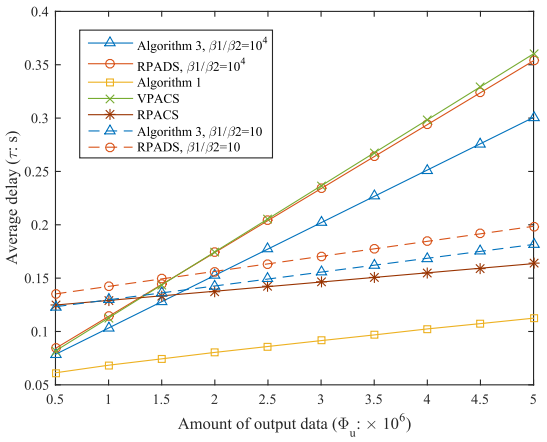


(b)

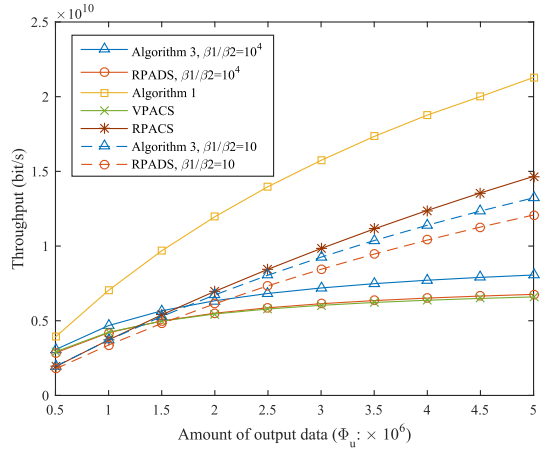
Fig. 5. Comparison of the proposed greedy algorithm for the centralized control scheme and two baseline ones, the VPACS and the RPACS: (a) is the average delay τ versus the amount of output data and (b) presents two parts of τ including the average execution delay $\tau^{(EX)}$ and the average transmission delay $\tau^{(TR)}$.

In Figs. 5(a) and 5(b), we compare the proposed greedy algorithm for the centralized control scheme (**Algorithm 1**) with the VPACS and RPACS over different values of the amount of output data. For simplicity, we assume that each task has the same amount of output data, i.e., $\Phi_u = \Phi, \forall u \in \mathcal{U}$. Similar to Figs. 4(a) and 4(b), Fig. 5(a) shows the trend of the average delay τ with respect to the amount of output data and Fig. 5(b) presents two components of τ including the average execution delay $\tau^{(EX)}$ and the average transmission delay $\tau^{(TR)}$. It is illustrated that the proposed greedy algorithm for the centralized control scheme outperforms the other ones. When the amount of output data Φ is small, suggesting the execution delay is dominant ($\tau^{(EX)} \gg \tau^{(TR)}$), the performance of the VPACS is close to that of the proposed greedy algorithm for the centralized control scheme because the VPACS first allocates VM resource according to greedy algorithms. However, as the amount of output data Φ increases, the transmission delay shown in the bottom subfigure of Fig. 5(b) also increases, as well as the gap between the performances of the proposed greedy algorithm for the centralized control scheme and the VPACS. In contrast to the VPACS, the performance of the RPACS becomes closer to that of the proposed greedy algorithm for the centralized control scheme. Based on Figs. 4 and 5, we can draw a conclusion that when the execution delay is dominant, the performance of the VPACS is close to that of the proposed greedy algorithm for the centralized control scheme. But when the transmission delay is dominant, the performances of the RPACS and the proposed greedy algorithm for the centralized control scheme are close.

Figs. 6(a) and 6(b) compare the proposed greedy algorithms for the centralized and distributed control schemes (**Algorithm 1** and **Algorithm 3**) with the three baseline algorithms in two cases $\frac{\beta_1}{\beta_2} = 10$ and $\frac{\beta_1}{\beta_2} = 10^4$, under the assumption that $V_k^{(MAX)} = 35, \forall k \in \mathcal{K}$. When the amount of output data is small, indicating the execution delay is dominant, the case with higher ratio of β_1 and β_2 (i.e., $\frac{\beta_1}{\beta_2} = 10^4$) shows better delay and throughput performances than the case with lower ratio of β_1 and β_2 (i.e., $\frac{\beta_1}{\beta_2} = 10$) because the former case gives priority to the execution efficiency than the proximity of RRHs to users. However, as the amount of output data becomes heavy, suggesting the transmission delay

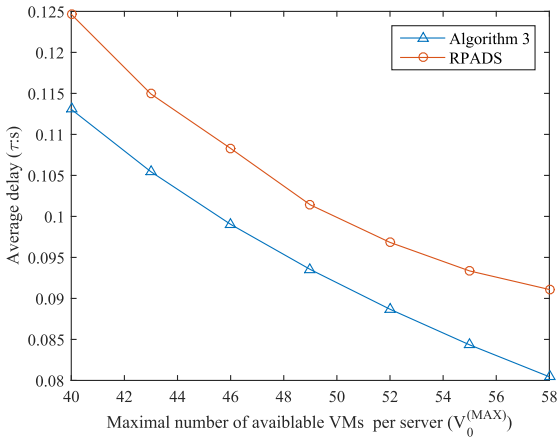


(a)

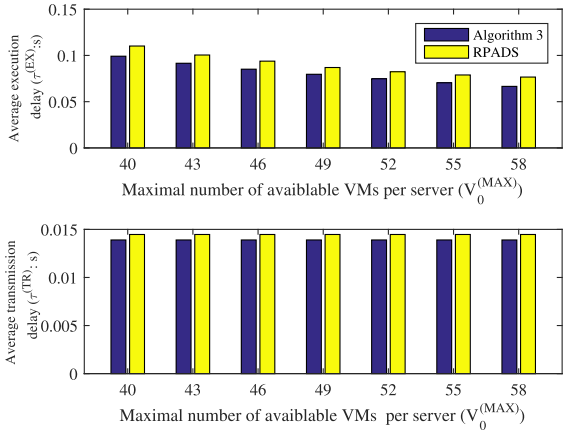


(b)

Fig. 6. Comparison of the proposed greedy algorithms for the centralized and distributed control schemes with the three baseline algorithms under different ratios of β_1 and β_2: (a) average delay and (b) throughput.



(a)



(b)

Fig. 7. Comparison of the proposed greedy algorithm for the distributed control scheme (**Algorithm 3**) and the RPADS: (a) is the average delay τ versus maximal number of available VM per server and (b) presents two parts of τ including the average execution delay $\tau^{(EX)}$ and the average transmission delay $\tau^{(TR)}$.

is dominant, then the case with lower ratio of β_1 and β_2 outperforms than the case with higher ratio. From Figs. 6(a) and 6(b), we find that the proposed greedy algorithm for the centralized control scheme always achieves better delay and throughput performances than other algorithms at the cost of communication overhead and computational complexity. Note that we assume $V^{(MAX)} = 560$ in the proposed greedy algorithm for the centralized control scheme for fairness. Besides, it is also observed that the lower delay suggests the higher throughput.

To further compare the proposed greedy algorithm for the distributed control scheme (**Algorithm 3**) and the RPADS, we assume that $V_k^{(MAX)} = V_0^{(MAX)}, \forall k \in \mathcal{K}$. Fig. 7(a) presents the trend of average delay τ with respect to different values of maximal number of available VMs per server and Fig. 7(b) shows two components including the average execution efficiency $\tau^{(EX)}$ and the average transmission efficiency $\tau^{(TR)}$. Consistent with the results of Fig. 6, the average delay of the proposed greedy algorithm for the distributed control scheme is always smaller than that of the RPADS. The detail of this fact can be found in Fig. 7(b), where the average execution delay $\tau^{(EX)}$ of the RPADS algorithm, as well as the average transmission delay $\tau^{(TR)}$, is larger than that of **Algorithm 3**. This is because in the RPADS the serving RRHs are chosen based on proximity of RRHs to users and VM are allocated to tasks based on the amount of task load.

VI. Conclusion

This work focused on a downlink C-RAN with a hierarchical structure of virtual controllers, where each cluster of RRHs was allocated one local controller and a high-level controller coordinated control plane decisions among local controllers. Furthermore, each local controller was equipped with one server for signal processing. Such hierarchical C-RANs not only separated the control plane from the data plane, but also decoupled baseband processing from RF transmission in the data plane. Therefore, the network performances such as scalability, flexibility, and energy efficiency were improved. However, the computational complexity and communication overhead restricted further enhancement of the network performance. Based on this fact, we considered the tradeoff between network performance and communication overhead as well as

computation complexity, and then proposed centralized and distributed control schemes. This work aimed to minimize the average network delay including transmission delay and processing delay under power constraint and we reformulated this problem as a submodular function maximization by exploiting submodularity. Then two heuristic algorithms with guaranteed approximation were proposed for the centralized and distributed control schemes.

This work is a trial to applying submodularity to solving the joint optimization of task scheduling and resource allocation in a hierarchical C-RAN. Actually, a lot of extension works worth a development. Here we discuss some practical aspects including but not limited to interference, ergodic transmission rate and task scheduling issues.

- Interference issue. In this work, we assumed that the interference χ was fixed by applying intelligent interference management schemes and only considered the variance of the signal power. In a more general case without intelligent interference management, the signal power and interference vary simultaneously when one RRH is transformed from interference source to signal source. We can also prove that the delay reduction function $D = \sum_{u\in\mathcal{U}} D_u$ is also a monotone submodular function in the more general case without intelligent interference management.
- Ergodic transmission rate issue. Due to the existence of small-scale fading, the channel varies fast and a lot of samples are needed to support the calculation of ergodic transmission rate, which causes high computational complexity and much communication overhead. However, when RRHs in C-RANs are equipped with more than one antenna, large-dimensional random matrix theory can be applied to derive the approximation of ergodic transmission rate, which only depends on statistical channel information instead of small-scale fading [36], thus the overhead and complexity can be reduced significantly.
- Task scheduling issue in the distributed control scheme. In this work, we forced the number of tasks scheduled on each server to be as even as possible based on the information about candidate serving clusters (servers). A better scheme should also take the parameters such as task load, execution efficiency of VMs, output data amount, and transmission capacity of each cluster into account and then schedule tasks on different servers to make the weighted load of each cluster as even as possible according to these parameters. Then each cluster can finish their own jobs of signal processing and transmission in a shorter duration.

APPENDIX

A. Proof of *Lemma 1*

According to **Definition 1**, we need to prove that $\mathcal{M}$ satisfies three conditions. The first and second conditions hold trivially and thus we focus on the third one. Assume $\mathcal{I}_1, \mathcal{I}_2 \in \mathcal{I}$ and $|\mathcal{I}_1| < |\mathcal{I}_2|$, there must exist index $n \in \{1,\ldots,N'\}$ satisfying $|\mathcal{I}_1 \cap \mathcal{A}_n| < |\mathcal{I}_2 \cap \mathcal{A}_n|$ or index $l \in \{1,\ldots,L\}$ satisfying $|\mathcal{I}_1 \cap \mathcal{B}_l| < |\mathcal{I}_2 \cap \mathcal{B}_l|$. Without losing generality, we assume that index n' exists with $|\mathcal{I}_1 \cap \mathcal{A}_{n'}| < |\mathcal{I}_2 \cap \mathcal{A}_{n'}|$. Then, we can add an element $a = \tilde{x}^u_{kv} \in \mathcal{A}_{n'} \cap (\mathcal{I}_2 \setminus \mathcal{I}_1)$, which meets the constraints $n' = (k-1)N + n$ and $\mathcal{S}_u((\mathcal{I}_1 \cup \{a\}) \bigcap \mathcal{A}) = \mathcal{W}_u((\mathcal{I}_1 \cup \{a\}) \bigcap \mathcal{B})$, into $\mathcal{I}_1$ and the new set still belongs to $\mathcal{I}$, i.e., $\mathcal{I}_1 \cup \{a\} \in \mathcal{I}$. Therefore, the third condition holds and the set family $\mathcal{M} = (\mathcal{C}, \mathcal{I})$ is a matroid.

B. Proof of *Theorem 1*

The sum of monotone submodular functions is also monotone submodular, so that we only need to prove that D_u is a monotone submodular function for each user $u \in \mathcal{U}$. To this end, we define $\mathcal{X} \subseteq \mathcal{X}' \in \mathcal{I}$ and let $a \in \mathcal{C} \setminus \mathcal{X}'$. Since a can be an action that VM v in server k is allocated to task u with $a = \tilde{x}^u_{kv}$ or an action that RRH r in cluster k serves user u with $a = \tilde{y}^u_{kr}$, we then discuss the submodularity in two cases.

Case I. $a = \tilde{x}^u_{kv} \in \mathcal{A}$. In this case, we have the marginal gain of function D_u when adding a into $\mathcal{X}$, as

$$\begin{aligned}\Delta_{D_u}(a|\mathcal{X}) &= D_u(\mathcal{X}\cup\{a\}) - D_u(\mathcal{X}) \\ &= \frac{L_u}{\zeta}\left(-\frac{1}{\phi^u_{\mathcal{X}} + \lambda^u_{kv}} + \frac{1}{\phi^u_{\mathcal{X}}}\right) > 0 \end{aligned} \tag{18}$$

where $\phi^u_{\mathcal{X}} = \sum_{\tilde{x}^u_{k'v'}\in\mathcal{X}} \tilde{x}^u_{k'v'}\lambda^u_{k'v'}$ is a sum of execution efficiency of the VMs allocated to task u. However, $\Delta_{D_{u'}}(a|\mathcal{X}) = 0, \forall u' \in \mathcal{U}, u' \neq u$. Similarly, we have

$$\begin{aligned}\Delta_{D_u}(a|\mathcal{X}') &= D_u(\mathcal{X}'\cup\{a\}) - D_u(\mathcal{X}') \\ &= \frac{L_u}{\zeta}\left(-\frac{1}{\phi^u_{\mathcal{X}'} + \lambda^u_{kv}} + \frac{1}{\phi^u_{\mathcal{X}'}}\right) > 0 \end{aligned} \tag{19}$$

and $\Delta_{D_{u'}}(a|\mathcal{X}') = 0, \forall u' \in \mathcal{U}, u' \neq u$, where $\phi^u_{\mathcal{X}'} \geq \phi^u_{\mathcal{X}}$. Then, the gap between $\Delta_{D_u}(a|\mathcal{X}')$ and $\Delta_{D_u}(a|\mathcal{X})$ is computed as

$$\begin{aligned}&\Delta_{D_u}(a|\mathcal{X}') - \Delta_{D_u}(a|\mathcal{X}) \\ &\quad= \frac{L_u}{\zeta}\left[\frac{1}{\phi^u_{\mathcal{X}'}(\phi^u_{\mathcal{X}'} + \lambda^u_{kv})} - \frac{1}{\phi^u_{\mathcal{X}}(\phi^u_{\mathcal{X}} + \lambda^u_{kv})}\right] \leq 0. \end{aligned} \tag{20}$$

Case II. $a = \tilde{y}^u_{kr} \in \mathcal{B}$. In this case, the marginal gain of function D_u by adding a into $\mathcal{X}$ comes from the transmission time reduction, which is given by

$$\begin{aligned}\Delta_{D_u}(a|\mathcal{X}) &= D_u(\mathcal{X}\cup\{a\}) - D_u(\mathcal{X}) \\ &= \frac{\Phi_u}{\bar{R}_u(\mathcal{X})} - \frac{\Phi_u}{\bar{R}_u(\mathcal{X}\cup\{a\})} > 0, \end{aligned} \tag{21}$$

where $\bar{R}_u(\mathcal{X}) = B\mathbb{E}_{\mathbf{h}}\left[\log(1 + \sum_{y^u_{kr}\in\mathcal{X}} y^u_{kr}|h^u_{kr}|^2\gamma_{kr})\right]$. However, $\Delta_{D_{u'}}(a|\mathcal{X}) = 0, \forall u' \in \mathcal{U}, u' \neq u$. Similarly, we have

$$\begin{aligned}\Delta_{D_u}(a|\mathcal{X}') &= D_u(\mathcal{X}'\cup\{a\}) - D_u(\mathcal{X}') \\ &= \frac{\Phi_u}{\bar{R}_u(\mathcal{X}')} - \frac{\Phi_u}{\bar{R}_u(\mathcal{X}'\cup\{a\})} > 0, \end{aligned} \tag{22}$$

and $\Delta_{D_{u'}}(a|\mathcal{X}') = 0, \forall u' \in \mathcal{U}, u' \neq u$. Then, the gap between $\Delta_{D_u}(a|\mathcal{X}')$ and $\Delta_{D_u}(a|\mathcal{X})$ is computed as

$$\Delta_{D_u}(a|\mathcal{X}') - \Delta_{D_u}(a|\mathcal{X}) \leq 0, \tag{23}$$

where the inequality holds because of the concavity of logarithm function.

Based on the results in (18)-(23), D_u is verified as a monotone submodular function. Therefore, the sum D of D_u's is also a monotone submodular function.

C. Proof of *Lemma 2*

See the proof of **Lemma 1** as reference. If no user is served in cluster k, then $\mathcal{H}_k = \emptyset$. Otherwise we assume $\mathcal{T}_{k1}, \mathcal{T}_{k2} \in \mathcal{T}_k$ and $|\mathcal{T}_{k1}| < |\mathcal{T}_{k2}|$, there must exist index $v \in \mathcal{V}_k$ satisfying $|\mathcal{T}_{k1} \cap \mathcal{E}_{kv}| < |\mathcal{T}_{k2} \cap \mathcal{E}_{kv}|$ or index $r \in \mathcal{R}_k$ satisfying $|\mathcal{T}_{k1} \cap \mathcal{F}_{kr}| < |\mathcal{T}_{k2} \cap \mathcal{F}_{kr}|$. Without losing generality, we assume that index v' exists with $|\mathcal{T}_{k1} \cap \mathcal{E}_{kv'}| < |\mathcal{T}_{k2} \cap \mathcal{E}_{kn'}|$. Then, we can add an element $a = \tilde{x}^u_{kv} \in \mathcal{E}_{kn'} \cap (\mathcal{T}_{k2} \setminus \mathcal{T}_{k1})$ into $\mathcal{T}_{k1}$ and the new set still belongs to $\mathcal{T}_k$, i.e., $\mathcal{T}_{k1} \cup \{a\} \in \mathcal{T}_k$. Therefore, the third condition holds and the set family $\mathcal{H}_k = (\mathcal{G}_k, \mathcal{T}_k)$ is a matroid.

REFERENCES

[1] W. Xia, T. Q. S. Quek, J. Zhang, S. Jin, and H. Zhu, "Resource allocation by submodular optimization in programmable hierarchical C-RAN," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Beijing, China, Aug. 2018, pp. 558–562.

[2] T. Q. S. Quek, M. O. Simeone, and W. Yu, *Cloud Radio Access Networks: Principles, Technologies Applications.* Cambridge, U.K.: Cambridge Univ. Press, 2017.

[3] X. Chen, Z. Han, Z. Chang, G. Xue, H. Zhang, and M. Bennis, "Adapting downlink power in fronthaul-constrained hierarchical software-defined RANs," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.

[4] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proc. ACM SIGCOMM HotSDN Workshop*, Hong Kong, Aug. 2013, pp. 25–30.

[5] L. Cheng, Y. Gao, J. Fu, X. Zhang, Z. Wei, and D. Yang, "Energy efficient control for software defined cloud radio access network based on small cell," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, Glasgow, U.K., May 2015, pp. 1–5.

[6] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 44–53, Jul. 2013.

[7] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: Potential and challenges," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 150–156, Jan. 2015.

[8] M. Zhou, H. Zhang, S. Zhang, L. Song, Y. Li, and Z. Han, "Design and implementation of device-to-device software-defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[9] R. Yu, G. Xue, M. Bennis, X. Chen, and Z. Han, "HSDRAN: Hierarchical software-defined radio access network for distributed optimization," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8623–8636, Sep. 2018.

[10] C. Pan, H. Zhu, N. J. Gomes, and J. Wang, "Joint precoding and RRH selection for user-centric green MIMO C-RAN," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 2891–2906, May 2017.

[11] V. N. Ha, L. B. Le, and N.-D. Dào, "Energy-efficient coordinated transmission for cloud-RANs: Algorithm design and trade-off," in *Proc. 48th Annu. Conf. Inf. Sci. Syst. (CISS)*, Princeton, NJ, USA, Mar. 2014, pp. 1–6.

[12] Z. Yu, K. Wang, H. Ji, X. Li, and H. Zhang, "Joint user association and downlink beamforming for green cloud-RANs with limited fronthaul," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[13] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1607–1620, Jun. 2017.

[14] O. Dhifallah, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "Joint hybrid backhaul and access links design in cloud-radio access networks," in *Proc. IEEE 82nd Veh. Technol. Conf.*, Boston, MA, USA, Sep. 2015, pp. 1–5.

[15] J. Tang, W. P. Tay, T. Q. S. Quek, and B. Liang, "System cost minimization in cloud RAN with limited fronthaul capacity," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3371–3384, May 2017.

[16] J. Tang, T. Q. S. Quek, C. Tsung-Hui, and S. Byonghyo, "Systematic resource allocation in cloud RAN with caching as a service under two time-scale," *IEEE J. Sel. Areas Commun.*, to be published.

[17] K. Guo, M. Sheng, J. Tang, T. Q. S. Quek, and Z. Qiu, "Exploiting hybrid clustering and computation provisioning for green C-RAN," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4063–4076, Dec. 2016.

[18] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 760–770, Jul. 2018.

[19] T. X. Tran and D. Pompili, "Octopus: A cooperative hierarchical caching strategy for cloud radio access networks," in *Proc. IEEE 13th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Brasília, Brazil, Oct. 2016, pp. 154–162.

[20] Q. Liu, T. Han, and G. Wu, "Computing resource aware energy saving scheme for cloud radio access networks," in *Proc. IEEE Int. Conf. Big Data Cloud Comput. (BDCloud)*, Atlanta, GA, USA, Oct. 2016, pp. 541–547.

[21] M. Dehghan *et al.*, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, May 2015, pp. 936–944.

[22] R. Li, W. Wang, A. Huang, and Z. Zhang, "Content caching at sleeping-enabled base stations in heterogeneous networks," in *Proc. 8th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Yangzhou, China, Oct. 2016, pp. 1–5.

[23] T.-W. Kuo, K. C.-J. Lin, and M.-J. Tsai, "Maximizing submodular set function with connectivity constraint: Theory and application to networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 533–546, Apr. 2015.

[24] H. Xu, T. Zhang, Z. Zeng, and D. Liu, "Joint base station operation and user association in cloud based HCNs with hybrid energy sources," in *Proc. IEEE 26th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Hong Kong, Sep. 2015, pp. 2369–2373.

[25] Y. Yang, L. Chen, W. Dong, and W. Wang, "Active base station set optimization for minimal energy consumption in green cellular networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 11, pp. 5340–5349, Nov. 2015.

[26] K. Son, H. Kim, Y. Yi, and B. Krishnamachari, "Base station operation and user association mechanisms for energy-delay tradeoffs in green cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 8, pp. 1525–1536, Sep. 2011.

[27] K. Son, E. Oh, and B. Krishnamachari, "Energy-aware hierarchical cell configuration: From deployment to operation," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Shanghai, China, Apr. 2011, pp. 289–294.

[28] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[29] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*, Cambridge, U.K.: Cambridge Univ. Press, 2014.

[30] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, Dec. 2011.

[31] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Services (ACM MobiSys)*, San Francisco, CA, USA, Jun. 2010, pp. 49–52.

[32] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 4–10.

[33] L. Yang, J. Cao, S. Tang, T. Li, and A. T. S. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Honolulu, HI, USA, Jun. 2012, pp. 794–802.

[34] J. Liu, B. Bai, J. Zhang, and K. B. Letaief, "Cache placement in Fog-RANs: From centralized to distributed algorithms," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7039–7051, Nov. 2017.

[35] K. Son, R. Guruprasad, S. Nagaraj, M. Sarkar, and S. Dey, "Dynamic cell reconfiguration framework for energy conservation in cellular wireless networks," *J. Commun. Netw.*, vol. 18, no. 4, pp. 567–579, Aug. 2016.

[36] W. Xia, J. Zhang, S. Jin, C.-K. Wen, F. Gao, and H. Zhu, "Large system analysis of resource allocation in heterogeneous networks with wireless backhaul," *IEEE Trans. Commun.*, vol. 65, no. 11, pp. 5040–5053, Nov. 2017.

[37] K. Bernhard and J. Vygen, *Combinatorial optimization: Theory and Algorithms*. New York, NY, USA: Springer, 2008.

[38] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsâĽ"I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, Dec. 1978.
[39] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

Wenchao Xia (S'16) received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications in 2014, where he is currently pursuing the Ph.D. degree. He was involved in cloud radio access networks, massive MIMO communications, edge computing, and large dimensional random matrix theory.

He was a recipient of the Best Paper Award from the 2016 IEEE Global Communications Conference, Washington, DC, USA.

Tony Q. S. Quek (S'98–M'08–SM'12–F'18) received the B.E. and M.E. degrees in electrical and electronics engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1998 and 2000, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008.

He is currently a tenured Associate Professor with the Singapore University of Technology and Design (SUTD). He also serves as the Acting Head of Information Systems Technology and Design Pillar and the Deputy Director of the SUTD-ZJU IDEA. He is a co-author of the books *Small Cell Networks: Deployment, PHY Techniques, and Resource Allocation* (Cambridge University Press, 2013) and *Cloud Radio Access Networks: Principles, Technologies, and Applications* (Cambridge University Press, 2017). His current research interests include wireless communications and networking, the Internet-of-Things, network intelligence, wireless security, and big data processing.

Dr. Quek has been actively involved in organizing and chairing sessions, and has served as a member of the Technical Program Committee and symposium chairs in a number of international conferences. He is currently an elected member of the IEEE Signal Processing Society SPCOM Technical Committee. He was an Executive Editorial Committee Member for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He was a recipient of the 2008 Philip Yeo Prize for Outstanding Achievement in Research, the IEEE GLOBECOM 2010 Best Paper Award, the 2012 IEEE William R. Bennett Prize, the 2015 SUTD Outstanding Education Awards–Excellence in Research, the 2016 IEEE Signal Processing Society Young Author Best Paper Award, the 2017 CTTC Early Achievement Award, and the 2017 IEEE ComSoc AP Outstanding Paper Award, and was the 2016–2018 Clarivate Analytics Highly Cited Researcher. He is a Distinguished Lecturer of the IEEE Communications Society. He was an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and the IEEE WIRELESS COMMUNICATIONS LETTERS.

Jun Zhang (S'10–M'14) received the M.S. degree in statistics from the Department of Mathematics, Southeast University, Nanjing, China, in 2009, and the Ph.D. degree in communications information system from the National Mobile Communications Research Laboratory, Southeast University in 2013. From 2013 to 2015, he was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design, Singapore. Since 2015, he has been with the Faculty of the Jiangsu Key Laboratory of Wireless Communications, College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, where he is currently an Associate Professor. His research interests include massive MIMO communications, physical layer security, edge caching and computing, and large dimensional random matrix theory. He was a recipient of the GLOBCOM Best Paper Award in 2016 and the IEEE APCC Best Paper Award in 2017. He serves as an Associate Editor for the IEEE COMMUNICATIONS LETTERS.

Shi Jin (S'06–M'07) received the B.S. degree in communications engineering from the Guilin University of Electronic Technology, Guilin, China, in 1996, the M.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2003, and the Ph.D. degree in communications and information systems from the Southeast University, Nanjing, in 2007. From 2007 to 2009, he was a Research Fellow with the Adastral Park Research Campus, University College London, London, U.K. He is currently with the Faculty of the National Mobile Communications Research Laboratory, Southeast University. His research interests include space time wireless communications, random matrix theory, and information theory. He serves as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE COMMUNICATIONS LETTERS, and IET COMMUNICATIONS. He and his co-authors have been awarded the 2011 IEEE Communications Society Stephen O. Rice Prize Paper Award in the field of communication theory and the 2010 Young Author Best Paper Award by the IEEE Signal Processing Society.

Hongbo Zhu received the B.S. degree in communications engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1982, and the Ph.D. degree in information and communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1996. He is currently a Professor with the Nanjing University of Posts and Telecommunications. He is also the Head of the Coordination Innovative Center of IoT Technology and Application (Jiangsu), which is the first government authorized Coordination Innovative Center of IoT in China. He also serves as a referee or expert in multiple national organizations and committees. He has authored and co-authored over 200 technical papers published in various journals and conferences. He is currently leading a big group and multiple funds on IoT and wireless communications with current focus on architecture and enabling technologies for Internet of Things. His research interests include mobile communications, wireless communication theory, and electromagnetic compatibility.