Joint Task Offloading and Caching for Massive MIMO-Aided Multi-Tier Computing Networks

Kunlun Wang, *Member, IEEE*, Wen Chen, *Senior Member, IEEE*, Jun Li, *Senior Member, IEEE*, Yang Yang, *Fellow, IEEE*, and Lajos Hanzo, *Fellow, IEEE*

***Abstract*—In this paper, a massive multiple-input multiple-output (MIMO) relay assisted multi-tier computing (MC) system is employed to enhance the task computation. We investigate the joint design of the task scheduling, service caching and power allocation to minimize the total task scheduling delay. To this end, we formulate a robust non-convex optimization problem taking into account the impact of imperfect channel state information (CSI). In particular, multiple task nodes (TNs) offload their computational tasks either to computing and caching nodes (CCN) constituted by nearby massive MIMO-aided relay nodes (MRN) or alternatively to the cloud constituted by nearby fog access nodes (FAN). To address the non-convexity of the optimization problem, an efficient alternating optimization algorithm is developed. First, we solve the non-convex power allocation optimization problem by transforming it into a linear optimization problem for a given task offloading and service caching result. Then, we use the classic Lagrange partial relaxation for relaxing the binary task offloading as well as caching constraints and formulate the dual problem to obtain the task allocation and software caching results. Given both the power allocation, as well as the task offloading and caching result, we propose an iterative optimization algorithm for finding the jointly optimized results. The simulation results demonstrate that the proposed scheme outperforms the benchmark schemes, where the power allocation may be controlled by the asymptotic form of the effective signal-to-interference-plus-noise ratio (SINR).**

***Index Terms*—Multi-tier Computing (MC), massive MIMO, service caching, task scheduling.**

The work of K. Wang was supported by the National Natural Science Foundation of China (NSFC) under grant 61801463. The work of W. Chen was supported by National key project 2020YFB1807700 and 2018YFB1801102, by Shanghai Kewei 20JC1416502, and NSFC 62071296. The work of J. Li was supported by National Natural Science Foundation of China under Grant 61872184. The work of Y. Yang was supported by the National Key Research and Development Program of China under grant 2020YFB2104300, and Shanghai Sailing Program under grant 19YF1455900. L. Hanzo would like to acknowledge the financial support of the Engineering and Physical Sciences Research Council projects EP/P034284/1 and EP/P003990/1 (COALESCE) as well as of the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028). (Corresponding author: Jun Li.)

K. Wang is with the School of Communication and Electronic Engineering, East China Normal University, Shanghai 200241, China. (e-mail: klwang@cee.ecnu.edu.cn). W. Chen is with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. (e-mail: wenchen@sjtu.edu.cn). J. Li is with the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. (email: jun.li@njust.edu.cn). Y. Yang is with Shanghai Institute of Fog Computing Technology (SHIFT), ShanghaiTech University, Shanghai 201210, China, the Research Center for Network Communication, Peng Cheng Laboratory, Shenzhen 518000, China, and Shenzhen SmartCity Technology Development Group Co. Ltd., Shenzhen 518046. (email: yangyang@shanghaitech.edu.cn). L. Hanzo is with the Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U. K. (e-mail: lh@ecs.soton.ac.uk).

Part of this work was presented at IEEE International Conference on Communications, Seoul, South Korea, May 16-20, 2022.

I. Introduction

The intelligent internet of things (IoT) is emerging in support of billions of resource-constrained devices around us connected to heterogeneous networks [1]–[3], including a wide range of new computation-intensive applications such as augmented reality (AR), industrial robots and intelligent robotic cars, etc, [4], [5]. However, these novel applications require agile mobility support, location awareness and low latency. These sophisticated services rely on high-speed real-time signal processing and high-rate data exchange [6]. Due to the substantial overhead of inter-cloud communications, cloud computing is unsuitable for latency-sensitive applications. Hence, a new low-latency platform is needed for meeting these requirements of the intelligent IoT.

Multi-tier computing (MC) constitutes a beneficial virtualized platform for addressing the above concerns [7]–[9], which is capable of providing computing, storage, and networking services. Although the associated task scheduling has been considered in [7], [10], advanced caching may be conceived for alleviating the associated bottleneck of the fronthaul and for reducing the delay, hence increasing the quality-of-service (QoS). With the aid of caching services storing the software of mobile applications at edge nodes, the computation tasks may be executed at the network edge to reduce the latency with the advent of optimized service computation. However, there is a paucity of literature on the joint design of service caching, computational task offloading and radio resource allocation in MC scenarios.

In MC, many users offload their computational tasks to a computation node in the uplink. Hence, the communication performance plays a key role in determining the performance of MC, especially in massive multiple-input multiple-output (MIMO) scenarios [11], [12]. More specifically, the achievable data rates of massive MIMO schemes are predominantly determined by the large-scale fading, which hence also critically influences the resource management. This implies that there is no need for frequently updating the resource management, which mitigates the signalling overhead. Lastly, massive MIMO schemes improve the spectral versus energy efficiency trade-off, and as an explicit benefit, they are capable of supporting for a large number of users, which is particularly important for MC systems. The application of massive MIMO to MC is of pivotal significance since, since the massive MIMO is of salient importance in future wireless communication networks, since apart from a whole host of benefits, it improves computation offloading. Specifically, the

increased spectral and energy efficiencies provided by massive MIMO can yield improved transmission rates and reduced energy consumption for offloading in multi-tier computing, while simultaneously reducing task execution delay. Motivated by these facts, we aim for showing the benefits of massive MIMO schemes in MC networks.

A. Related Work

As a benefit of efficiently harnessing their abundant computing resources, MC systems have been proposed to address the computing and communication resource allocation issues in multi-user task scheduling. However, one of the challenges is the efficient joint computation and wireless communication resource allocation among the nodes to simultaneously support multi-tier computation, since the computational capability and energy resources for task offloading are limited. To address the above challenges and improve the performance of MC systems, sophisticated joint communication and computing resource allocation techniques have been proposed in [13]–[16]. In many practical scenarios, the tasks of the individual users cannot be partitioned and hence they can only be offloaded as a whole [7], [17], [18]. The multi-user offloading strategies proposed in [14], [19] and [7] are based on frequency-division multiple access (FDMA) and code division multiple access (CDMA), respectively. Furthermore, to allow the partitioning of tasks for conceiving partial task offloading schemes, time-division multiple access (TDMA) [10], orthogonal frequency-division multiple access (OFDMA) [20] and non-orthogonal multiple access (NOMA) [21] based multiuser task offloading have also been designed. Explicitly, they can use both local and remote computing.

The above mentioned impressive contributions have been dedicated to studying computation offloading for realizing joint computation and communication resources allocation. However, the benefits of massive MIMO-aided MC schemes have not been explored. Upon deploying very large of number of antennas, the resultant massive MIMO configurations significantly improve the task offloading data rate in contrast to a family of conventional MIMO. Benefiting from the massive MIMO, the integration of MC and massive MIMO arrangements significantly improves the performance of task scheduling and resource management in multi-user MC systems [22]–[26]. Although the benefits of massive MIMO-aided MC have been demonstrated by establishing high-date rate and low-latency links, the joint benefits of service caching and task offloading have not been exploited in resource allocation, which are particularly pivotal for delay-sensitive MC systems.

Given that the edge nodes have limited storage resources, beneficially distributing the tasks among edge and cloud nodes is vital for optimizing the QoS at a high resource efficiency. Caching services at edge nodes relieves the burden of both the backhual transmission and of the central clouds. Hence, service caching has received significant attention in the literature. To elaborate, Borst *et al.* [27] have proposed popularity-based distributed caching algorithms for content distribution networks. Zhang *et al.* [28] have proposed a cooperative edge caching scheme for user-centric large-scale wireless networks to minimize the delay, which optimizes both the content caching and the network's cluster size based on the time-variant network topology, service distribution, channel state, and content popularity. Yang *et al.* [29] have devised caching schemes based on the user location for maximizing the total hit rate of the cached content. In particular, the future content hit rate is estimated by their linear prediction based model. Specifically, dynamic caching at edge nodes has been discussed in [30]–[32]. Bi *et al.* [30] have studied the joint optimization of content placement, computational resource allocation, and communication resource allocation. By considering the realistic unknown arrival orders of service requests, online algorithms have been proposed in [31] for dynamic service caching at the edge nodes for coping with the limited computational capability of edge computing nodes, and for mitigating the task scheduling costs of forwarding requests and downloading new services. Xu *et al.* [32] have studied a joint service caching and computation offloading scheme in dense cellular networks relying on edge computing, and considered an efficient decentralized online service caching algorithm by exploiting both the temporal and spatial service content popularity patterns. However, the above contributions have not taken into account the multi-tier task computation issues of massive MIMO-enabled resource management and task scheduling, which are crucial for MC systems. Furthermore, since massive MIMO systems substantially improve the communication performance at the edge of the network, a main issue is in this context how to jointly manage task offloading and service caching. More precisely, we should decide which tasks have to be computed and cached in the massive MIMO-enabled node stratum, versus the fog node stratum. All in all, both the cache size and computational capability of edge nodes are limited. To guarantee the stringent QoS requirements of resource-intensive and delay-sensitive services, service caching and task offloading should be jointly considered.

B. Contributions and Organization

Against the above backdrop, we aim for addressing the aforementioned challenges by proposing a technique for jointly optimized computing, communication and caching in massive MIMO-enabled MC systems. Nevertheless, offloading incurs extra overhead due to the communication required between the mobile device and the computing server. The additional communication requirement affects both the energy consumption and the latency, and consequently, the offloading power allocation is particularly important, which is the focus of this treatise. We propose a multi-tier computation based system, in which the task nodes (TNs) offload their tasks to the massive MIMO-aided relay node (MRN) in their proximity, which have under-utilized caching resources. Again, caching popular services reduces the task execution latency. By exploiting the intricate relationship between the task execution delay and service caching, as well as the optimal power allocation, the intertwined service caching and computational resource allocation can be jointly optimized. However, due to the non-convex feature of the resultant optimization problem, it is

TABLE I
NOVELTY COMPARISON

	[7]-2016	[14]-2017	[22]-2019	[23]-2019	[26]-2020	[30]-2020	[31]-2018	[32]-2018	Our work
Joint task offloading and power allocation	✓	✓			✓	✓			✓
Service caching						✓	✓	✓	✓
Massive MIMO			✓	✓	✓				✓
Minimizing task execution time	✓								✓
Multi-tier fog computing					✓				✓
Imperfect channel state information					✓				✓

challenging to find the optimal solution. To circumvent this challenge, we solve the joint task offloading, software package caching, and MRN power allocation problem by conceiving a bespoke alternating optimization technique for decoupling the communication, caching and computation optimization. Furthermore, we design an iterative algorithm for joint task offloading and software caching optimization. Our unique contributions are boldly and explicitly contrasted to the state-of-the-art in Table I, and they are further detailed as follows:

1) We develop massive MIMO-aided multi-tier task scheduling and service caching systems, where services are cached in the computing and caching nodes (CCN) constituted by nearby MRN. Based on this scheme, we design a mechanism that jointly performs task offloading, service caching and communication resource allocation for efficient task execution.
2) Due to the NP hard nature of the joint task scheduling and service caching optimization problem, we propose an alternating optimization technique for solving the task offloading, service caching and power allocations puzzle. We transform the original joint optimization problem into two subproblems, namely into the task offloading and service caching subproblems plus a MRN power allocation subproblem.
3) We transform the non-convex power allocation subproblem into a linear optimization problem, and use the classic Lagrange partial relaxation to relax both the task offloading as well as caching constraints and formulate the dual problem for carrying out the joint task offloading and software caching decisions. Based on our power allocation, task offloading and software caching results, we propose an iterative optimization algorithm for finding the jointly optimal solution. Additionally, the proposed iterative algorithm is formally shown to be converge.
4) We evaluate the performance of the proposed massive MIMO-aided multi-tier MC systems through extensive simulations. The simulation results demonstrate that the proposed algorithm substantially outperforms the traditional solutions for a diverse range of system parameters.

The rest of the paper is organized as follows. Section II describes the system model and our problem formulation, including the caching model, the computation model, and the communication model. In Section III, we optimize the MRN power allocation by transforming it into a linear optimization problem. In Section IV, we use the Lagrange partial relaxation technique for relaxing the binary constraints and formulate the dual problem to obtain the task allocation and software caching results. Furthermore, we conceive an popular alternating optimization technique to obtain the joint power allocation, task offloading and software caching result. In Section V, we discuss our simulation results, followed by our conclusions in Section VI. Table II lists the frequently used notations.

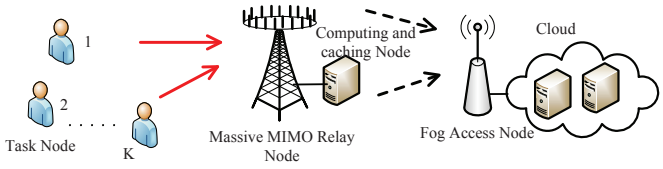

Fig. 1. Illustration of a massive MIMO-enabled multi-tier computing network, where the CCN and the cloud compute the tasks and cache the softwares.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this paper, we design joint service caching and task offloading for MC systems. We consider an MC system, which consists of a decode-and-forward (DF) MRN and FAN equipped with M and N antennas, respectively, plus multiple TNs each equipped with a single antenna. The data transmissions among TN, MRN and FAN in the system model can be regarded as data transmission between a user (TN) and a base station (BS) (FAN) via a relay access point (AP) (MRN). This happens, when the communication is not successful via the direct link, but can be completed with the help of a relay. Thus, we propose to use an MRN regarded as a relay without computing resource for significantly improving the data rate of computational task offloading as well as the task execution efficiency. An example of this scenario is illustrated in Fig. 1. A library of $\mathcal{S} = \{1, \ldots, S\}$ services is provided, such as video streaming, AR, intelligent driving control software, etc, which have different computation and storage requirements, and each service type is requested by one of the TNs of

TABLE II
FREQUENTLY USED NOTATION

Definition	Notation	Definition	Notation
Caching result of the sth software package	$z_{M,s}$	Cache memory size of MRN	C_{MRN}
Data length of type-s software package	l_s	Task offloading decision	$\rho_{k,s}$
Task size	$L_{k,s}$	Number of CPU cycles per bit	C_k
Clock frequency of each core at CCN	f_k^M	Computational delay of the task k_s at CCN	$t_{k,s}^c$
Computational delay of the task k_s at cloud	$t_{k,s}^{c,F}$	Clock frequency of each core in the cloud	f_k^F
task transmission rate from the TN k to the MRN	r_k	Task offloading time from the MRN to the FAN	D_k^R
Task offloading time from the TN k to the MRN	D_k^L	Total transmission energy consumption	E_k^{off}
Task transmission rate from TN k to the MRN	r_k	Power allocation for the TN k at MRN	p_k
Total task scheduling delay for TN k	$D_{\text{total,k}}$	Cost per unit of delay at the kth TN	β_k
Symbol vector received at MRN	$\mathbf{y}_M$	Transmit symbol vector	$\mathbf{s}$
Effective received SINR at MRN	η_k	Effective received SINR at FAN of the kth signal stream	γ_k
Symbol vector received at the FAN	$\mathbf{y}_F$	Task transmission rate from the MRN to the FAN	$\mathcal{R}_k$

Fig. 1.[1] The MRN is equipped with a CCN, which is capable of providing caching and computing services. We assume that the CCN and the MRN of Fig. 1 are co-located, and they are connected using a high-throughput low-latency optical fiber. Hence, the data communication between the MRN and CCN is assumed to be delay-free. The centralized cloud is assumed to have cached all the services in its library, because it has much larger storage and computational capacity than the CCN. We also assume that all services must be executed either by the CCN or by the cloud, given the limited computational resources of the TN. Additionally, we assume having slow channel fading, where the fading envelope of the channels between the TNs and the MRN remains constant during the entire task offloading session (downlink or uplink transmission).

Under the setup in Fig. 1, all the TNs first offload their computational tasks to the MRN. Then, provided that the corresponding services are cached in the CCN, the CCN executes the tasks. Otherwise, the tasks are offloaded to the FAN via the MRN, and then the FAN computes the offloaded tasks. After that, the FAN transmits the task execution results back to TNs. As a rule, the time required by the MRN and FAN for transmitting back the execution results is negligible. The MRN of Fig. 1 is assumed to have all the communication information of the TNs, so that it can coordinate the task offloading for multiple TNs.

A. Service Caching Model

We assume that each service $s \in \mathcal{S}$ corresponds to a particular software package, indexed by $s \in \mathcal{S}$. Specifically, a type-s software package has a data length of l_s bits, $s \in \mathcal{S}$. It should be noted that the software packages frequently requested by the TNs can be proactively cached at the CCN in our cache-aided MC system of Fig. 1.

Based on the popularity distribution, the software packages are indexed in descending order, which means the sth file is the sth most popularly requested software package. The cache size of the MRN is C_{MRN} bits, and the software package library size is larger than C_{MRN}. The software packages requested by the TNs are downloaded from the library based on the Zipf popularity distribution [33]–[35]. With this model, the software packages can be ordered according to their descending order of popularity. We denote $z_{M,s}$ as the caching result of the sth software package by the CCN. For the CCN that is not caching any software package or does not provide sufficient computational resources, the computational tasks should be offloaded to the cloud via the FAN. Similar to the previous assumption, we assume that the cloud and the FAN are co-located and connected using high-throughput low-latency optical fiber. As a result, the task offloading delay imposed by the task transmission between the FAN and the cloud is deemed to be negligible. As the CCN caching capacity is limited, we assume that the storage size allocated for the content of the tasks at the CCN is C_{MRN} bits. Then, we have:

$$\sum_{s\in\mathcal{S}} z_{M,s} l_s \leq C_{\text{MRN}}. \tag{1}$$

B. Computational Model

Note that each service has a specific task, hence we denote k_s as the task of TN k for the requested service s, which is specified by the size of the task input data of $L_{k,s}$ bits. Let $\rho_{k,s}$ denote the task offloading decision of TN k, where

$$\rho_{k,s} \in [0,1], \forall k \in \mathcal{K}, \forall s \in \mathcal{S}. \tag{2}$$

Here $\rho_{k,s} = 1$ represents that task k_s is computed at the cloud and $\rho_{k,s} = 0$ means that task k_s is carried out at CCN. Let $\boldsymbol{\rho} = (\rho_{k,s})_{k\in\mathcal{K},s\in\mathcal{S}}$ denote the computation offloading action.

In our MC system the total number of CPU cycles is deemed to be linearly proportional to the number of bits to be processed [21], [26]. Hence the number of CPU cycles used for completing the task of TN kth is $C_k L_{k,s}$, where the number of CPU cycles per bit is C_k, which depends on by both of the CPU type and on the software task to be executed.

For simplicity, the CCN server is assumed to have multi-core, and each core is independently assigned to a particular offloaded task. Additionally, we assume that each core has the same maximum clock frequency $f_0^{\max}$ (in cycles per second). Then, the computational delay of the task k_s at CCN is given

[1] In particular, to fully unleash the potential of MC for service caching, we assume that the number of computational requests is equal for different services. According to this characteristic, each type of service is associated with one corresponding TN, and indices of service are not differentiated from those of TNs throughout this paper. Note that a more general case on non-equal number computational requests from TNs over different services is left for our future work.

by

$$t_{k,s}^{c} = \frac{C_k(1-\rho_{k,s})L_{k,s}}{f_k^M}, \tag{3}$$

where f_k^M, $k \in \mathcal{K}$ denotes the clock frequency of each core. As a benefiting of dynamic voltage and frequency scaling techniques (DVFS) [36], f_k^M is adjustable. The computational delay of the task k_s at cloud is given by

$$t_{k,s}^{c,F} = \frac{C_k\rho_{k,s}L_{k,s}}{f_k^F}, \tag{4}$$

where f_k^F, $k \in \mathcal{K}$ denotes the clock frequency of each core at cloud. If the required software package is not cached by the CCN, the computational delay of the task k_s at cloud is given by

$$\hat{t}_{k,s}^{c,F} = \frac{C_k L_{k,s}}{f_k^F}. \tag{5}$$

C. Communication Model

In our MC architecture, the task can be offloaded either for execution by the cloud or CCN. If the software is cached by the MRN, the task can be offloaded for execution by the CCN or by the cloud. Then, the task offloading time delay for the kth TN from the MRN to the FAN is given by

$$D_k^R = \frac{\rho_{k,s}L_{k,s}}{\mathcal{R}_k}, \forall k, \tag{6}$$

where $\mathcal{R}_k$ represents the task transmission rate from the MRN to the FAN. If the software is not cached by the MRN, the task is offloaded for execution by the cloud. Thus, the task offloading time delay for the kth TN from the MRN to the FAN is given by

$$\hat{D}_k^R = \frac{L_{k,s}}{\mathcal{R}_k}, \forall k. \tag{7}$$

Correspondingly, the task transmission time from the kth TN to the MRN is given by

$$D_k^L = \frac{L_{k,s}}{r_k}, \forall k, \tag{8}$$

where r_k represents the task transmission rate from the TN k to the MRN.

Then, the total transmission energy consumption of MC for task of the kth TN is given by

$$E_k^{\text{off}} = P_t t_k + p_k D_k^R = \frac{P_t L_{k,s}}{r_k} + \frac{p_k\rho_{k,s}L_{k,s}}{\mathcal{R}_k}, \tag{9}$$

where r_k and $p_k \in \mathbf{p} = [p_1, \cdots, p_K]$ represent the task transmission rate from the kth TN to the MRN and the MRN transmit power allocated to offload the task of the TN k, respectively. Upon denoting P_r as the maximum transmit power of each stream at the MRN. As a result, we have $p_k \leq P_r$.

For ensuring that the overall task offloading delay analysis becomes tractable, we stimulate with the following assumptions.

- Since the task partitioning operation critically hinges both on the data-size and on the computation workload of each task [37] and [38], the CCN distributed in the MRN cannot partition a task until receiving all the data for optimal partitioning.
- In general, task computation may rely both on the specific task structure as well as on the correlation between adjacent task in practical systems, as exemplified by the analysis of media streaming. Recall that for ensuring the reliability of the task execution result, both the CCN and the cloud are unable to start computing a task until the task offloading between the TNs and the MRN or the transmission between the MRN and FAN is completed.

Based on the above assumptions, the FAN allocates its computing resources for each task after receiving all the computational tasks. Accordingly, the total task scheduling delay consists of the task computation delay plus the task transmission delay. By combining (3)-(5) and (6)-(8), the total task scheduling delay of the kth TN is given by

$$D_{\text{total},k} = \sum_{s\in\mathcal{S}} \vartheta_s \Big(D_k^L + \max(t_{k,s}^c, D_k^R + t_{k,s}^{c,F}) z_{M,s} + (\hat{D}_k^R + \hat{t}_{k,s}^{c,F})(1 - z_{M,s}) \Big). \tag{10}$$

D. Problem Formulation

Next, we formulate the weighted-sum delay minimization problem of our massive MIMO-aided task offloading system considered subject to both the task scheduling constraints, the caching and computational capacity constraints at the CCN, and the MRN power allocation constraint.

Explicitly, a positive weighting coefficient $\beta_k \in (0,1)$ is assigned to each TN. By employing this coefficient, the fairness among TNs can be controlled, which satisfies $\sum_{k=1}^{K} \beta_k = 1$. [2] To account for this effect, we adopt the service priority of each TN to determine the value of each weighting factor, which represents the level of importance. Therefore, the resultant optimization problem is formulated as:

$$\begin{aligned}
(\text{P1}) \quad \min_{\mathbf{p},\boldsymbol{\rho},\mathbf{z}} \quad & \sum_{k\in\mathcal{K}} \beta_k D_{\text{total},k}, && \text{(11a)}\\
\text{s.t.} \quad & \rho_{k,s} \in \{0,1\}, \forall k, && \text{(11b)}\\
& 0 \leq p_k \leq P_r, \forall k, && \text{(11c)}\\
& \gamma_k \geq \gamma_0, \forall k, && \text{(11d)}\\
& z_{M,s} \in \{0,1\}, \forall s \in \mathcal{S}, && \text{(11e)}\\
& \sum_{s\in\mathcal{S}} z_{M,s} l_s \leq C_{MRN}, && \text{(11f)}\\
& \sum_{k=1}^{K} f_k^M \leq F_{MRN}, && \text{(11g)}
\end{aligned}$$

where β_k is the (normalized) weighting coefficient representing the cost per unit of delay at the kth TN, so that $\sum_{\mathcal{K}} \beta_k = 1$; (11b) specifies the task offloading indicator function, determining when their task is computed at the CCN or in the cloud; (11c) gives the range of the power allocation

[2]Note that the weighted min-max delay of all TNs is another optimized criteria to be considered in practical systems. In particular, our goal is to minimize the system weighted-sum delay, and the optimal solution is identical to the weighted min-max delay problem by adjusting the weights appropriately based on [39].

variables at the MRN; (11d) quantifies the QoS constraints to ensure that the SINR γ_k of the kth signal stream at the FAN is higher than γ_0; (11e) specifies the software package caching results, constraint (11f) indicates that the amount of cached data must not exceed the capacity C_{MRN} of the CCN's storage, (11g) is imposed to ensure that the sum of computational resources allocated to all offloaded tasks at the CCN must not exceed the computational capability F_{MRN} of the CCN.

Remark 1. *Due to the binary variables of the task offloading vector $\boldsymbol{\rho}$ and software package caching vector $\mathbf{z}$, solving Problem* (P1) *is known to be challenging. Additionally, the feasible set and objective function of Problem* (P1) *are non-convex, so this non-convex problem is NP hard. Therefore, we obtain a locally optimal solution of Problem* (P1) *in this paper, and we rely on transformations as well as simplifications of the original problem. Specifically, to circumvent the non-convex nature of the feasible set of Problem* (P1)*, the binary variables $\boldsymbol{\rho}$ and $\mathbf{z}$ are relaxed into continuous variables. Subsequently, we adopt the popular alternating optimization technique for decoupling the computing, caching and communication designs. Therefore, the resultant optimization problem is transformed to a more tractable form.*

III. Optimal Power Allocation Strategy

In this section, we solve the sub-problem of MRN power allocation. For a given software caching strategy, $\boldsymbol{z} = \boldsymbol{z}^*$, and task offloading strategy $\boldsymbol{\rho}^*$, the power allocation can be performed by the FAP in a centralized way.

A. Received SINR

As for the task computation, both the CCN and the cloud have the options of either executing the task only after receiving all the data or alternatively provided that the task leads itself to partitioning executing the task while still receiving more data. For analytical tractability, let us assume that the CCN and the cloud only carries out the task after receiving all the data from the TN. In this way, the task scheduling in our proposed massive MIMO-aided MC network consists of the TN $\rightarrow$ MRN transmission and the MRN $\rightarrow$ FAN transmission.

For the TN $\rightarrow$ MRN transmission, the symbols of all offloaded tasks are simultaneously transmitted to the MRN. Let $\mathbf{s} = [s_1, \cdots, s_K]^T$ denote the symbol vector with $\mathsf{E}(\mathbf{s}\mathbf{s}^\dagger) = \mathbf{I}_K$, and s_k be the symbol transmitted from the TN k. According to our slow-fading channel model, the symbol vector $\mathbf{y}_M \in \mathbb{C}^{M\times 1}$ received at the MRN is given by

$$\mathbf{y}_M = \mathbf{H}\mathbf{x} + \mathbf{n}_M, \tag{12}$$

where $\mathbf{n}_M \in \mathbb{C}^{M\times 1}$ represents the additive white Gaussian noise (AWGN) at the MRN with zero-mean and a variance of $\mathsf{E}(\mathbf{n}_M\mathbf{n}_M^{\mathrm{H}}) = \sigma_r^2\mathbf{I}_M$, while the transmitted signal is given by

$$\mathbf{x} = \sqrt{P_t}\mathbf{s}. \tag{13}$$

Based on (12), the signal received by the MRN for the kth TN is

$$y_{M,k} = \sqrt{P_t}\mathbf{h}_k s_k + n_{M,k}, \tag{14}$$

where $y_{M,k}$, s_k, and $n_{M,k}$ represent the kth element of $\mathbf{y}_M$, $\mathbf{s}$, and $\mathbf{n}_M$, respectively, and $\mathbf{h}_k$ represents the kth row of $\mathbf{H}$. Hence, the effective SINR of the kth signal stream at the MRN is expressed as

$$\eta_k = \frac{P_t(\mathbf{h}_k\mathbf{h}_k^H)}{\sigma_u^2}. \tag{15}$$

Given (22), the task transmission rate from the TN k to the MRN is given by

$$r_k = B\log_2(1+\eta_k), \tag{16}$$

where B indicates the transmission bandwidth. It is reasonable to assume that the CSI is perfectly known at the receiver [40], since the receiver is capable of acquiring accurate CSI at the receiver (CSIR) with the aid of training, especially, when for example decision-directed joint iterative channel estimation and data detection is employed. However, the transmitter can only acquire imperfect CSI through a finite-rate feedback channel, which introduces both quantization errors and feedback delays. Consequently, we assume that the CSI in the TN $\rightarrow$ MRN phase is perfectly known at the MRN as the up-link receiver, while the CSI in the MRN $\rightarrow$ FAN phase is imperfectly known at the MRN as the down-link transmitter. Thus, the MRN precodes its transmit signal $\mathbf{x}$. Then, the filtered symbol vector $\mathbf{x}_M \in \mathbb{C}^{M\times 1}$ is given by

$$\mathbf{x}_M = \hat{\mathbf{W}}\mathbf{x}, \tag{17}$$

where $\hat{\mathbf{W}} \in \mathbb{C}^{M\times M}$ is the transmit precoding (TPC) matrix of the MRN, which is given by

$$\hat{\mathbf{W}} = \hat{\mathbf{G}}^\dagger\mathbf{P}, \tag{18}$$

where we have $\hat{\mathbf{G}}^\dagger = (\hat{\mathbf{G}}^{\mathrm{H}}\hat{\mathbf{G}})^{-1}\hat{\mathbf{G}}^{\mathrm{H}}$ and $\mathbf{H}^\dagger = (\mathbf{H}^{\mathrm{H}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{H}}$. The diagonal matrix $\mathbf{P} \in \mathbb{R}^{K\times K}$ represents the power allocation results at the MRN. In particular, the kth diagonal element $[\mathbf{P}]_{k,k} = \sqrt{p_k}$ denotes the transmit power allocated for the task of TN k.

As for the MRN $\rightarrow$ FAN phase, the decoded $\mathbf{x}$ is broadcast to the FAN by the MRN. Then, the signal received at the FAN is given by

$$\mathbf{y}_F = \hat{\mathbf{W}}\mathbf{G}\mathbf{P}\mathbf{x} + \hat{\mathbf{W}}\mathbf{n}_F, \tag{19}$$

where $\mathbf{y}_F = [y_{F,1}, \cdots, y_{F,K}] \in \mathbb{C}^{K\times 1}$, and $\mathbf{n}_F$ is the AWGN at the FAN with having zero mean and a variance of $\mathsf{E}(\mathbf{n}_F\mathbf{n}_F^{\mathrm{H}}) = \sigma_F^2\mathbf{I}_K$.

Given (19), we rewrite the symbol vector received at the FAN as

$$\begin{aligned}\mathbf{y}_F =& \hat{\mathbf{G}}^\dagger\mathbf{G}\mathbf{P}\mathbf{x} + \hat{\mathbf{G}}^\dagger\mathbf{n}_F \\ =& \hat{\mathbf{G}}^\dagger(\sqrt{1-\tau_D^2}\hat{\mathbf{G}} + \tau_D\boldsymbol{\Omega}_D)\mathbf{P}\mathbf{x} + \hat{\mathbf{G}}^\dagger\mathbf{n}_F \\ =& \sqrt{1-\tau_D^2}\mathbf{P}\mathbf{x} + \tau_D\hat{\mathbf{G}}^\dagger\boldsymbol{\Omega}_D\mathbf{P}\mathbf{x} + \hat{\mathbf{G}}^\dagger\mathbf{n}_F.\end{aligned} \tag{20}$$

Based on (20), the signal received by the FAN from the kth TN is

$$y_{F,k} = \sqrt{1-\tau_D^2}\sqrt{p_k P_t}s_k + \boldsymbol{\zeta}_k\mathbf{x} + n_k, \tag{21}$$

where $y_{F,k}$, s_k, and n_k represent the kth element of $\mathbf{y}_F$, $\mathbf{s}$, and $\mathbf{n}_F$, respectively, and $\boldsymbol{\zeta}_k$ represents the kth row of $\tau_D\hat{\mathbf{G}}^\dagger\boldsymbol{\Omega}_D\mathbf{P}$.

Hence, the effective SINR of the kth signal stream at the FAN is formulated as:

$$\gamma_k = \frac{(1-\tau_D^2)p_k P_t}{P_t(\boldsymbol{\zeta}_k \boldsymbol{\zeta}_k^H) + \sigma_u^2}. \tag{22}$$

Given (22), the task transmission rate from the MRN to the FAN becomes

$$\mathcal{R}_k = B\log_2(1+\gamma_k). \tag{23}$$

B. Optimal Power Allocation

In order to solve the power allocation optimization problem, we first characterize the asymptotic value of the SINR in (22) when the number of antennas, $M \to \infty$, as formulated from the massive MIMO, which is shown in the following theorem.

Theorem 1. *Upon assuming that the number of antennas at the MRN obeys $M \to \infty$, the received SINR at the FAN in* (22) *has a asymptotical expression, which is given by*

$$\gamma_{k,\infty} = \frac{\lambda_k(1-\tau_D^2)p_k P_t}{\sigma_u^2}. \tag{24}$$

Proof: Please refer to Appendix A. ∎

Given the asymptotic value of the SINR, the power allocation subproblem can be solved as a linear optimization problem. To this end, we can rewrite the power allocation optimization problem as shown in the following proposition.

Proposition 1. *The power allocation that minimizes the task scheduling delay can be found by solving the following linear optimization problem*

$$\min_{\boldsymbol{\varpi}} \quad \sum_{k\in\mathcal{K}} \beta_k \Gamma(\varpi_k) \tag{25a}$$

$$\text{s.t.} \quad (11c), \tag{25b}$$

where we have $\varpi_k = \log_2 p_k$ and

$$\Gamma(\varpi_k) = -\kappa_k \varpi_k - \kappa_k \log_2 \lambda_k(1-\tau_D^2)P_t + 2\kappa_i \log_2 \sigma_u - \upsilon_k. \tag{26}$$

Proof: Please refer to Appendix B. ∎

According to the upper-bound of the task offloading rate, the optimized power allocation results for Problem (25) may be found by Algorithm 1. The power allocation obtained for each iteration tightens the upper-bound in (48), until convergence is achieved. Therefore, the value of the delay decreases after each iteration of the proposed power allocation Algorithm 1.

Proposition 2. *Algorithm 1 converges within a finite number of iterations.*

Proof: Please refer to Appendix C. ∎

IV. Joint Power Allocation, Task Offloading and Software Placement Strategy

In this section, first we will use the Lagrange partial relaxation for relaxing the constraints and formulate the dual problem to obtain the task allocation and software caching results; secondly, the update method conceived for the sub-gradient algorithm are presented; then, the relaxed continuous variables are converted to binary results; finally, we summarize the overall joint optimization algorithm.

Algorithm 1 Power Allocation Algorithm

Step 1: set $i = 0$.
Randomly construct the initial points of the power allocation $\mathbf{p}$, and the maximum number of iterations $I_{\max}$. Calculate $\gamma'_{k,\infty}(0)$ based on $\mathbf{p}$.
Step 2: Set the convergence threshold ε.
while $|\gamma'_{k,\infty}(i) - \gamma'_{k,\infty}(i-1)| \geq \varepsilon$ **do**
  $i = i + 1$;
  Set $\gamma'_{k,\infty}(i) = \gamma_{k,\infty}(i-1)$, and calculate κ_k, υ_k.
  Solve problem (25) for given κ_k and υ_k, and set the results as $\boldsymbol{\varpi}(i)$.
  Update $\mathbf{p}$, where $p_k(i) = 2^{\varpi_k(i)}$.
  Calculate $\gamma_{k,\infty}(i)$ for given $p_k(i)$.
end while
Step 3: Results: $p_k^* = p_k(i)$, $\forall k \in \mathcal{K}$.

A. Task offloading and Software Package Caching

We have relaxed the binary caching constraints into continuous values, and the continuous caching variable can be regarded as the probability of a software package being cached. According to the caching strategy of [41], the sth software package has a probability of $z_{M,s}$ to be independently cached by the CCN. Based on the probabilistic caching scheme of [41], [42], each software package's caching probability has to satisfy $z_{M,s} \in [0,1]$. Given the task offloading rate now, we can obtain the delay of task transmission. However, solving Problem (11) is still NP-hard due to coupled nature of the task offloading variable and the software package caching variable. To tackle this challenge, we introduce the new variable $x_{k,s} = \rho_{k,s} z_{M,s}$ and resort to the following problem transformation:

$$\min_{\boldsymbol{\rho},\mathbf{z},\mathbf{x}} \quad \sum_{k\in\mathcal{K}} \beta_k D_{\text{total},k}, \tag{27a}$$

$$\text{s.t.} \quad (11b), (11e), (11f), \tag{27b}$$

$$x_{k,s} = \rho_{k,s} z_{M,s}. \tag{27c}$$

To solve the transformed problem (27) subject to the non-convex constraint (27c), the equality constraint (27c) should be transformed into an inequality constraint. To proceed, by using McCormick envelopes [43], the McCormick convex relaxation of [43] is applied for equivalently transforming (27c) to a series of inequality constraints, which is given by

$$x_{k,s} \leq z_{M,s}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{28}$$

$$x_{k,s} \leq \rho_{k,s}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{29}$$

$$0 \leq x_{k,s} \leq 1, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}. \tag{30}$$

To deal with the task offloading and software package caching problem, the classic Lagrange partial relaxation can

be used for relaxing the constraints (28) and (29). Then, the respective set of dual Lagrange multipliers are defined as

$$\kappa_{k,s} \geq 0, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{31}$$

$$\upsilon_{k,s} \geq 0, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}. \tag{32}$$

Given the dual Lagrange multipliers, we have the Lagrange function as

$$\begin{aligned} L(\boldsymbol{\kappa}, \boldsymbol{\upsilon}, \boldsymbol{\rho}, \mathbf{z}, \mathbf{x}) = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}} \Big[& \beta_k \left(\frac{L_{k,s}\vartheta_s}{\mathcal{R}_k} + \frac{C_k L_{k,s}\vartheta_s}{f_k^F}\right) x_{k,s} + \\ & \beta_k (\hat{D}_k^R \vartheta_s + \hat{t}_{k,s}^{c,F} \vartheta_s)(1 - z_{M,s}) + \kappa_{k,s}(x_{k,s} - z_{M,s}) \\ & + \upsilon_{k,s}(x_{k,s} - \rho_{k,s}) \Big]. \end{aligned} \tag{33}$$

As a result, we can rewrite the dual problem as

$$\max_{\boldsymbol{\kappa}, \boldsymbol{\upsilon}} \min_{\boldsymbol{\rho}, \mathbf{z}, \mathbf{x}} \quad L(\boldsymbol{\kappa}, \boldsymbol{\upsilon}, \boldsymbol{\rho}, \mathbf{z}, \mathbf{x}), \tag{34a}$$

$$\text{s.t.} \quad (11b), (11e), (11f), (30) - (32). \tag{34b}$$

Thus, we can decompose the dual problem (34) into three sub-problems having independent feasible regions, respectively, which can be expressed as

$$\min_{\boldsymbol{\rho}} \quad \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}} \rho_{k,s}(-\upsilon_{k,s}), \tag{35a}$$

$$\text{s.t.} \quad (11b), \tag{35b}$$

$$\min_{\mathbf{z}} \quad \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}} z_{M,s} \left(-\beta_k (\hat{D}_k^R \vartheta_s + \hat{t}_{k,s}^{c,F} \vartheta_s) - \kappa_{k,s}\right), \tag{36a}$$

$$\text{s.t.} \quad (11e), (11f), \tag{36b}$$

$$\min_{\mathbf{x}} \quad \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}} x_{k,s} \left(\beta_k \left(\frac{L_{k,s}\vartheta_s}{\mathcal{R}_k} + \frac{C_k L_{k,s}\vartheta_s}{f_k^F}\right) + \kappa_{k,s} + \upsilon_{k,s}\right), \tag{37a}$$

$$\text{s.t.} \quad (30). \tag{37b}$$

To elaborate a little further, after we decompose the dual problem, we obtain a set of separate problems from the joint task offloading and software package caching optimization problem. Note that sub-problem (35) is a task offloading problem. The optimal task partitioning strategy is derived in Proposition 3, which is shown as follows:

Proposition 3. *The optimized results* $\{\rho_{k,s}\}$ *of our task scheduling strategy can be expressed as*

$$\rho_{k,s}^* = \frac{\frac{C_k L_{k,s}}{f_k^M}}{\frac{L_{k,s}}{\mathcal{R}_k} + \frac{C_k L_{k,s}}{f_k^F} + \frac{C_k L_{k,s}}{f_k^M}}. \tag{38}$$

Proof: Please refer to Appendix D. ∎

On the other hand, both sub-problems (36) and (37) are linear optimization problems. In this case, generic linear programming methods can be applied for solving them. Then, software package caching problem can be solved on a time scale that considers service package popularity, data size and computing resources required for the service. By solving the sub-problems (35), (36) and (37), we arrive at the optimized values of $\boldsymbol{\rho}$, $\mathbf{z}$ and $\mathbf{x}$. Additionally, the dual variables are updated by the sub-gradient method. In particular, the dual variables in iteration t_1 are updated as follows:

$$\kappa_{k,s}(t_1 + 1) = [\kappa_{k,s}(t_1) + \psi(t_1) d(\kappa_{k,s}(t_1))]^+, \tag{39}$$

$$\upsilon_{k,s}(t_1 + 1) = [\upsilon_{k,s}(t_1) + \psi(t_1) d(\upsilon_{k,s}(t_1))]^+, \tag{40}$$

where we have $[x]^+ = \max\{0, x\}$ and $\psi(t_1)$ is the step size, while $d(\kappa_{k,s}(t_1)$, $d(\upsilon_{k,s}(t_1)$ are the sub-gradients of the dual problems, which are given by

$$d[\kappa_{k,s}(t_1)] = x_{k,s} - z_{M,s}, \tag{41}$$

$$d[\upsilon_{k,s}(t_1)] = x_{k,s} - \rho_{k,s}. \tag{42}$$

According to the proof in [44], the optimal solutions of the transformed problem (27) are guaranteed to be found, which can be regarded as the best task offloading and software caching strategy.

B. Binary Variable Recovery

To efficiently solve problem (11), the binary variables $\boldsymbol{\rho}$ and $\mathbf{z}$ are relaxed into continuous variables and (11) is transformed it into a convex problem. Therefore, the binary variables have to be recovered after the sub-gradient process converges for problem (27). Accordingly, the binary variables $\boldsymbol{\rho}$ and $\mathbf{z}$ are recovered by our specifically constructed Algorithm 2. In particular, an example of $\boldsymbol{\rho}$ is used in Algorithm 2. Meanwhile, the same process is applied to $\mathbf{z}$. Due to the NP-hard nature of the original optimization problem, there is no guarantee that the optimized results acquired actually represent the optimal solution. As a result, there may be a gap between them. However, we will show in our simulations that the gap is modest.

Algorithm 2 Binary Task Scheduling Recovery

1: Calculating the first partial derivations.
   Based on (33), calculate the first partial derivations of Lagrangian $G_{k,s} = \frac{\partial L(\boldsymbol{\kappa}, \boldsymbol{\upsilon}, \boldsymbol{\rho}, \mathbf{z}, \mathbf{x})}{\partial \rho_{k,s}}$ with respect to task scheduling variable $\rho_{k,s}$.
2: Sort all $G_{k,s}$, $\forall k, s$ from largest to smallest. Mark them with G_1, G_2, …, G_g, …. Accordingly, mark the corresponding $\rho_{k,s}$ as ρ_1, ρ_2, …, ρ_g, ….
3: **for** $g = 1, 2, \ldots$ **do**
4: Set $\varsigma_g = 1$ and $\varsigma_{g+1}, \varsigma_{g+2}, \varsigma_{g+3}, \ldots = 0$.
5: **if** Any of the constraints (11b)-(11g) does not hold, **then**
6: Break
7: **end if**
8: **end for**
9: Output the recovered task scheduling solutions $\{\rho_{k,s}\}$, $\forall k, s$.

C. Joint Power, Task Allocation and Software Caching Optimization

With the optimized solution from the above-mentioned two subproblems in place, the joint optimization of power and task allocations, as well as software caching is summarized in Algorithm 3.

As pointed out in the previous subsection, the subproblem of power allocation is solved as a linear optimization problem at a polynomial complexity. There are numerous methods of solving linear problems, including Dantzig's simplex method [43] and the interior-point method. Furthermore, as the subproblem of task allocation and software caching optimization is transformed into a convex one, we may readily find the optimal solution with a polynomial complexity. As a result, the computational complexity of our proposed alternating optimization algorithm is only polynomially increasing, which increases with the problem dimension.

Algorithm 3 Joint Power, Task Allocation and Software Caching Optimization Algorithm

1: Initialize $j = 0$, $\epsilon = 1$, and give feasible points $\mathbf{p}^{(0)}$ and $\boldsymbol{\rho}^{(0)}$.
2: **while** $\epsilon > 0.001$ **do**
3: $j = j + 1$;
4: Solve problem (27), and obtain $\boldsymbol{\rho}^{(j)}$ and $\mathbf{z}^{(j)}$ via Algorithm 2 with $\mathbf{p}^{(j-1)}$.
5: Obtain $\mathbf{p}^{(j)}$ via Algorithm 1 with $\boldsymbol{\rho}^{(j)}$ and $\mathbf{z}^{(j)}$;
6: Calculate $\epsilon = \max_k |\frac{(\sum_{k\in\mathcal{K}} \beta_k D_{\text{total},k})^{(j)} - (\sum_{k\in\mathcal{K}} \beta_k D_{\text{total},k})^{(j-1)}}{(\sum_{k\in\mathcal{K}} \beta_k D_{\text{total},k})^{(j-1)}}|$;
7: **end while**

Proposition 4. *Algorithm 3 converges within a finite number of iterations.*

Proof: Please refer to Appendix E. ■

V. Performance Evaluation

In this section, we quantify the accuracy of the analytical results characterizing our massive MIMO-aided task offloading, software caching and resource allocation regime in the context of MC by means of simulations.

A. Simulation Setup

The corresponding simulation parameters are as follows. There are 10 TNs for task offloading, which are uniformly scattered within a 100 m $\times$ 100 m square area at the left-hand side of Fig. 1. The MRN and FAN are located at positions of $(100, 0)$ and (200,0) respectively, at the right-hand side of Fig. 1. The elements of each channel matrix are distributed as $\mathcal{CN}(0, 1)$, while the transmission bandwidth of the system is $B = 20$ MHz. The variance of the AWGN noise is $\sigma^2 = 10^{-9}$W. As for the CCN, the CPU's computational capability F_{MRN} is uniformly selected from the set of $\{3, 4, \cdots, 10\}$GHz [45].

B. Performance of Software Package Caching

We adopt a benchmark of the simulated optimal solution to verify the superiority of our proposed algorithm, which achieves the best task scheduling performance at the cost of a high computational complexity. In Fig. 2, we investigate the weighted-sum delay versus the cache size of the MRN. As can be observed from Fig. 2 that performance gap between the proposed algorithm and the simulated optimal solution is not significant for $M = 500$ and 1000 antennas, and the gap between the two algorithms is only about $10 \sim 20$ ms. It can also be observed that for $M = 1000$ antennas the weighted-sum delay is substantially lower than for $M = 500$ antennas, which verifies the analytical results at Section III. Furthermore, the weighted-sum delay decreases upon increasing the cache size. This is due to the fact that more software packages are cached at the CCN for larger cache sizes. As a result, the computational task is more likely to be executed locally. However, as the cache size of the CCN increases, the weighted-sum delay reduction rate slows. This is because the computing resources of the CCN are limited.

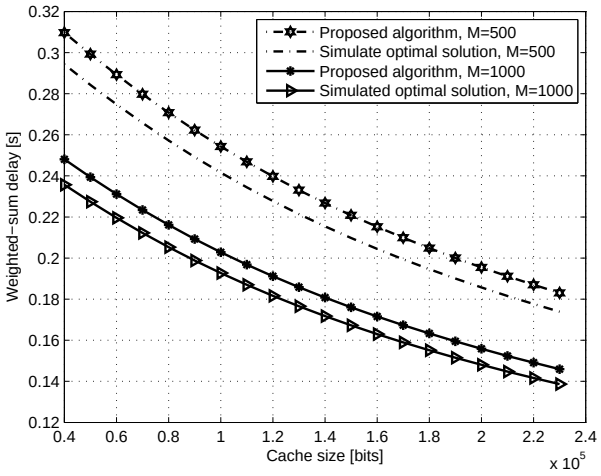

Fig. 2. The weighted-sum delay versus the cache size of the MRN.

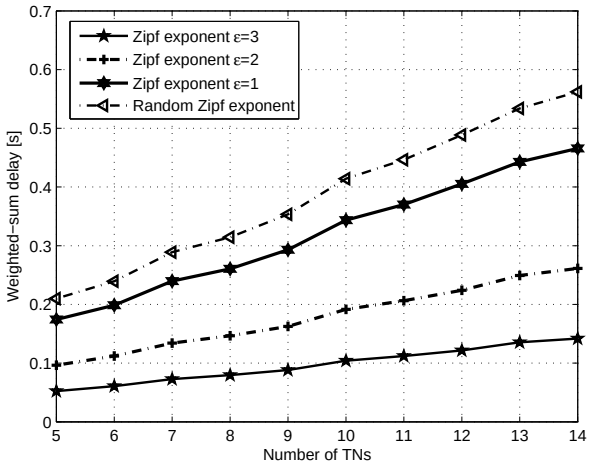

Fig. 3. The effect of the Zipf popularity exponent of ε on the delay.

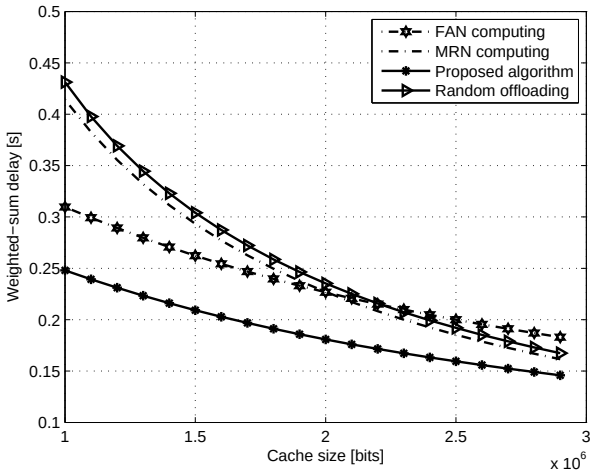

Fig. 4. The weighted-sum delay versus the cache size for different offloading strategies, where the number of TNs is 10.

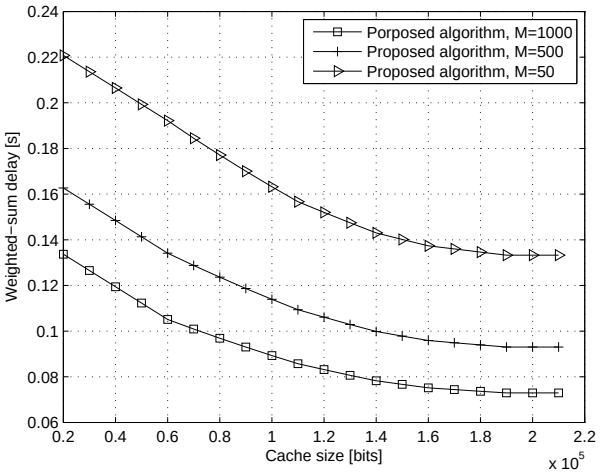

Fig. 6. The weighted-sum delay versus the cache size for different number of MRN antennas, the total number of TNs is 10.

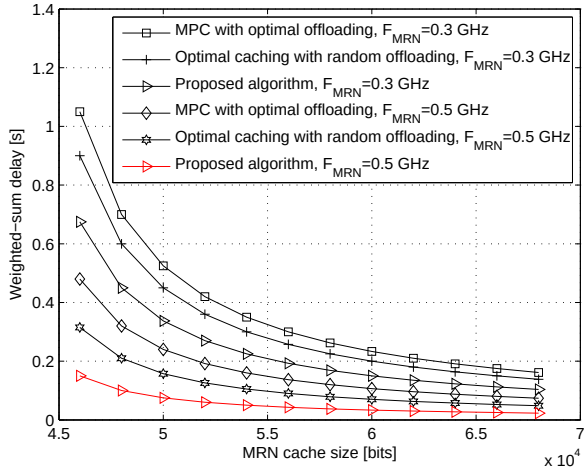

Fig. 5. The weighted-sum delay versus the cache size F_{MRN} of the MRN.

Fig. 3 portrays the weighted-sum delay versus the number of TNs for different Zipf popularity exponents. Observe from the figure that the performance associated with the Zipf distribution is better than that of the uniform random offloading strategy. Furthermore, as the Zipf exponent increases, the weighted-sum delay decreases. This is due to the fact that most of the users access an increasingly smaller number of popular software packages upon increasing the Zipf exponent. Meanwhile, the weighted-sum delay increases near-linearly with the number of TNs, because the number of the computational tasks increases upon increasing the number of TNs.

C. Performance of Joint Task Scheduling and Software Package Caching Algorithm

In Fig. 4, we show the weighted-sum delay versus the cache size for different offloading strategies, and we set the number of TNs to 10. When the cache size of the CCN increases, the weighted-sum delay of the pure CCN computing scheme decreases significantly. Under these circumstances, offloading the tasks to the CCN will achieve lower delay than offloading the tasks to the cloud, which eventually leads to the reduced weighted-sum delay observed. Similarly, the random offloading scheme exhibits a reduced delay upon increasing the cache size. By contrast, when the cache size of the CCN becomes smaller, the pure cloud computing based scheme will have better performance than the pure CCN computing scheme. As expected, the proposed algorithm exhibits the best weighted-sum delay performance among all the schemes. However, similar to Fig. 2, the weighted-sum delay converges to a constant minimum value due to the limited computing resources of the CCN upon increasing the cache size.

We now show the simulation results of the optimal task offloading and optimal software package caching strategies having different system parameters. In Fig. 5, we show the weighted-sum delay versus the cache size F_{MRN} of the CCN, where we compare the most popular content caching (MPC) to our optimal software package caching strategy, and the random task offloading to our optimal task offloading strategy, respectively. Again, we also set the number of TNs to 10. As can be observed from Fig. 5 that the proposed algorithm having different CCN computational resources always performs best among all the schemes. Furthermore, compared to the MPC strategy, we observe that our proposed caching strategy reduces the total cost at the same cache size and CCN computational capacity, which coincides with our analytical results. Additionally, it can be also observed that the weighted-sum delay of the optimal task scheduling scheme is creased slower than that of the random task offloading scheme, when the optimal software package caching strategy is utilized. This is due to the dominance of the task computation delay over the transmission delay. Furthermore, note that upon increasing both the cache size and the computational capability, the delay

is significantly reduced. This phenomenon indicates that if the cache size and the computational resources are increased, then more tasks will be executed locally. This is an explicit benefit of the fact that more popular software packages are cached in the CCN, hence the tasks are more likely to be computed locally with the aid of more computational resources.

For characterizing the impact of the number MRN antennas on the delay, we also conduct weighted-sum delay simulations. In Fig. 6, we show the weighted-sum delay versus the cache size for different number of MRN antennas, where the total number of TNs is 10. Observe that the weighted-sum delay decreases when the number of MRN antennas is increased, which is a benefit of the higher task transmission rate upon having more antennas. On the other hand, as the cache size of the CCN increases, we observe from the figure that the weighted-sum delay is reduced. This is mainly because more software packages are cached at MRN to reduce the task transmission delay. This coincides with the benefits of service caching. However, as the cache size of the CCN increases much larger, the weighted-sum delay converges to a stable value due to the limited computing resources of the CCN.

VI. Conclusions

In this paper, we proposed a massive MIMO-aided MC framework, where the computational tasks of multiple TNs are offloaded to the CCN constituted by the nearby MRN, and to the cloud constituted by the nearby FAN via the MRN. We formulated a weighted-sum delay minimization problem for reducing the total task scheduling delay, while considering realistic imperfect CSI. Since the task offloading and caching variables are binary, we solved the resultant non-convex power allocation, task offloading, and service caching problem by proposing an alternative optimization scheme. We first determined the power allocation for a given task offloading and service caching result, followed by conceiving a beneficial iterative optimization algorithm to obtain the power allocation results. Based on the power allocation, we used the Lagrange partial relaxation for relaxing the binary constraints and for formulating the dual problem to obtain the task allocation and software caching results. Finally, we proposed an iterative optimization algorithm for determining the joint task offloading, service caching and power allocation solution. The simulation results demonstrate that the proposed scheme outperforms the benchmark schemes, and we can choose the minimum-delay optimal offloading strategy of our proposed massive MIMO-aided MC system according to the received SINR found for $M \to \infty$.

Appendix

A. Proof of Theorem 1

Based on (20), the relative strengths of the interference is given by

$$\begin{aligned} & E\left\{\left(\tau_D\hat{\mathbf{G}}^{\dagger}\mathbf{\Omega}_D\mathbf{P}\right)\left(\tau_D\hat{\mathbf{G}}^{\dagger}\mathbf{\Omega}_D\mathbf{P}\right)^H\right\}_{k,k} \\ &= \tau_D^2 E\left\{\hat{\mathbf{G}}^{\dagger}\mathbf{\Omega}_D\mathbf{P}\mathbf{P}^H\mathbf{\Omega}_D^H\hat{\mathbf{G}}^{\dagger H}\right\}_{k,k} \\ &= \tau_D^2\sum_{k=1}^{K} p_k E\left\{\hat{\mathbf{g}}_k^{\dagger}\hat{\mathbf{g}}_k^{\dagger H}\right\}, \\ &= \frac{\tau_D^2(1-\tau_D^2)}{M(1+\tau_D^2)}\sum_{i=1}^{K} p_k. \end{aligned} \tag{43}$$

Next, the eigenvalue/eigenvector decomposition of $\hat{\mathbf{G}}^{\mathrm{H}}\hat{\mathbf{G}}$ is adopted. Then, we have

$$\hat{\mathbf{G}}^{\mathrm{H}}\hat{\mathbf{G}} = \mathbf{Q}^{\mathrm{H}}\mathbf{\Lambda}\mathbf{Q}, \tag{44}$$

where $\mathbf{Q}$ and $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \cdots, \lambda_{\mathrm{K}}\}$ respectively represent the unitary eigenvector matrix and the nonnegative diagonal eigenvalue matrix. As a result, the noise power at the FAN is given by

$$\begin{aligned} & E\left\{\mathbf{n}_U^H\hat{\mathbf{G}}^{\dagger H}\hat{\mathbf{G}}^{\dagger}\mathbf{n}_U\right\} \\ &= \mathrm{tr}\left\{\hat{\mathbf{G}}^{\dagger\mathrm{H}}\hat{\mathbf{G}}^{\dagger}\mathbf{n}_{\mathrm{U}}\mathbf{n}_{\mathrm{U}}^{\mathrm{H}}\right\} \\ &= \mathrm{tr}\left\{\hat{\mathbf{G}}(\hat{\mathbf{G}}^{\mathrm{H}}\hat{\mathbf{G}})^{-1}(\hat{\mathbf{G}}^{\mathrm{H}}\hat{\mathbf{G}})^{-1}\hat{\mathbf{G}}^{\mathrm{H}}\mathbf{n}_{\mathrm{U}}\mathbf{n}_{\mathrm{U}}^{\mathrm{H}}\right\}, \\ &= \mathrm{tr}\left\{\hat{\mathbf{G}}(\mathbf{Q}^{\mathrm{H}}\mathbf{\Lambda}\mathbf{Q})^{-1}(\mathbf{Q}^{\mathrm{H}}\mathbf{\Lambda}\mathbf{Q})^{-1}\hat{\mathbf{G}}^{\mathrm{H}}\mathbf{n}_{\mathrm{U}}\mathbf{n}_{\mathrm{U}}^{\mathrm{H}}\right\}, \\ &= \mathrm{tr}\left\{\hat{\mathbf{G}}\mathbf{Q}^{\mathrm{H}}(\mathbf{\Lambda})^{-1}\mathbf{Q}\mathbf{Q}^{\mathrm{H}}(\mathbf{\Lambda})^{-1}\mathbf{Q}\hat{\mathbf{G}}^{\mathrm{H}}\mathbf{n}_{\mathrm{U}}\mathbf{n}_{\mathrm{U}}^{\mathrm{H}}\right\}, \\ &= \mathrm{tr}\left\{\frac{\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}}{(\mathbf{\Lambda})^2}\mathbf{n}_{\mathrm{U}}\mathbf{n}_{\mathrm{U}}^{\mathrm{H}}\right\}, \\ &= \mathrm{tr}\left\{\frac{\sigma_{\mathrm{u}}^2}{\mathbf{\Lambda}}\mathbf{I}_{\mathrm{K}}\right\}. \end{aligned} \tag{45}$$

Based on (43) and (45), we obtain the effective SINR of the kth data stream at the FAN, which is given by

$$\gamma_k = \frac{(1-\tau_D^2)p_k P_t}{\frac{\tau_D^2(1-\tau_D^2)}{M(1+\tau_D^2)}\sum_{i=1}^{K} p_k + \frac{\sigma_u^2}{\lambda_k}}. \tag{46}$$

B. Proof of Proposition 1

According to the approximation in [46], we can rewrite the upper-bound of the task transmission rate as a linear function of $\log_2\gamma_{k,\infty}$. As a result, for a given variable $\gamma'_{k,\infty}$, we have the following inequality

$$\log_2(1+\gamma_{k,\infty}) \geq \kappa_k \log_2\gamma_{k,\infty} + \upsilon_k, \tag{47}$$

where κ_k and υ_k are respectively defined as $\kappa_k = \frac{\gamma'_{k,\infty}}{1+\gamma'_{k,\infty}}$, $\upsilon_k = \log_2(1+\gamma'_{k,\infty}) - \frac{\gamma'_{k,\infty}}{1+\gamma'_{k,\infty}}\log_2\gamma'_{k,\infty}$. To proceed, it is obvious that $\log_2(1+\gamma_{k,\infty}) = \kappa_k\log_2\gamma_{k,\infty} + \upsilon_k$ is satisfied when $\gamma_{k,\infty} = \gamma'_{k,\infty}$.

By employing the inequality in (47), the upper bound of the objective function in (11) can be obtained, which is given by

$$\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}=\sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left(D_k^L+z_{M,s}^*(\frac{\rho_{k,s}L_{k,s}}{\mathcal{R}_k})+(1-z_{M,s}^*)(\frac{L_{k,s}}{\mathcal{R}_k})\right)\leq\sum_{k\in\mathcal{K}}\beta_k\Theta(p_k),\quad(48)$$

where $\Theta(p_k)$ is defined as

$$\begin{aligned}\Theta(p_k)&=\sum_{s\in\mathcal{S}}\frac{z_{M,s}^*\rho_{k,s}L_{k,s}\vartheta_s+(1-z_{M,s}^*)L_{k,s}\vartheta_s}{\kappa_i\log_2\gamma_{k,\infty}+\upsilon_i}\\&=\sum_{s\in\mathcal{S}}\frac{z_{M,s}^*\rho_{k,s}L_{k,s}\vartheta_s+(1-z_{M,s}^*)L_{k,s}\vartheta_s}{\kappa_i\log_2 p_k+\kappa_i\log_2\lambda_k(1-\tau_D^2)P_t-2\kappa_i\log_2\sigma_u+\upsilon_i}.\end{aligned}\quad(49)$$

Next, we define $\varpi_k=\log_2 p_k$ and denote $\boldsymbol{\varpi}=\{\varpi_1,\ldots,\varpi_{|\mathcal{I}|}\}$. Therefore, the power allocation that minimizes the task scheduling delay can be found by solving the following linear optimization problem

$$\min_{\boldsymbol{\varpi}}\quad\sum_{k\in\mathcal{K}}\beta_k\Gamma(\varpi_k)\quad(50a)$$

$$\text{s.t.}\quad(11c),\quad(50b)$$

where $\Gamma(\varpi_k)$ is given by

$$\Gamma(\varpi_k)=-\kappa_k\varpi_k-\kappa_k\log_2\lambda_k(1-\tau_D^2)P_t+2\kappa_i\log_2\sigma_u-\upsilon_k.\quad(51)$$

C. Proof of Proposition 2

The optimal solution of problem (25) is $\boldsymbol{\varpi}(i)$ after the ith iteration. Set $p_k(i)=2^{\varpi_k(i)}$. Then, we can obtain the following inequalities:

$$\sum_{k\in\mathcal{K}}\beta_k\Gamma(\log_2 p_k(i)),\quad(52a)$$

$$=\sum_{k\in\mathcal{K}}\beta_k\Gamma(\varpi_k(i)),\quad(52b)$$

$$\geq\sum_{k\in\mathcal{K}}\beta_k\Gamma(\varpi_k(i+1)),\quad(52c)$$

$$=\sum_{k\in\mathcal{K}}\beta_k\Gamma(\log_2 p_k(i+1)),\quad(52d)$$

the second inequality (52c) holds because $\varpi_k(i+1)$ is the optimal solutions of problem (25) for the $(i+1)$th iteration; Since the value of $\sum_{k\in\mathcal{K}}\beta_k\Gamma(\varpi_k)$ is lower bounded due to limited energy resources, Algorithm 1 finally converges and outputs the power allocation results.

D. Proof of Proposition 3

For a given software caching strategy and based on Problem (P1), Problem (35) can be transformed into

$$(\text{P2})\quad\min_{\boldsymbol{\rho}}\quad\sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left(\max(t_{k,s}^c,D_k^R+t_{k,s}^{c,F})z_{M,s}^*\right),\quad(53a)$$

$$\text{s.t.}\quad 0\leq\rho_{k,s}\leq 1,\forall k,\quad(53b)$$

By substituting (3), (6) and (4) into the objective function in (P1), the objective function can be rewritten as

$$\sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left(\max(\frac{C_k(1-\rho_{k,s})L_{k,s}}{f_k^M},\frac{\rho_{k,s}L_{k,s}}{\mathcal{R}_k}+\frac{C_k\rho_{k,s}L_{k,s}}{f_k^F})z_{M,s}\right).\quad(54)$$

Then, when $\rho_{k,s}\in[0,1]$, we can readily show that $t_{k,s}^c$ decreases with $\rho_{k,s}$ and $t_{k,s}^c\in[0,\frac{C_kL_{k,s}}{f_k^M}]$. On the other hand, we have $D_k^R+t_{k,s}^{c,F}\in[0,\frac{L_{k,s}}{\mathcal{R}_k}+\frac{C_kL_{k,s}}{f_k^F}]$, which increases with $\rho_{k,s}$. Thus, we can rewrite the objective function of Problem (P2) as (55), which is shown at the top of next page. The optimal task scheduling ratio $\rho_{k,s}^*$ is obtained by solving the following equation

$$\frac{C_k(1-\rho_{k,s})L_{k,s}}{f_k^M}=\frac{\rho_{k,s}L_{k,s}}{\mathcal{R}_k}+\frac{C_k\rho_{k,s}L_{k,s}}{f_k^F}.\quad(56)$$

As a result, the optimal task scheduling ratio $\rho_{k,s}^*$ is given by

$$\rho_{k,s}^*=\frac{\frac{C_kL_{k,s}}{f_k^M}}{\frac{L_{k,s}}{\mathcal{R}_k}+\frac{C_kL_{k,s}}{f_k^F}+\frac{C_kL_{k,s}}{f_k^M}}.\quad(57)$$

E. Proof of Proposition 4

Based on Algorithm 3, we have the inequalities for the jth iteration as follows

$$\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}\left(\mathbf{p}^{(j-1)},\boldsymbol{\rho}^{(j-1)},\mathbf{z}^{(j-1)}\right),\quad(58a)$$

$$\geq\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}\left(\mathbf{p}^{(j-1)},\boldsymbol{\rho}^{(j-1)},\mathbf{z}^{(j)}\right),\quad(58b)$$

$$\geq\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}\left(\mathbf{p}^{(j-1)},\boldsymbol{\rho}^{(j)},\mathbf{z}^{(j)}\right),\quad(58c)$$

$$\geq\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}\left(\mathbf{p}^{(j)},\boldsymbol{\rho}^{(j)},\mathbf{z}^{(j)}\right),\quad(58d)$$

where (58b) is given by solving linear sub-problem (36); (58c) and (58d) are valid according to Proposition 52a and Proposition 1, and solution $\boldsymbol{\rho}^{(j)}$ represents its optimal solution. Note that $\sum_{k\in\mathcal{K}}\beta_k D_{\text{total},k}(\mathbf{p},\boldsymbol{\rho},\mathbf{z})$ is decreased at each iteration based on (58a) and (58d). Additionally, the OF is obviously lower-bounded by a finite value due to the associated constraints. Therefore, within a finite number of iterations, Algorithm 3 converges given a threshold.

REFERENCES

[1] X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: Resource-efficient edge computing for intelligent IoT applications," *IEEE Network*, vol. 22, no.1, pp. 61–65, Feb. 2018.

[2] S. Andreev, V. Petrov, K. Huang, M. A. Lema, and M. Dohler, "Dense moving fog for intelligent IoT: Key challenges and opportunities," *IEEE Commun. Mag.*, vol. 57, no.5, pp. 34–41, May 2019.

[3] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no.3, pp. 1457–1477, Apr. 2017.

[4] K. Yang, Y. Shi, Y. Zhou, Z. Yang, L. Fu, and W. Chen, "Federated machine learning for intelligent IoT via reconfigurable intelligent surface," *IEEE Network*, vol. 34, no.5, pp. 16–22, Sep. 2020.

[5] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no.7, pp. 101–109, July 2017.

$$\sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left(\max(t^c_{k,s},D^R_k+t^{c,F}_{k,s})z^*_{M,s}\right)=$$
$$\begin{cases}\sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left(\frac{C_k(1-\rho_{k,s})L_{k,s}}{f^M_k}z^*_{M,s}\right), & 0\le\rho_{k,s}\le\rho^*_{k,s},\\ \sum_{k\in\mathcal{K}}\beta_k\sum_{s\in\mathcal{S}}\vartheta_s\left((\frac{\rho_{k,s}L_{k,s}}{\mathcal{R}_k}+\frac{C_k\rho_{k,s}L_{k,s}}{f^F_k})z^*_{M,s}\right), & \rho^*_{k,s}\le\rho_{k,s}\le 1.\end{cases} \tag{55}$$

[6] X. Liu and N. Ansari, "Toward green IoT: Energy solutions and key challenges," *IEEE Commun. Mag.*, vol. 57, no.3, pp. 104–110, Mar. 2019.

[7] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Networking*, vol. 24, no.5, pp. 2795–2808, Oct. 2016.

[8] 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.

[9] N. Chen, Y. Yang, T. Zhang, X. Luo, and J. Zao, "FA2ST: Fog as a service technology," *IEEE Commun. Mag.*, pp. 1–1, 2018.

[10] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, pp. 4076–4087, Oct. 2018.

[11] J. G. A. et al., "What will 5G be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no.6, pp. 1065–1082, Jun. 2014.

[12] K. Wang, W. Chen, J. Li, and B. Vucetic, "Green MU-MIMO/SIMO switching for heterogeneous delay-aware services with constellation optimization," *IEEE Trans. Commun.*, vol. 64, no.5, pp. 1984–1995, May 2016.

[13] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Trans. Veh. Technol*, vol. 68, pp. 11399–11403, Nov. 2019.

[14] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no.9, pp. 5994–6009, Sept. 2017.

[15] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no.2, pp. 2061–2073, Apr. 2019.

[16] K. Wang, W. Chen, J. Li, Y. Yang, and L. Hanzo, "Minimum-delay task offloading and caching in multi-tier computing networks," in *Proc. of the IEEE ICC*, (Seoul, South Korea), pp. 1–6, Jun. 2022.

[17] 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, Sept. 2018.

[18] W. Xia, J. Zhang, T. Q. S. Quek, S. Jin, and H. Zhu, "Power minimization-based joint task scheduling and resource allocation in downlink C-RAN," *IEEE Trans. Wireless Commun.*, vol. 17, no.11, pp. 7268–7280, Nov. 2018.

[19] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Trans. Mobile Comput.*, vol. 20, no.3, pp. 1076–1091, Mar. 2021.

[20] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no.3, pp. 1397–1411, Mar. 2017.

[21] K. Wang, Y. Zhou, Z. Liu, Z. Shao, X. Luo, and Y. Yang, "Online task scheduling and resource allocation for intelligent NOMA-based industrial Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 38, no.5, pp. 803–815, May 2020.

[22] O. Y. Bursalioglu, G. Caire, R. K. Mungara, H. C. Papadopoulos, and C. Wang, "Fog massive MIMO: A user-centric seamless hot-spot architecture," *IEEE Trans. Wireless Commun.*, vol. 18, pp. 559–574, Jan. 2019.

[23] H. Pirzadeh, C. Wang, and H. Papadopoulos, "Machine-learning assisted outdoor localization via sector-based fog massive MIMO," in *Proc. IEEE ICC*, (Shanghai, China), pp. 1–6, Jun. 2019.

[24] D. Chen, "Low complexity power control with decentralized fog computing for distributed massive MIMO," in *Proc. IEEE WCNC*, (Barcelona, Spain), pp. 1–6, Apr. 2018.

[25] R. K. Mungara, G. Caire, O. Y. Bursalioglu, C. Wang, and H. C. Papadopoulos, "Fog massive MIMO with on-the-fly pilot contamination control," in *Proc. IEEE ISIT*, (Vail, CO, USA), pp. 1–5, Jun. 2018.

[26] K. Wang, Y. Zhou, J. Li, S. L, W. Chen, and L. Hanzo, "Energy-efficient task offloading in massive MIMO-aided multi-pair fog-computing networks," *IEEE Trans. Commun.*, vol. 69, no.4, pp. 2123–2137, Apr. 2021.

[27] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, pp. 1–9, Apr. 2010.

[28] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, no.8, pp. 1791–1805, Aug. 2018.

[29] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no.4, pp. 915–929, Apr. 2019.

[30] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no.7, pp. 4947–4963, Jul. 2020.

[31] T. Zhao, I.-H. Hou, S. Wang, and K. Chan, "Red/LeD: An asymptotically optimal and scalable online algorithm for service caching at the edge," *IEEE J. Sel. Areas Commun.*, vol. 36, 8, pp. 1857–1870, Aug. 2018.

[32] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM*, pp. 207–215, Apr. 2018.

[33] M. Ji, G. Caire, and A. F. Molisch, "Wirelss device-to-device caching networks: basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no.1, pp. 176–189, Jan. 2016.

[34] S. H. Chae and W. Choi, "Caching placement in stochastic wireless caching helper networks: Channel selection diversity via caching," *IEEE Trans. Wireless Commun.*, vol. 15, no.10, pp. 6626–6637, Oct. 2016.

[35] D. Malak, M. Al-Shalash, and J. G. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networks," *IEEE Trans. Commun.*, vol. 64, no.10, pp. 4365–4380, Oct. 2016.

[36] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no.4, pp. 2322–2358, Aug. 2017.

[37] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "POST: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, pp. 3170–3183, Apr. 2020.

[38] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no.10, pp. 4268–4282, Oct. 2016.

[39] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidisciplinary Optim.*, vol. 26, no.6, pp. 369–395, Apr. 2004.

[40] A. Zappone, P. Cao, and E. A. Jorswieck, "Energy efficiency optimization in relay-assisted MIMO systems with perfect and statistical CSI," *IEEE Trans. Sig. Proc.*, vol. 62, no.2, pp. 443–457, Jan. 2014.

[41] Z. Chen, N. Pappas, and M. Kountouris, "Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal," *IEEE Commun. Lett.*, vol. 21, no.3, pp. 584–587, Mar. 2017.

[42] K. Wang, J. Li, Y. Y, W. Chen, and L. Hanzo, "Content-centric heterogeneous fog networks relying on energy efficiency optimization," *IEEE Trans. Veh. Tech.*, vol. 69, no.11, pp. 13579–13592, Nov. 2020.

[43] L. Liberti and C. Pantelides, "An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms," *J.Global Optim.*, vol. 36, pp. 161–189, Oct. 2006.

[44] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[45] "Hewlett-packard company - enterprise computer server systems and network solutions," 2017, accessed: 2017-04-25. [Online]. Available: https://www.hpe.com/au/en/servers.html.

[46] J. Papandriopoulos and J. S. Evans, "Low-complexity distributed algorithms for spectrum balancing in multi-user DSL networks," in *Proc. of the IEEE ICC*, vol. 7, pp. 3270–3275, Jun. 2006.

Kunlun Wang (M'19) received the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. From 2016 to 2017, he was with Huawei Technologies Company, Ltd., where he was involved in energy efficiency algorithm design. From 2017 to 2019, he was with the Key Lab of Wireless Sensor Network and Communication, SIMIT, Chinese Academy of Sciences, Shanghai, China. From 2019 to 2020, he was with the School of Information Science and Technology, ShanghaiTech University. Since 2021, he has been a Professor with the School of Communication and Electronic Engineering, East China Normal University. His current research interests include energy efficient communications, fog/edge computing networks, resource allocation, and optimization algorithm. He is the Lead Guest Editor for IEEE Journal on Selected Areas in Communications on Multi-Tier Computing for Next Generation Wireless Networks, and the Review Editor for Signal Processing for Communications.

Wen Chen (M'03-SM'11) is a tenured Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University, China, where he is the director of Broadband Access Network Laboratory. He is a fellow of Chinese Institute of Electronics and the distinguished lecturers of IEEE Communications Society and IEEE Vehicular Technology Society. He is the Shanghai Chapter Chair of IEEE Vehicular Technology Society, an Editors of IEEE Transactions on Wireless Communications, IEEE Transactions on Communications, IEEE Access and IEEE Open Journal of Vehicular Technology. His research interests include multiple access, wireless AI and meta-surface communications. He has published more than 110 papers in IEEE journals and more than 120 papers in IEEE Conferences, with citations more than 6000 in google scholar.

Jun Li (M'09-SM'16) received Ph. D degree in Electronic Engineering from Shanghai Jiao Tong University, Shanghai, P. R. China in 2009. From January 2009 to June 2009, he worked in the Department of Research and Innovation, Alcatel Lucent Shanghai Bell as a Research Scientist. From June 2009 to April 2012, he was a Postdoctoral Fellow at the School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia. From April 2012 to June 2015, he was a Research Fellow at the School of Electrical Engineering, the University of Sydney, Australia. From June 2015 to now, he is a Professor at the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. He was a visiting professor at Princeton University from 2018 to 2019. His research interests include network information theory, game theory, distributed intelligence, multiple agent reinforcement learning, and their applications in ultra-dense wireless networks, mobile edge computing, network privacy and security, and industrial Internet of things. He has co-authored more than 200 papers in IEEE journals and conferences, and holds 1 US patents and more than 10 Chinese patents in these areas. He was serving as an editor of IEEE Communication Letters and TPC member for several flagship IEEE conferences. He received Exemplary Reviewer of IEEE Transactions on Communications in 2018, and best paper award from IEEE International Conference on 5G for Future Wireless Networks in 2017.

Yang Yang (S'99-M'02-SM'10-F'18) received the B.S. and M.S. degrees in Radio Engineering from Southeast University, Nanjing, China, in 1996 and 1999, respectively, and the PhD degree in Information Engineering from the Chinese University of Hong Kong in 2002. He is currently a full professor with School of Information Science and Technology, Master of Kedao College, and Director of Shanghai Institute of Fog Computing Technology (SHIFT), ShanghaiTech University, China. He is also an Adjunct Professor with the Research Center for Network Communication, Peng Cheng Laboratory, China, as well as a Senior Consultant for Shenzhen SmartCity Technology Development Group, China. Before joining ShanghaiTech University, he has held faculty positions at the Chinese University of Hong Kong, Brunel University, U.K., University College London (UCL), U.K., and SIMIT, CAS, China.

Yang's research interests include 5G/6G, computing networks, service-oriented collaborative intelligence, IoT applications, and advanced testbeds and experiments. He has published more than 300 papers and filed more than 80 technical patents in these research areas. He has been the Chair of the Steering Committee of Asia-Pacific Conference on Communications (APCC) since January 2019. In addition, he is a General Co-Chair of the IEEE DSP 2018 conference and a TPC Vice-Chair of the IEEE ICC 2019 conference.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk, https://en.wikipedia.orgwikiLajos_Hanzo) (FIEEE'04) received his Master degree and Doctorate in 1976 and 1983, respectively from the Technical University (TU) of Budapest. He was also awarded the Doctor of Sciences (DSc) degree by the University of Southampton (2004) and Honorary Doctorates by the TU of Budapest (2009) and by the University of Edinburgh (2015). He is a Foreign Member of the Hungarian Academy of Sciences and a former Editor-in-Chief of the IEEE Press. He has served several terms as Governor of both IEEE ComSoc and of VTS. He has published 2000+ contributions at IEEE Xplore, 19 Wiley-IEEE Press books and has helped the fast-track career of 123 PhD students. Over 40 of them are Professors at various stages of their careers in academia and many of them are leading scientists in the wireless industry. He is also a Fellow of the Royal Academy of Engineering (FREng), of the IET and of EURASIP. He is the recipient of the 2022 Eric Sumner Field Award.