Decentralized Computation Offoading and Resource Allocation for Mobile-Edge Computing: A Matching Game Approach

Article *in* IEEE Access · November 2018

DOI: 10.1109/ACCESS.2018.2882800

Quoc-Viet Pham
Pusan National University

Nguyen H. Tran
The University of Sydney

Choong Seon Hong
Kyung Hee University

Decentralized Computation Offloading and Resource Allocation for Mobile-Edge Computing: A Matching Game Approach

Quoc-Viet Pham, Tuan LeAnh, Nguyen H. Tran, Bang Ju Park, and Choong Seon Hong

***Abstract*—In this paper, we propose an optimization framework of computation offloading and resource allocation for mobile-edge computing (MEC) with multiple servers. Concretely, we aim to minimize the system-wide computation overhead by jointly optimizing the individual computation decisions, transmit power of the users, and computation resource at the servers. The crux of the problem lies in the combinatorial nature of multi-user offloading decisions, the complexity of the optimization objective, and the existence of inter-cell interference. To overcome these difficulties, we adopt a suboptimal approach by splitting the original problem into two parts: (i) computation offloading decision and (ii) joint resource allocation. To enable distributed computation offloading, two matching algorithms are investigated. Moreover, the transmit power of offloading users is found using a bisection method with approximate inter-cell interference, and the computation resources allocated to offloading users is achieved via the duality approach. Simulation results validate that the proposed framework can significantly improve the percentage of offloading users and reduce the system overhead with respect to the existing schemes. Our results also show that the proposed framework performs close to the centralized heuristic algorithm with a small optimality gap.**

***Index Terms*—Heterogeneous Networks, Matching Theory, Mobile Edge Computing, Resource Allocation.**

I. Introduction

WITH the radically increasing popularity of mobile terminals such as smart phones and tablet computers, a wide-range of mobile applications are constantly emerging, including real-time online gaming, augmented reality, natural language processing, and ultra-high-definition video streaming. These applications have stringent requirements of real-time communication, high energy efficiency, and intensive computation. However, mobile devices are usually constrained with limited battery capacity and computation capability. To tackle these issues, *mobile-edge computing (MEC)* has been developed by the European Telecommunications Standards Institute (ETSI) as a supplement to mobile cloud computing. The idea behind the MEC concept is to move the cloud services and functions to the network edges, thus enabling the executions of computation heavy tasks in close proximity to end users.

The emergence of MEC provides a promising solution for resource-limited users that can migrate a part or all of the intensive computations to powerful MEC servers at the network edges. However, computation offloading may incur additional overhead in terms of energy consumption and latency, e.g., the local execution only suffers from the locally processing delay whereas the remote execution latency includes the transmission delay from the users to the MEC server, the remotely processing delay at the MEC server, and the response delay from the MEC server back to the users. Optimally determining the offloading decisions is an important issue in MEC networks [1], [2]. In the presence of multiple users, the MEC server must be able to simultaneously execute multiple computation tasks and the scarce wireless channel needs to be shared among multiple users. As opposed to the resourceful cloud, the MEC server usually has finite resources. Therefore, the joint optimization of offloading decisions and resource allocation is considered as one of the most important problems in MEC networks to improve the MEC network performance [1]. The last few years have seen the enormous amount of attention for aforementioned problems including offloading decisions in single-user systems [3]–[5] and in multi-user systems [6]–[12], joint optimization of communication and computation resource [13], [14], and joint offloading decision and resource allocation [11], [12], [15], all in MEC networks with a single server, i.e., single-server MEC networks.

The densification of heterogeneous networks (HetNets) provides a promising solution for future networks to increase the spectral-and-energy efficiency and enhance the coverage for indoor and edge users [16], [17]. In HetNets, various kinds of small cells with different characteristics can be deployed by the network operators and/or users, e.g., microcell, femtocell, and picocell. With the concept of network densification and recent advancements in computing hardware, there may exist multiple MEC servers, where each one is connected to and collocated with a smallcell eNB (SeNB). In MEC HetNets, a user can either locally handle the computation or send a request to one among multiple MEC servers for computation offloading. To fully reap the benefits of multi-server MEC HetNets, numerous technical challenges must be addressed. First, beside offloading decisions, one must answer the question "which server does a user offload to?", i.e., *user association*. In general, user association is coupled with *subchannel allocation*, i.e., "which

Quoc-Viet Pham, Tuan LeAnh, and Choong Seon Hong are with Department of Computer Science and Engineering, Kyung Hee University, Republic of Korea. Nguyen H. Tran is with Department of Computer Science and Engineering, Kyung Hee University and School of Computer Science, The University of Sydney. Bang Ju Park is with Department of Electronic Engineering, Gachon University, Gyeonggi-do 13120, South Korea. Dr. Choong Seon Hong is the corresponding author. E-mail: vietpq90@gmail.com, nguyen.tran@sydney.edu.au, sooyong1320@naver.com, latuan, cshong@khu.ac.kr.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2017R1A2A2A05000995).

subchannel does a user utilize to offload the task", and *power allocation*, which greatly complicates the joint optimization of computation offloading and resource allocation in *multi-server MEC HetNets*. Second, since simultaneous offloadings from multiple users can result in severe co-channel inter-cell interference, the offloading rates of users, which offload to different MEC servers but on the same subchannel, are coupled and/or small, thus making the resource allocation problem very hard to solve and requiring efficient schemes for interference management. In contrast, one common setting from existing studies in single-MEC networks based on orthogonal frequency-division multiple access (OFDMA) is that both intra-cell and inter-cell interference are ignored.

Unlike conventional multi-cell networks, where MEC servers are typically deployed at fixed-location macrocell eNBs (MeNBs), MEC servers are collocated at randomly-deployed SeNBs in HetNets without network predimensioning and planning. This strongly affects the user association since an MEC server is able to serve multiple users with different channel qualities and a user is under the service coverage of multiple MEC servers with different computation capabilities. Due to the dynamics and unplanned deployment of HetNets and the possibility of missing a central entity, the centralized optimization of both computation offloading and resource allocation is not always efficient in the distributed networks, for example, wireless sensor networks and Internet of Things (IoT). Although the centralized approach can achieve the optimal solution and high performance for the resource optimization problem and can fulfill various quality-of-service (QoS) requirements of offloading users, all users need to report their channel qualities to the centralized entity, e.g., MEC server, and the centralized server is lonely responsible for accomplishing the optimization for the entire network [18]. Therefore, the centralized optimization has the weakness of high computational complexity and huge reporting overhead. Moreover, there may not exist a dedicated backhaul for information exchange between the MEC servers and even if there is, the wireless backhaul in 5G HetNets would be congested due to the high burden of huge data sharing [19]. All of these points call for efficient schemes of distributed computation offloading multi-server MEC HetNets. Most of existing works in MEC focus on solutions for single-server MEC homogeneous networks [10], [14], [15], single-server MEC HetNets [12], [13], [20], and different aspects in multiple-server MEC networks, for example, computation offloading [21], [22], computation offloading and server provisioning [23], [24], and centralized task offloading and resource allocation [25].

More recently, some distributed schemes for computation offloading in single-server MEC networks were proposed using game theory [10], [26] and game-theoretic machine learning [27]. Nevertheless, none of the existing works discussed the joint optimization of distributed offloading decisions and resource allocation for multi-server MEC HetNets. Further, it is worth noticing that game-theoretic approaches present some shortcomings [28]. For instance, finding the best response of a user usually requires the knowledge of other players' actions, thus limiting the distributed implementations. Next, analyzing the tractability of equilibrium in the game-theoretic methods requires the specific structure in the objective function, which is however not always feasible in some practical scenarios. Matching theory, a subfield of economics for designing low-complexity and distributed algorithms in wireless networks [29], [30], is a promising solution to overcome some limitations of game-theoretic approaches and centralized optimizations. There are several benefits to apply matching theory for resource allocation in wireless networks, e.g., the ability to characterize interactions between heterogeneous players and define the general preferences and the flexibility to reflect different optimization objectives [28], [31].

In this paper, we develop an efficient framework of *distributed computation offloading* and resource (*computation* and *communication*) allocation to minimize the system-wide computation overhead in multi-server MEC HetNets. Here, computation offloading pertains to finding the solution for the user association and subchannel assignment and resource allocation relates to the transmit power allocation of offloading users and computation resource allocation at the MEC servers. The considered problem faces difficulties caused by the combinatorial nature of multi-user offloading decisions, the complexity of the optimization objective, and the existence of inter-cell interference among offloading users, i.e., an NP-hard combinatorial problem. In a nutshell, the main contributions and features of our work are summarized as follows:

- In terms of the system model and problem formulation, we consider a network scenario with multiple SeNBs collocated with the corresponding MEC servers and multiple users and define the objective function as the system-wide computation overhead. Then, an optimization problem is formulated subject to constraints on the MEC server and subchannel selections, maximum transmit power of mobile devices, and maximum computation resources at the MEC servers. To solve the problem, we adopt a suboptimal approach by splitting the original problem into two parts: (i) computation offloading decision, which includes user association and subchannel assignment, and (ii) joint resource allocation, which is further decomposed into the transmit power of offloading users and computation resource allocation at the MEC servers.
- In terms of the mathematical framework, matching theory is adopted to provide the distributed computation offloading decision. Concretely, we employ two matching games to design algorithms for user association and subchannel assignment. Accordingly, a decentralized approach is investigated to determine the offloading decision. With the proposed computation offloading scheme, (i) users decide to offload their computation tasks only if they benefit from computation offloading and remote execution and (ii) users and servers make the offloading decision in a distributed and autonomous fashion. Moreover, we approximate the inter-cell interference and find the transmit power of offloading users using a bisection method, and then solve the computation resource allocation problem via the KKT optimality conditions. Detailed analyses of the matching stability, convergence, and algorithmic complexity of the proposed algorithm are also provided.

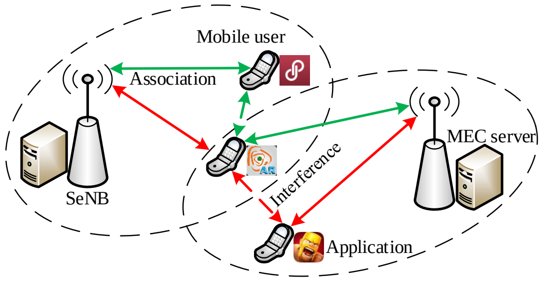


Fig. 1: An MEC system with two SeNBs, each with an MEC server, and three users, each with a distinct application.

- In terms of the performance evaluation, we validate the performance of the proposed algorithm through extensive numerical experiments. Furthermore, we compare our proposed algorithm with four existing solutions: local computing only, offloading only, the heuristic offloading decision algorithm (HODA) proposed in [15], and the heuristic joint task offloading and resource allocation (hJTORA) proposed in [25]. The simulation results illustrate that our proposed algorithm can achieve significant performance improvement in terms of the number of offloading users and the system-wide computation overhead and can perform close to the centralized heuristic algorithm with a small optimality gap.

Our paper is organized as follows. The system model and optimization problem are explained in Section II. In Section III, we propose to decompose the underlying problem into two subproblems and obtain the computation offloading decisions using the matching games. Simulation results are presented and discussed in Section IV. Finally, conclusions and future works are drawn in Section V.

II. System Model and Problem Formulation

A. Network Model

We consider a multi-user multi-server MEC HetNet as illustrated in Fig. 1, where each MEC server is collocated with an SeNB[1]. In general, MEC servers deployed by the network operator have a moderate computing capability and have wireless channel connections to users through the corresponding SeNBs. Small cells operate in an overlaid manner, i.e., each small cell is able to reuse the whole spectrum of the macro cell and interference among small cells exists. To enable tractable analysis and obtain useful insights, we employ a quasi-static network scenario where users remain unchanged during the computation offloading period while they change across different periods. The general scenario, where users leave or join dynamically during the offloading period, is not our focus in this paper and can be considered as future work.

Denote by $\mathcal{M} = \{1, 2, ..., M\}$ the set of SeNBs and by m the index of the mth SeNB. The set of users is denoted by $\mathcal{N} = \{1, 2, ..., N\}$ and n is referred to as the nth user. In this paper, we use OFDMA as the multiple access scheme in the uplink. Assume that there are S subchannels in a small cell, then the set of subchannels is denoted by $\mathcal{S} = \{1, 2, ..., S\}$ and s is referred to as the sth subchannel. Each user is assigned to at most one subchannel and a subchannel is assigned to at most one user. The general case that a user is assigned to multiple subchannels and a subchannel is assigned to multiple users will be considered in future work.

B. Communication Model

For every small cell, an MEC server is collocated with the corresponding SeNB; therefore, a user can offload its computation task to the MEC server via the SeNB. We define the offloading decision profile as $\boldsymbol{A} = \{a_{nm}^s | n \in \mathcal{N}, m \in \mathcal{M}, s \in \mathcal{S}\}$. Specifically, $a_{nm}^s = 1$ if the user n offloads its computation task to the MEC server m on the subchannel s, and $a_{nm}^s = 0$ otherwise. Since each computation task can be either computed locally or remotely, i.e., binary offloading, we have the following constraint:

$$\sum\nolimits_{m \in \mathcal{M}} \sum\nolimits_{s \in \mathcal{S}} a_{nm}^s \leq 1, \forall n \in \mathcal{N}. \tag{1}$$

Each SeNB assigns a subchannel to at most one user, so the following constraint must be satisfied:

$$\sum\nolimits_{n \in \mathcal{N}} a_{nm}^s \leq 1, \forall m \in \mathcal{M}, s \in \mathcal{S}. \tag{2}$$

Moreover, since each server is collocated with a SeNB, which is usually restricted by hardware limitations [32], the number of users offloading to a server should be constrained by

$$\sum\nolimits_{n \in \mathcal{N}} \sum\nolimits_{s \in \mathcal{S}} a_{nm}^s \leq q_m, \forall m \in \mathcal{M}, \tag{3}$$

where q_m is called a *quota*, which represents the maximum number of users the SeNB m can serve. In other words, each server can accommodate at most q_m users to share its CPU.

Given the offloading decision profile $\boldsymbol{A}$, the transmission rate from the user n to the server m over the subchannel s is defined as $R_{nm}^s(\boldsymbol{A}^s, \boldsymbol{P}) = B_s \log_2(1 + \Gamma_{nm}^s(\boldsymbol{A}^s, \boldsymbol{P}))$, where B_s is the bandwidth of the subchannel s, $\boldsymbol{A}^s = \{a_{nm}^s | n \in \mathcal{N}, m \in \mathcal{M}\}$ is the offloading decision profile on the subchannel s, and Γ_{nm}^s is the received signal-to-interference-plus-noise ratio (SINR) from the user n at the server m using the subchannel s, which can be written as

$$\Gamma_{nm}^s(\boldsymbol{A}^s, \boldsymbol{P}) = \frac{p_n^s h_{nm}^s}{n_0 + \sum_{j \neq m} \sum_{k \in \mathcal{N}_j} a_{km}^s p_k^s h_{km}^s}. \tag{4}$$

Here, n_0 is the noise power which is identical for all users, $\mathcal{N}_j$ is the set of users that offload their computation tasks to the MEC server j, $\boldsymbol{P} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n, ..., \boldsymbol{p}_N\}$ is the transmit power vector of all users, $\boldsymbol{p}_n = \{p_n^1, ..., p_n^S\}$ is the transmit power vector of the user n with p_n^s being the transmit power (in Watts) of the user n on subchannel s, and h_{nm}^s is the channel gain from the user n to the SeNB m on the subchannel s. The second term of the denominator in (4) is the inter-cell interference from users offloading to other servers using the same subchannel s. Correspondingly, the data rate from the user n to the server m is $R_{nm}(\boldsymbol{A}, \boldsymbol{P}) = \sum_{s \in \mathcal{S}} a_{nm}^s R_{nm}^s(\boldsymbol{A}^s, \boldsymbol{P})$.

[1] From that point, we use "MEC server" when referring to the related concepts of computation resource and computation offloading, while using "SeNB" when mentioning interference, radio resource, and user association.

C. Computation Model of Mobile Devices

Each user n has a computation task $I_n = \{\alpha_n, \beta_n\}$ [1], [10], where α_n is the computation input data size (in bits) and β_n is the number of CPU cycles required to complete the task I_n (i.e., computation workload/intensity). Binary offloading is considered in our work. The extension to consider partial offloading is highly promising.

We denote f_n^l as the computation capability (in CPU cycles per second) of the user n, where the superscript l stands for *local*. Due to heterogeneous mobile devices, users can have different computation capabilities. Let t_n^l be the completion time of the task I_n by the user n, which can be computed as $t_n^l = \frac{\beta_n}{f_n^l}$. To compute the energy consumption E_n^l (in Joules) of the user when the task is executed locally, we adopt the model in [1], [15], [33]. Specifically, E_n^l can be derived as $E_n^l = \kappa_n \beta_n \left(f_n^l\right)^2$, where κ_n is a coefficient relating to the chip's hardware architecture. According to the measurements in [33], we set $k_n = 5 \times 10^{-27}$. It is worth noting that t_n^l and E_n^l depend on unique features of the user n and the running application; therefore, they can be computed in advance.

The local computation overhead in terms of the computational time and energy consumption is computed as $Z_n^l = \lambda_n^t t_n^l + \lambda_n^e E_n^l$, where $\lambda_n^t \in [0,1]$ and $\lambda_n^e + \lambda_n^t = 1$ are respectively weighted parameters of the computational time and energy consumption of the user n. Similar to heterogeneous users, different users may have different values of λ_n^t and λ_n^e. The weighted parameters can affect the offloading decisions of users. Consider a network scenario with three users as an example, where the first user with a latency-sensitive application sets $\lambda_n^t = 1$ and $\lambda_n^e = 0$, the second user with an energy-hungry application and low battery state can set $\lambda_n^t = 0$ and $\lambda_n^e = 1$, and the third user can set $0 < \lambda_n^t, \lambda_n^e < 1$ if it takes both computational time and energy consumption into consideration of the offloading decision.

D. Computation Model of MEC Servers

To offload the computation task, a user incurs extra overhead in terms of the energy consumption and latency. The extra overhead in time is composed of the transmission time from the users to the server, the execution time at the server, and the response time of the computational result back to the user. The extra overhead in energy consumption includes the energy consumption for computation offloading, remote execution, and transmission of the computational result back to the user. Since our current work focuses on the user perspective and since MEC servers are generally powered by cable power supply [10], [11], we ignore the energy consumption for remote execution. Moreover, the computational result is relatively small compared to the input data, so the response time and consumed energy consumption are neglected.

The time and energy costs for offloading the computation task I_n are, respectively, computed as

$$t_n^{\text{off}}(\boldsymbol{A}, \boldsymbol{P}) = \sum\nolimits_{m \in \mathcal{M}} \frac{\alpha_n a_{nm}}{R_{nm}}, \forall n \in \mathcal{N}, \tag{5}$$

where $a_{nm} = \sum_{s \in \mathcal{S}} a_{nm}^s$, and

$$E_n^{\text{off}}(\boldsymbol{A}, \boldsymbol{P}) = p_n t_n^{\text{off}} = \frac{p_n}{\zeta_n} \alpha_n \sum\nolimits_{m \in \mathcal{M}} \frac{a_{nm}}{R_{nm}}, \forall n \in \mathcal{N}, \tag{6}$$

where $p_n = \boldsymbol{p}_n^T \boldsymbol{1}$ with $\boldsymbol{X}^T$ being the normal transpose $\boldsymbol{X}$, ζ_n is the power amplifier efficiency of the user n. The execution time of the computation task I_n is given by $t_n^{\text{exe}}(\boldsymbol{A}, \boldsymbol{F}_n) = \sum_{m \in \mathcal{M}} \frac{a_{nm}\beta_n}{f_{nm}}, \forall n \in \mathcal{N}$, where f_{nm} is the computation capability (in CPU cycles per second) that is assigned to the user n by the server m. Here, $\boldsymbol{F} = \{\boldsymbol{F}_n | n \in \mathcal{N}\}$ is the computation resource profile, where $\boldsymbol{F}_n = \{f_{nm} | \forall m \in \mathcal{M}\}$ is the computation resource vector of the user n.

Since each MEC server typically has the limited computation capability, the total computation resources assigned to offloading users cannot exceed its maximum computation capability $f_m^{\max}$, i.e., the constraint, $\sum_{n \in \mathcal{N}} f_{nm} \leq f_m^{\max}, \forall m \in \mathcal{M}$, must be satisfied. Similar to the local computation overhead, the remote computation overhead can be computed as $Z_n^r(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}_n) = \lambda_n^t \left(t_n^{\text{off}} + t_n^{\text{exe}}\right) + \lambda_n^e E_n^{\text{off}}$.

E. Problem Formulation

Since our focus is to minimize the system-wide computation overhead in terms of the computational time and energy consumption, the objective function is defined as $Z(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}) = \sum_{n \in \mathcal{N}} Z_n(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}_n)$, where Z_n is given by $Z_n = \left(1 - \sum_{m \in \mathcal{M}} a_{nm}\right) Z_n^l + \left(\sum_{m \in \mathcal{M}} a_{nm}\right) Z_n^r, \forall n \in \mathcal{N}$.

For a given offloading decision profile $\boldsymbol{A}$, power allocation $\boldsymbol{P}$, and computation resource $\boldsymbol{F}$, as well as the objective function $Z(\cdot)$, we have the optimization formulation problem of joint computation offloading decision and resource allocation (OPT-JCORA) as follows:

$$\begin{aligned}
&\min_{\{\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}\}} Z(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}) \\
&\text{s.t. C1: } 0 < p_n^s \leq p_n^{\max}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}_{\text{off}} \\
&\quad \text{C2: } a_{nm}^s = \{0,1\}, \forall n \in \mathcal{N}, m \in \mathcal{M}, s \in \mathcal{S} \\
&\quad \text{C3: } \sum\nolimits_{m \in \mathcal{M}} \sum\nolimits_{s \in \mathcal{S}} a_{nm}^s \leq 1, \forall n \in \mathcal{N} \\
&\quad \text{C4: } \sum\nolimits_{n \in \mathcal{N}} a_{nm}^s \leq 1, \forall m \in \mathcal{M}, s \in \mathcal{S} \\
&\quad \text{C5: } \sum\nolimits_{n \in \mathcal{N}} \sum\nolimits_{s \in \mathcal{S}} a_{nm}^s \leq q_m, \forall m \in \mathcal{M} \\
&\quad \text{C6: } f_{nm} > 0, \forall n \in \mathcal{N}_m, \forall m \in \mathcal{M} \\
&\quad \text{C7: } \sum\nolimits_{n \in \mathcal{N}} f_{nm} \leq f_m^{\max}, \forall m \in \mathcal{M},
\end{aligned} \tag{7}$$

where $p_n^{\max}$ is the maximum transmit power of the user n, and $\mathcal{N}_{\text{off}} = \bigcup_{m \in \mathcal{M}} \mathcal{N}_m$ is the set of offloading users that are not able to compute their tasks locally. In the optimization formulation (7), if $n \notin \mathcal{N}_{\text{off}}$, $p_n = 0$, i.e., the user n executes the task locally. In addition, if $n \notin \mathcal{N}_m$, $f_{nm} = 0$, i.e., the MEC server m does not assign any computation resource to the user n. The constraint C1 makes sure that the transmit power of the user n does not exceed the maximum value. The constraint C2 denotes that $\boldsymbol{A}$ is a binary vector. The constraints C3, C4, and C5 guarantee that a computation task is executed either locally or remotely, a subchannel is assigned to at most one user, and the maximum number of users q_m can offload to an MEC server m, as explained in (1), (2), and (3), respectively. The constraint C6 denotes that the assigned computation resource from an MEC server to a user is positive, and the constraint C7 makes sure that the total computation resource of an MEC server assigned to offloading users does not exceed its maximum value $f_m^{\max}$.

The considered problem (7) is difficult to solve due to the following reasons:

- There exist relationships among $\boldsymbol{A}$, $\boldsymbol{P}$, and $\boldsymbol{F}$. Further, the data rate of users in the denominator in (5) and (6) makes the objective function highly complicated. Thus, the objective function $Z(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F})$ is not a convex function.
- There are three set of optimization variables: offloading decision $\boldsymbol{A}$, power allocation $\boldsymbol{P}$, and computation resource $\boldsymbol{F}$. $\boldsymbol{P}$ and $\boldsymbol{F}$ are continuous variables while $\boldsymbol{A}$ is a binary variable; therefore, the feasible solution set of the problem (7) is not convex.

To enable distributed computation offloading, in the next section, we propose to decompose the problem (7) into two parts: computation offloading decision and joint resource allocation.

III. Proposed Algorithm

A. Problem Decomposition

The OPT-JCORA is a mixed-integer and non-linear optimization problem since the offloading decision $\boldsymbol{A}$ is an binary vector and $\boldsymbol{P}$ and $\boldsymbol{F}$ are continuous vectors. Moreover, the OPT-JCORA problem is NP-Hard [15], [34]. Hence, it is hard to obtain the global centralized optimal solution. It is observed that the resource constraints C1, C6, and C7 are decoupled from the offloading constraints C2, C3, C4, and C5. Therefore, we proceed with solving the problem (7) by adopting a suboptimal approach and splitting it into two parts: (i) computation offloading (CO) decision and (ii) joint computation and communication resource allocation (JCCRA). The CO subproblem is written as follows:

$$\begin{aligned} &\min_{\boldsymbol{A}} Z(\boldsymbol{A}) && (8)\\ &\text{s.t. C2, C3, C4, C5,} \end{aligned}$$

where $Z(\boldsymbol{A})$ is the objective value of the JCCRA subproblem, which is formulated as

$$\begin{aligned} &\min_{\boldsymbol{P},\boldsymbol{F}} \sum\nolimits_{n\in\mathcal{N}_{\text{off}}} Z_n(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{F}_n) && (9)\\ &\text{s.t. C1, C6, C7.} \end{aligned}$$

By solving two subproblems (8) and (9) sequentially in each iteration until convergence, the solution to the underlying problem (7) can be finally obtained. The proposed framework for solving the OPT-JCORA problem is summarized in Fig. 2.

Next, we present our algorithm to solve the OPT-JCORA problem by solving first the CO subproblem via two matching games and then deriving the solution for the JCCRA subproblem. After that, we propose an algorithm for the OPT-JCORA problem. Besides, we also provide detailed analyses of the matching stability, convergence, and computational complexity of the proposed algorithm.

B. Computation Offloading As a Matching Problem

We consider the offloading decision problem for a given $\boldsymbol{P}$ and $\boldsymbol{F}$ by solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{A}} Z(\boldsymbol{A}) && (10)\\ &\text{s.t. C2, C3, C4, C5}\\ &\quad p_n^s = p_n^{\max}/S, \forall n \in \mathcal{N} && (11)\\ &\quad f_{nm} = f_m^{\max}/q_m, \forall n \in \mathcal{N}, m \in \mathcal{M}. && (12) \end{aligned}$$

In order to evaluate the average contribution of each user to the total objective, the transmit power of each user is divided equally among all the subchannels and the total computation resources of each server is uniform among the maximum q_m offloading users, i.e., $p_n^s = p_n^{\max}/S$ and $f_{nm} = f_m^{\max}/q_m$, as illustrated (11) and (12), respectively. This approach has been widely used in literature for scheduling, user association, and resource allocation in OFDMA networks [16], [35], [36], where simulation results showed that the achieved gains are negligible compared to that of optimal power allocation.

The CO subproblem is a mixed integer nonlinear programming and an NP-Hard problem [15], [34]. To solve the CO problem, the objective (10) is boiled down to the following:

$$\begin{aligned} Z(\boldsymbol{A}) = &\sum_{n\in\mathcal{N}} Z_n^l + && (13)\\ &\sum_{n\in\mathcal{N}} a_n \left[\sum_{m\in\mathcal{M}} a_{nm} \left(\frac{\lambda_n^t \alpha_n + \lambda_n^e p_n \alpha_n \zeta_n^{-1}}{R_{nm}} + \frac{\lambda_n^t \beta_n}{f_{nm}} \right) - Z_n^l \right], \end{aligned}$$

where $a_n = \sum_{m\in\mathcal{M}} a_{nm}$, whose value denotes the user n's offloading decision, i.e., $a_n = 1$ ($a_n = 0$) indicates that the user n offloads (executes locally). Recall that this paper considers binary offloading. To be executed remotely, users must benefit from computation offloading in terms of the execution latency and/or energy consumption. From (13), the user n executes the task locally if the following condition holds

$$\Upsilon_n := \left(\frac{\lambda_n^t \alpha_n + \lambda_n^e p_n^{\min} \alpha_n \zeta_n^{-1}}{R_{nm}^{\max}} + \frac{\lambda_n^t \beta_n}{f_0} \right) - Z_n^l \geq 0, \quad (14)$$

where $R_{nm}^{\max} = B_s \log_2\left(1 + p_n \max_{m\in\mathcal{M}, s\in\mathcal{S}} \{h_{nm}^s\} / n_0\right)$ and $f_0 = \max_{m\in\mathcal{M}} \{f_m^{\max}\}$. Let $\mathcal{N}_{\text{loc}} = \{n \in \mathcal{N} | \Upsilon_n \geq 0\}$ be the set of users that execute their tasks locally, and $\mathcal{N}_{\text{pof}} = \{n \in \mathcal{N} | \Upsilon_n < 0\}$ be the set of users that potentially offload their tasks to the MEC servers. To find an offloading solution for the remaining users, we temporarily assume that $a_n = 1, \forall n \in N_{\text{pof}}$, ignore the fixed parts, and rewrite the objective function in (13) as follows:

$$Z(\boldsymbol{A}) = \sum_{n=1}^{|\mathcal{N}_{\text{pof}}|} \sum_{m=1}^{M} a_{nm} \left(\overbrace{\frac{\lambda_n^t \alpha_n + \lambda_n^e p_n \alpha_n \zeta_n^{-1}}{\sum\limits_{s=1}^{S} a_{nm}^s B_s \log_2\left(1 + \Gamma_{nm}^s\right)} + \frac{\lambda_n^t \beta_n}{f_{nm}}}^{Z_{nm}} \right). \quad (15)$$

In the case of offloading, each user needs to select the server for the remote computation and the subchannel for the task offloading. From (15), if Z_{nm} is given for all $n \in \mathcal{N}_{\text{pof}}, m \in \mathcal{M}$, each user n is required to choose $a_{nm}, m \in \mathcal{M}$ such that $Z(\boldsymbol{A})$ is minimized. Further, if a_{nm}

is given for the user n, it is required to select $a^s_{nm}, s \in \mathcal{S}$ such that the offloading rate R_{nm} is maximized, thus minimizing $Z(\boldsymbol{A})$. Therefore, the computation offloading decision problem can now be further divided into two subproblems: 1) *which MEC server does a user offload to* and 2) *which subchannel does a user utilize to offload the task*. The first subproblem finds the matching $\boldsymbol{X} = [a_{nm}]_{|\mathcal{N}_{\text{pof}}| \times M}$ between $|\mathcal{N}_{\text{pof}}|$ users and M servers so as to minimize the overhead Z_{nm}. For a given $\boldsymbol{X}$, the second subproblem is to determine the matching $\boldsymbol{Y} = [a^s_{nm}]_{|\mathcal{N}_{\text{pof}}| \times M \times S}$ between users in $\mathcal{N}_m, \forall m \in \mathcal{M}$, that offload their tasks to the same MEC server and subchannels $\mathcal{S}$ in order to maximize the achievable offloading rate R^s_{nm}.

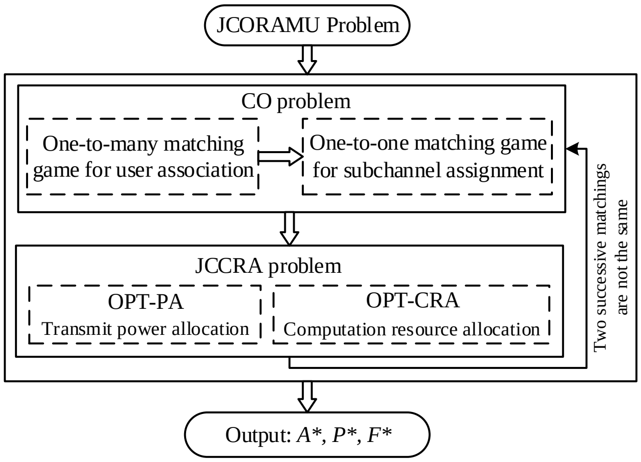

JCORAMU Problem
CO problem
One-to-many matching game for user association
One-to-one matching game for subchannel assignment
JCCRA problem
OPT-PA
Transmit power allocation
OPT-CRA
Computation resource allocation
Two successive matchings are not the same
Output: A*, P*, F*

Fig. 2: Proposed framework.

In what follows, we propose distributed processes based on a one-to-many matching game to find which MEC server a user offloads to, i.e, the user association, and based on a one-to-one matching game to find which subchannel a user utilizes to offload the task, i.e., the subchannel assignment, [29], [30]. In the first matching game, there are two types of players: users and servers. The strategy of users is to select the best MEC server to maximize the benefit of computation offloading and the strategy of servers is to either accept or reject the offloading requests from users. From the constraints, each user can offload the task to at most one MEC server according to C3 and each MEC server can execute multiple tasks from users according to C5. The two types of players in the second matching game are users and subchannels. The strategy of users is to select the most preferred subchannel to maximize the offloading rate, and that of the subchannels is to either accept or reject the bids from users. In this second game, each user is assumed to offload on at most one subchannel and each subchannel can be matched with at most one offloading user. These two matching games are inspired by the stable marriage problem: there are two sets of matching players (one for men and one for women), who seek to be married to each other and a matching comprises (man, woman) pairs [28]. Similarly in our work, a matching consists of (user, server) pairs and (user, subchannel) pairs in the first many-to-one matching game and in the second one-to-one matching game, respectively.

1) Matching Game for User Association:

Definition 1. *Given two disjoint sets of players, $\mathcal{M}$ and $\mathcal{N}_{pof}$, a one-to-many matching function $\Psi : \mathcal{N}_{pof} \to \mathcal{M}$ is defined such that for all $n \in \mathcal{N}_{pof}$ and $m \in \mathcal{M}$*

1) *$\Psi(m) \subseteq \mathcal{N}_{pof}$ and $|\Psi(m)| \leq q_m$;*
2) *$\Psi(n) \subseteq \mathcal{M}$ and $|\Psi(n)| \in \{0, 1\}$;*
3) *$m = \Psi(n) \leftrightarrow n = \Psi(m)$.*

This matching game is defined by a tuple $(\mathcal{M}, \mathcal{N}_{\text{pof}}, \Pi)$, with $\Pi = \{q_m | \forall m \in \mathcal{M}\}$ being the MEC servers' quota vector. The first condition implies that each MEC server m can execute at most q_m computation tasks as in C5, the second condition indicates that each user can offload the task to at most one server, and the third condition implies that if the user n is matched with the MEC server m, then the server m is also matched with the user n. The output of this game is a user association mapping Ψ between users and servers.

Next, we define $\phi_{n,\text{UA}}(m)$ as the preference of the user n for MEC server m and $\phi_{m,\text{UA}}(n)$ as the preference of MEC server m for the user n. We also define $\succ_{n,\text{UA}}$ and $\succ_{m,\text{UA}}$ as the preference relations of the user n and MEC server m, respectively. The notation $m_1 \succ_{n,\text{UA}} m_2$ implies that the user n prefers the MEC server m_1 to m_2, i.e., $\phi_{n,\text{UA}}(m_1) > \phi_{n,\text{UA}}(m_2)$. Similarly, the notation $n_1 \succ_{m,\text{UA}} n_2$ implies that the MEC server m prefers the user n_1 to n_2 if the computation overhead with n_1 is smaller than that with n_2, i.e., $\phi_{m,\text{UA}}(n_1) < \phi_{m,\text{UA}}(n_2)$.

Preference of the mobile user: For user association problems in wireless multicell networks, the average SINR over all subchannels is considered as one of the most common criteria [16], [36], [37]. The preference value of the user n when it offloads the task I_n to the MEC server m is defined as

$$\phi_{n,\text{UA}}(m) = \varphi_{\text{UA}} \alpha_n^{-1} \log_2 \left(1 + \sum_{s \in \mathcal{S}} \Gamma^s_{nm} \right) + \varepsilon_{\text{UA}} \beta_n^{-1} f_{nm}, \tag{16}$$

where φ_{UA} and ε_{UA} are two weighted parameters and $f_{nm} = f^{\max}_m / q_m$. Intuitively, the user n prefers to offload to the server that provides the higher offloading rate and computation resource. Since we allow each server to accommodate more than one user, multiple users may coexist and share computation resources when they offload to the same server. Similar to [31], we assume that the coexisting users share the server's CPU rate equally with effect from their computation workloads, i.e., each user assumes a share of $f^{\max}_m / \beta_n q_m$. Thus, the preference of user is based on the aggregation of the transmission delay (1st component) and remote processing time (2nd component).

Preference of the MEC server: We define the preference value of the MEC server m when it executes the computation task from the user n as the computation overhead as follows:

$$\phi_{m,\text{UA}}(n) = \frac{\lambda^t_n \alpha_n + \lambda^e_n p_n \alpha_n \zeta_n^{-1}}{R_{nm}} + \frac{\lambda^t_n \beta_n}{f_{nm}}, \tag{17}$$

where $R_{nm} = B_s \log_2 \left(1 + \sum_{s \in \mathcal{S}} \Gamma^s_{nm}\right)$. We say that the MEC server m prefers the user n_1 to n_2 (when n_1 and n_1 select the same MEC server m) if the user n_2 has lower computation overhead than the user n_2.

Now, we propose a distributed algorithm to find the matching between users and MEC servers while minimizing the computation overhead. The specific details of the proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2882800, IEEE Access

algorithm are given in Alg. 1, with new notations introduced below. First, the list of potential MEC servers for each user n is defined as $\mathcal{M}_n$ and initialized as $\mathcal{M}$, and the sets of unmatched users is $\mathcal{N}_{\text{unmatched}} = \mathcal{N}_{\text{pof}}$, the set of rejected and requested users for each MEC server m are defined as empty sets $\mathcal{N}_m^{\text{rej}}$ and $\mathcal{N}_m^{\text{req}}$, respectively. In addition, each user constructs its preference over all potential servers according to Eq. (16).

Algorithm 1 Users - MEC servers matching algorithm

```
1:  Initialization
      1) $\mathcal{M}$, $\mathcal{N}_{\text{pof}}$, $b_{nm}^{\text{UA}} = 0, \forall n \in \mathcal{N}_{\text{pof}}$.
      2) Construct the preference for all users in $\mathcal{N}_{\text{pof}}$ via (16) and the preference list $\mathcal{M}_n = \mathcal{M}, \forall n \in \mathcal{N}_{\text{pof}}$, set $\mathcal{N}_{\text{unmatched}} = \mathcal{N}_{\text{pof}}$, and initialize the list of requested users $\mathcal{N}_m^{\text{req}} = \emptyset$ and the list of rejected users $\mathcal{N}_m^{\text{rej}} = \emptyset, \forall m \in \mathcal{M}$.
2:  Find a stable matching $\Psi^*$
3:  while $\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_{\text{pof}}} b_{nm}^{\text{UA}} \neq 0$ do
4:      for $n = 1$ to $|\mathcal{N}_{\text{unmatched}}|$ do
5:          Find $m = \text{argmax}_{m\in\mathcal{M}_n} \phi_{n,\text{UA}}(m)$.
6:          Send a request to the server $m$ by setting $b_{nm}^{\text{UA}} = 1$.
7:      end for
8:      for $m = 1$ to $M$ do
9:          Update $\mathcal{N}_m^{\text{req}} \leftarrow \{n : b_{nm}^{\text{UA}} = 1, \forall n \in \mathcal{N}_{\text{pof}}\}$.
10:         Construct the preference according to (17).
11:         if $|\mathcal{N}_m^{\text{req}}| \leq q_m$ then
12:             Update $\mathcal{N}_m \leftarrow \mathcal{N}_m^{\text{req}}$.
13:         else
14:             repeat
15:                 Accept $n = \underset{n\in\succ_{m,\text{UA}}}{\text{argmax}} \sum_{n\in\mathcal{N}_m^{\text{req}}} \phi_{m,\text{UA}}(n)$.
16:                 Update $\mathcal{N}_m \leftarrow \mathcal{N}_m \cup n$.
17:             until $|\mathcal{N}_m| = q_m$
18:         end if
19:         Update $\mathcal{N}_m^{\text{rej}} \leftarrow \{\mathcal{N}_m^{\text{req}} \setminus N_m\}$.
20:         Update $\mathcal{M}_n \leftarrow \{\mathcal{M}_n \setminus m\}, \forall n \in \mathcal{N}_m^{\text{rej}}$.
21:     end for
22:     Update $\mathcal{N}_{\text{unmatched}} \leftarrow \mathcal{N}_{\text{unmatched}} \cap \{\mathcal{N}_1^{\text{rej}} \cup ... \cup \mathcal{N}_M^{\text{rej}}\}$.
23: end while
24: End of the algorithm: output is a stable matching $\Psi^*$.
```

Next, each user n decides the best MEC server, which has the largest preference among $\mathcal{M}_n$, by using its preference relation in line 5 and sends a bit request b_{nm}^{UA} to the MEC server m. The bid function b_{nm}^{UA} is 1 if the user m offloads to the MEC server m and 0 otherwise (line 6).

After bidding of all users, each MEC server collects the bid requests from users and updates the list of requested users (line 9) and constructs the preference over all requested users via Eq. (17). The MEC server m is able to accept all of the requested users if the number of requested users is smaller than its quota q_m (lines 12), otherwise it will select q_m among $\left|\mathcal{N}_m^{\text{req}}\right|$ requested users (lines 15 and 16). Unmatched users are then inserted into the set of rejected users of each MEC server. Meanwhile, each MEC server is removed from the preference list of its rejected users. Finally, the list of unmatched users is updated (line 22). Once there is no further requested user (condition checking at line 3), the algorithm stops.

If a stable matching does not exist, it is computationally difficult to find the solution. Fortunately, the outcome of Alg. 1 is a stable matching Ψ^*. To explain how the matching Alg. 1 achieves a stable matching, we present the definitions of a blocking pair and a stable matching in Definitions 2 and 3, respectively [30], [38].

Definition 2 (Blocking Pair)**.** *The pair (m_0, n_0) is a blocking pair for the matching Ψ, only if $m_0 \succ_{n,\text{UA}} m$, $m \in \Psi(n_0)$ and $n_0 \succ_{m,\text{UA}} n$, $n \in \Psi(m_0)$, for $m_0 \notin \Psi(n_0)$ and $n_0 \notin \Psi(m_0)$. In other words, there exists a partnership (m_0, n_0) such that m_0 and n_0 are not matched with each other under the current matching Ψ but prefer to be matched with each other.*

Definition 3 (Stable Matching)**.** *A matching Ψ is said to be stable if it admits no blocking pair.*

Theorem 1. *The matching Ψ^* produced by Alg. 1 is stable and ensures a local optimal solution to the CO problem.*

Proof. For a given transmit power allocation, computation resource allocation, and subchannel assignment, the preference of each user and MEC server is fixed. Therefore, Alg. 1 is known as the deferred acceptance algorithm in the two-sided matching problem between users and MEC servers, which guarantees a stable matching [29]. The first part is proved.

At each iteration t, the outcome of Alg. 1 maps to a user-server association $\boldsymbol{X}^{(t)}$, which captures to the objective

$$Z\left(\boldsymbol{X}^{(t)}\right) = \sum_{n=1}^{|\mathcal{N}_{\text{pof}}|} \sum_{m=1}^{M} x_{nm}^{(t)} \left(\frac{\lambda_n^t \alpha_n + \lambda_n^e p_n \alpha_n \zeta_n^{-1}}{R_{nm}} + \frac{\lambda_n^t \beta_n}{f_{nm}} \right).$$

Assume there exists a blocking pair (m_0, n_0) at iteration t such that the preference of the MEC server m and user n can be improved when (m_0, n_0) is added to the current matching Ψ. Accordingly, $Z(\boldsymbol{X}^{(t)}) > Z(\boldsymbol{X}^{(t+1)})$, i.e., the computation overhead is reduced. According to Definition 3, there is no blocking pair at the final matching. As a result, the matching algorithm converges to a local optimal solution to the CO problem. The second part is proved. □

2) Matching Game for Subchannel Assignment: After determining the user association mapping Ψ or $\mathcal{N}_m, m \in \mathcal{M}$, the one-to-one matching game is defined to find the subchannel assignment[2]. It is shown in (15) that for a given UA mapping (i.e., a_{nm} is given for all users) each user tries to maximize its transmission rate to the associated server in order to minimize the computation overhead Z(A). Therefore, the subchannel assignment problem for each user can be determined by solving the following problem:

$$\begin{aligned} \max_{\boldsymbol{Y}} & \sum_{s=1}^{S} \left[R_{nm}^s = B_s \log_2 \left(1 + \Gamma_{nm}^s\right)\right] \\ \text{s.t. } & a_{nm}^s = \{0,1\}, \forall n \in \mathcal{N}_m, s \in \mathcal{S} \\ & \textstyle\sum_{s\in\mathcal{S}} a_{nm}^s \leq 1, \forall n \in \mathcal{N}_m \\ & \textstyle\sum_{n\in\mathcal{N}_m} a_{nm}^s \leq 1, s \in \mathcal{S} \\ & p_n^s = p_n^{\max}/S, \forall n \in \mathcal{N}_m. \end{aligned} \tag{18}$$

[2] Here, we omit the subscript of MEC server m and consider the matching for the users in $\mathcal{N}_m$ and the set of subchannel $\mathcal{S}$.

Definition 4. *Given two disjoint sets of players, $\mathcal{N}_m$ and $\mathcal{S}$, a one-to-one matching function $\Omega : \mathcal{N}_m \to \mathcal{S}$ is defined such that for all $n \in \mathcal{N}_m$ and $s \in \mathcal{S}$*

1) $\Omega(s) \subseteq \mathcal{N}_m$ *and* $|\Omega(s)| \in \{0, 1\}$*;*
2) $\Omega(n) \subseteq \mathcal{S}$ *and* $|\Omega(n)| \in \{0, 1\}$*;*
3) $n = \Omega(s) \leftrightarrow s = \Omega(n)$*.*

The first two conditions ensure that each user can utilize at most one subchannel and a subchannel is assigned to at most one user and condition 3 implies that if the user n is matched with subchannel s, then subchannel s is also matched with the user n. The outcome of this one-to-one matching game is the association mapping Ω between the set of users $\mathcal{N}_m$ and the set of subchannels $\mathcal{S}$. Similar to the matching definition for user association, we define $\phi_{n,\mathrm{CA}}(s)$ as the preference of the user n for subchannel s and $\phi_{s,\mathrm{CA}}(n)$ as the preference of subchannel s for the user n. The notation $s_1 \succ_{n,\mathrm{CA}} s_2$ denotes that the user n prefers subchannel s_1 to s_2, i.e., $\phi_{n,\mathrm{CA}}(s_1) > \phi_{n,\mathrm{CA}}(s_1)$, and the notation $n_1 \succ_{s,\mathrm{CA}} n_2$ indicates that subchannel s prefers user n_1 to n_2, i.e., $\phi_{s,\mathrm{CA}}(n_1) > \phi_{s,\mathrm{CA}}(n_2)$.

Preference of the mobile user: After determining the MEC server selection, the user m achieves the following preference when it accesses to the subchannel s

$$\phi_{n,\mathrm{CA}}(s) = R_{nm}^{s}. \tag{19}$$

The preference in (19) implies that 1) subchannel selection of a user only affects the achievable offloading rate, which in turn determines the offloading time, as illustrated in (5), and 2) each user prefers to offload over the subchannel that offers a higher offloading rate.

Preference of the MEC server for subchannels: The preference of the MEC server m on subchannel s to be matched with the user n can be written as

$$\phi_{s,\mathrm{CA}}(n) = \varphi_{\mathrm{CA}} R_{nm}^{s} - \sum\nolimits_{m' \in \mathcal{M}, m' \neq m} \delta_{m'}^{s} g_{nm'}^{s} p_n^{s}, \tag{20}$$

where φ_{CA} and δ_m^s are two weighted coefficients. The preference in (20) implies that the SeNB m assigns the subchannel s to the user n so as to maximize the achievable offloading rate and minimize the aggregated interference to other SeNBs.

A distributed algorithm is designed to allocate subchannels of an SeNB to its associated users. The specific details of the algorithm are summarized in Alg. 2. First, we obtain the sets of users $\mathcal{N}_m$ from Alg. 1 and the subchannels $\mathcal{S}$, and initialize the set of unmatched users $\mathcal{N}_{\text{unmatched}}$, the set of potential subchannels for each user $\mathcal{S}_n$, the lists of requested users $\mathcal{N}_s^{\text{req}}$ and rejected users $\mathcal{N}_s^{\text{rej}}$ for each subchannel s. Each user n also constructs its preference over all potential subchannels $\mathcal{S}_n$ (step 3 in Initialization). Next, each user n selects the best subchannel s (line 5) and sends an access request for subchannel s to SeNB m (line 6). Here, the bid value b_{ns}^{CA} is set to 1 if the user n bids for the subchannel s and 0 otherwise. At SeNB m, the list of requested users to each subchannel s is updated (line 9). Then, the MEC server m selects the best user among the $\left|\mathcal{N}_s^{\text{req}}\right|$ requested users for each subchannel s (line 11) and assigns the subchannel s to that user (line 12). After that, the list of unmatched users for each subchannel s, $\mathcal{N}_s^{\text{rej}}$, is updated (line 13) and each subchannel s is removed from the list of potential subchannels of its rejected users $\mathcal{N}_s^{\text{rej}}$ (line 14). Based on the list of rejected users for all subchannels, the list of unmatched users is also updated (line 16). The algorithm stops if there is no further bidding between users $\mathcal{N}_m$ and subchannels $\mathcal{S}$ (line 3).

Algorithm 2 Users - subchannels matching algorithm

```
1:  Initialization
      1) $\mathcal{N}_m$, $\mathcal{S}$, $b_{ns}^{\mathrm{CA}} = 0, \forall n \in \mathcal{N}_m$.
      2) Set $\mathcal{N}_{\text{unmatched}} = \mathcal{N}_m$, $\mathcal{S}_n = \mathcal{S}, \forall n \in \mathcal{N}_m$, the list
         of requested users $\mathcal{N}_s^{\text{req}} = \emptyset$ and the list of rejected
         users $\mathcal{N}_s^{\text{rej}} = \emptyset, \forall s \in \mathcal{S}$.
      3) Construct the preference for all users in $\mathcal{N}_m$ via (19).
2:  Find a stable matching $\Omega^*$
3:  while $\sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}_m} b_{ns}^{\mathrm{UA}} \neq 0$ do
4:      for $n = 1$ to $|\mathcal{N}_{\text{unmatched}}|$ do
5:          Find $s = \mathrm{argmax}_{s \in \mathcal{S}_n} \phi_{n,\mathrm{CA}}(s)$.
6:          Send a request to the server $m$ by setting $b_{ns}^{\mathrm{CA}} = 1$.
7:      end for
8:      for $s = 1$ to $S$ do
9:          Update $\mathcal{N}_s^{\text{req}} \leftarrow \{n : b_{ns}^{\mathrm{CA}} = 1, \forall n \in \mathcal{N}_m\}$.
10:         Construct the preference via (20).
11:         Find $n = \mathrm{argmax}_{n \in \mathcal{N}_s^{\text{req}}} \phi_{s,\mathrm{CA}}(n)$.
12:         Assign the subchannel $s$ to the user $n$.
13:         Update $\mathcal{N}_s^{\text{rej}} \leftarrow \{\mathcal{N}_s^{\text{req}} \setminus n\}$.
14:         Update $\mathcal{S}_n \leftarrow \{\mathcal{S}_n \setminus s\}, \forall n \in \mathcal{N}_s^{\text{rej}}$.
15:     end for
16:     Update $\mathcal{N}_{\text{unmatched}} \leftarrow \mathcal{N}_{\text{unmatched}} \cap \{\mathcal{N}_1^{\text{rej}} \cup \ldots \cup \mathcal{N}_S^{\text{rej}}\}\}$.
17: end while
18: End of the algorithm: outcome is a stable matching $\Omega^*$.
```

Theorem 2. *The matching Ω^* generated by Alg. 2 is stable and can achieve a local maximum of the problem* (18)*.*

Proof. The definitions of a blocking pair and stable matching for the matching problem between users and subchannels are similar to those in Definition 2 and 3, respectively. For a given power allocation, computation resource allocation, and association matching Ψ^*, the matching in Alg. 2 has the nature of deferred acceptance. Thus, a stable matching Ω^* can be found by Alg. 2. Note that the outcome of Alg. 2 at each iteration t maps to a subchannel assignment $\boldsymbol{Y}^{(t)}$ and the objective for a $\boldsymbol{Y}^{(t)}$ is $R_n(\boldsymbol{Y}^{(t)}) = \sum_{s=1}^{S} R_{nm}^{s}(\boldsymbol{Y}^{(t)})$. Moreover, the matching at iteration $(t+1)$ guarantees that $R_n(\boldsymbol{Y}^{(t)}) \leq R_n(\boldsymbol{Y}^{(t+1)})$, i.e., the objective is monotonically improved during the matching process, thus reducing the computation overhead. Consequently, Theorem 2 is proved. □

C. Joint CCRA Subproblem

For a given $\boldsymbol{A}$, the objective function of the JCCRA subproblem can be rewritten as

$$\min_{\boldsymbol{P},\boldsymbol{F}} \left(\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} \frac{\lambda_n^t \alpha_n + \lambda_n^e u_n p_n}{R_{nm}} + \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m} \frac{\lambda_n^t \beta_n}{f_{nm}} \right), \tag{21}$$

where $u_n = \alpha_n \left(\zeta_n\right)^{-1}$. Observe from (21) that the first term is for the transmit power allocation of the users and the second term is for the computation resource allocation of the MEC servers. In addition, the constraints C1, C5, and C6 are decoupled in $\boldsymbol{P}$ and $\boldsymbol{F}$. Therefore, it is possible to further decompose the JCCRA subproblem into two subproblems of $\boldsymbol{P}$ and $\boldsymbol{F}$. The two following subsections are devoted to the optimization of transmit power of the users and the computation resource allocation of the MEC servers, respectively.

1) Transmit Power Allocation of Mobile Users: The optimization of the transmit power $\boldsymbol{P}$ can be written as

$$\begin{aligned}&\min_{\boldsymbol{P}} \sum_{(m,n)\in\mathcal{G}_s} \frac{\lambda_n^t \alpha_n + \lambda_n^e u_n p_n^s}{\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^s}\right)} \\ &\text{s.t. } 0 < p_n^s \le p_n^{\max}, \forall n \in \mathcal{G}_s,\end{aligned} \tag{22}$$

where $\mathcal{G}_s = \{(m,n) \mid n \in \Psi(m), n = \Omega(s), \forall \in \mathcal{N}_{\text{off}}, m \in \mathcal{M}, s \in \mathcal{S}\}$. In (22), we only consider users that offload to different MEC servers but on the same subchannel. Obviously, (22) is a non-linear fractional and non-convex problem due to the sum-of-ratios form of the objective function and the existence of inter-cell interference $I_{nm}^s = \sum_{(m',n')\in\mathcal{G}_s, n'\neq n} p_{n'}^s h_{n'm}^s$ among users in $\mathcal{G}_s$. One potential approach to the sum-of-ratios problem (22) relies on its transformation into a parametric convex programming problem [17]. However, we find the approximate upper bound[3] of I_{nm}^s such that (22) can be decomposed into individual subproblems for different offloading users.

Suppose that the transmit power of the user n is obtained by solving the following problem:

$$\begin{aligned}&\min_{p_n^s} \frac{\lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s}{\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^{s,0}}\right)} \\ &\text{s.t. } 0 < p_n^s \le p_n^{\max},\end{aligned} \tag{23}$$

where $I_{nm}^{s,0} = \sum_{(m',n')\in\mathcal{G}_s, n'\neq n} p_{n'}^{\max} h_{n'm}^s$. The problem (23) is still not easy to solve due to the fractional form of the objective function. In the following however, we show that the objective function of (23) is quasi-convex and the solution to (23) can be achieved using a bisection algorithm.

Theorem 3. *The objective function of* (23) *is quasiconvex.*

Proof. Let $\eta_n(p_n^s) = \frac{\lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s}{\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^{s,0}}\right)}$, which is the ratio of a linear function and a concave function, and its sublevel sets $S_a = \{p_n^s \in (0\ p_n^{\max}] \mid \eta_n(p_n^s) \le a\}, \forall a \in \mathbf{R}^+$. The set S_a can be equally expressed as

$$\begin{aligned}S_a = \{p_n^s \in (0\ p_n^{\max}] \mid\ &\lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s \\ &- a\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^{s,0}}\right) \le 0\}, \forall a \in \mathbf{R}^+.\end{aligned}$$

Let $f_n(p_n^s) = \lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s - a\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^{s,0}}\right)$. According to [39], S_a is a convex set if for any $\rho_1, \rho_2 \in S_a$ and any θ with $0 \le \theta \le 1$, we have

$$\theta\rho_1 + (1-\theta)\rho_2 \in S_a. \tag{24}$$

The condition (24) holds when $f_n\left(\theta\rho_1 + (1-\theta)\rho_2\right) \le 0$. Actually, $f_n(p_n^s)$ is a convex function due to the subtraction of a linear function and a concave function. By definition, $f_n\left(\theta\rho_1 + (1-\theta)\rho_2\right) \le \theta f_n(\rho_1) + (1-\theta) f_n(\rho_2)$. Due to $\rho_1, \rho_2 \in S_a$, we have $f_n(\rho_1) \le 0$ and $f_n(\rho_2) \le 0$. Therefore, the condition (24) holds, S_a is a convex set, and then $\eta_n(p_n^s)$ is a quasiconvex function. □

One approach to quasiconvex optimization is a bisection algorithm, which solves a convex feasibility problem at each step [39]. However, for further reduction of complexity, we use the approach proposed in [15] to solve the quasiconvex problem (23). The basic idea is that the optimal solution $\tilde{p}_n^{s,*}$ either lies at the border of the constraint or satisfies the constraint $\partial\eta_n\left(\tilde{p}_n^{s,*}\right)/\partial p_n^s = 0$. We have $\eta_n'\left(p_n^s\right) = \phi_n(p_n^s)/\left(\log_2\left(1+p_n^s h_{nm}^s/\left(n_0+I_{nm}^{s,0}\right)\right)\right)^2$, where

$$\begin{aligned}\phi_n(p_n^s) =&\lambda_n^e u_n B_s^{-1}\log_2\left(1+\frac{p_n^s h_{nm}^s}{n_0+I_{nm}^{s,0}}\right) \\ &- \frac{h_{nm}^s}{\ln 2}\frac{\lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s}{n_0+I_{nm}^{s,0}+p_n^s h_{nm}^s}.\end{aligned} \tag{25}$$

Moreover, the first-order derivative of (25) is expressed as

$$\phi_n'(p_n^s) = \frac{1}{\ln 2}\frac{\lambda_n^t \alpha_n B_s^{-1} + \lambda_n^e u_n B_s^{-1} p_n^s}{\left(\frac{n_0+I_{nm}^{s,0}}{h_{nm}^s}+p_n^s\right)^2}. \tag{26}$$

From (25) and (26), we have $\phi_n(0) < 0$ and $\phi_n'\left(\tilde{p}_n^s\right) > 0, \forall \tilde{p}_n^s \in (0\ p_n^{\max}]$, i.e., $\phi_n(\cdot)$ is a monotonically increasing function and diminishes at $\tilde{p}_n^s = 0$. Therefore, iteratively checking the condition $\phi_n(\tilde{p}_n^s) \le 0$, we can design an efficient bisection method as in Alg. 3. In each step, the interval is bisected, i.e., $p_n^{s,\mathrm{t}} = \left(p_n^{s,\mathrm{u}} + p_n^{s,\mathrm{l}}\right)/2$; therefore, the number of iterations required for Alg. 3 to terminate is $\lceil\log_2\left(p_n^{s,\mathrm{u}} - p_n^{s,\mathrm{l}}\right)/\varepsilon\rceil$.

Algorithm 3 Bisection method for the quasiconvex optimization problem.

```
Initialization
Set the tolerance $\varepsilon$, $p_n^{s,\mathrm{l}} = 0$, and $p_n^{s,\mathrm{u}} = p_n^{\max}$.
Compute $\phi_n\left(p_n^{s,\mathrm{u}}\right)$.
Find the optimal solution $\tilde{p}_n^{s,*}$
if $\phi_n\left(p_n^{s,\mathrm{u}}\right) \le 0$ then
    $\tilde{p}_n^{s,*} = p_n^{s,\mathrm{u}}$.
else
    repeat
        Set $p_n^{s,\mathrm{t}} = \left(p_n^{s,\mathrm{u}} + p_n^{s,\mathrm{l}}\right)/2$.
        if $\phi_n\left(p_n^{s,\mathrm{t}}\right) \le 0$ then
            $p_n^{s,\mathrm{l}} = p_n^{s,\mathrm{t}}$.
        else
            $p_n^{s,\mathrm{u}} = p_n^{s,\mathrm{t}}$.
        end if
    until $p_n^{s,\mathrm{u}} - p_n^{s,\mathrm{l}} \le \varepsilon$
    Set $\tilde{p}_n^{s,*} = \left(p_n^{s,\mathrm{u}} + p_n^{s,\mathrm{l}}\right)/2$.
end if
Output: the optimal solution $\tilde{p}_n^{s,*}$.
```

Note that the output of Alg. 3 is the approximate solution $\tilde{p}_n^{s,*}$ to the quasiconvex problem (23). After finding the solution to (23) for all users in $\mathcal{G}_s$, I_{nm}^s can be approximated as $\tilde{I}_{nm}^s = \sum_{n'\in\mathcal{G}_s, n'\neq n} \tilde{p}_{n'}^{s,*} h_{n'm}^s$. Replacing $I_{nm}^{s,0}$ in (23) with

[3] Upper bound of I_{nm}^s is due to the minimization problem (22).

$\tilde{I}^s_{nm}$, we obtain approximation problems for the power allocation of users. The transmit power of users is finally achieved by solving the approximation problems via the bisection Alg. 3.

2) Computation Resource Allocation of MEC servers: The computation resource allocation $\boldsymbol{F}$ is determined by solving the following optimization problem (OPT-CRA):

$$\begin{aligned}\min_{\boldsymbol{F}} \; & \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_m} \frac{\lambda^t_n \beta_n}{f_{nm}} \\ \text{s.t.} \; & \text{C6, C7.}\end{aligned} \tag{27}$$

The problem (27) can be decomposed into M individual problems, corresponding to M MEC servers. However, even with (27), we show that the optimal computation allocation of a single MEC server merely depends on the set of its associated users.

Theorem 4. *The OPT-CRA problem is a convex problem.*

Proof. It is clear that the feasible solution set of the OPT-CRA is convex. The remaining task is to show the convexity of the objective function. We have the following derivatives

$$\frac{\partial^2}{\partial f^2_{nm}} \left(\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_m} \frac{\lambda^t_n \beta_n}{f_{nm}} \right) = \frac{2\lambda^t_n \beta_n}{f^3_{nm}}, \forall n \in \mathcal{N}_{\text{off}}, m \in \mathcal{M}$$

$$\frac{\partial^2}{\partial f_{nm} \partial f_{kj}} \left(\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_m} \frac{\lambda^t_n \beta_n}{f_{nm}} \right) = 0, \forall (m, m) \neq (k, j).$$

Let $\nabla^2 g_l(\boldsymbol{F})$ be the Hessian matrix. Then, with all $\boldsymbol{v} \in \mathbb{R}^{N_{\text{off}}}$, $\boldsymbol{v}^T \nabla^2 g_l(\boldsymbol{F}) \boldsymbol{v} = \sum_{n\in\mathcal{N}_{\text{off}}} 2v^2_n \lambda^t_n \beta_n / f^3_{nm} \geq 0$, where the equality happens only if $\lambda^t_n = 0$, i.e., the user n is with an energy-hungry application. Therefore, the Hessian matrix is a positive semidefinite matrix. We conclude that OPT-CRA is a convex optimization problem [39]. □

Since OPT-CRA is a convex problem, the optimal solution can be optimally achieved via the KKT optimality conditions [40], [41]. In particular, the optimal computation resource for user n at MEC server m is calculated as follows:

$$f^*_{nm} = \frac{f^{\max}_m \sqrt{\lambda^t_n \beta_n}}{\sum_{n\in\mathcal{N}_m} \sqrt{\lambda^t_n \beta_n}}. \tag{28}$$

Remark 1. *It is revealed from* (28) *that the computation resource is determined by the weighted parameter λ^t_n and computation workloads β_n of all users. Here, λ^t_n can be interpreted as the importance level of computational time of the user n. Specifically, if all users have the same computation workload, i.e., $\beta_n = \beta_k, \forall n \neq k, n, k \in \mathcal{N}_m$, the larger the value of λ^t_n is, the more the computation resource should be assigned to the user n by the corresponding MEC server m in order to minimize the processing time.*

D. Algorithm for JCORA Problem

In this subsection, we propose a joint framework to find the optimal solution to the underlying problem (7). The details of the proposed algorithm are summarized in Alg. 4, which is referred to as JCORAMS (JCORA Multi-Server). In general, the proposed algorithm has three phases: the pre-computation offloading decision, computation offloading and resource allocation, and post-computation offloading decision. The purpose of the first phase is to filter out users who cannot benefit from computation offloading, i.e., those users who should execute their tasks locally, and to reduce the input dimension for the second phase, i.e., non-offloading users are not taken into account during the second phase.

Algorithm 4 JCORAMS algorithm.

```
Input: $\mathcal{M}$, $\mathcal{N}$, $\mathcal{S}$.
repeat
    Pre-computation offloading decision: each user $n$ computes $\Upsilon_n$ and checks the offloading condition (14) to determine the offloading decision.
    Computation offloading and resource allocation
    Matching between users and MEC servers
        Each user calculates its preference via (16).
        Each server constructs its preference via (17).
        Obtain the optimal matching $\Psi^*$ via Alg. 1.
    Matching for subchannel allocation in SeNBs $m \in \mathcal{M}$
        Each user has its preference over $S$ subchannels.
        Each subchannel gets its preference over $N_m$ users.
        Obtain the optimal matching $\Omega^*$ via Alg. 2.
    Transmit power allocation of users in $\mathcal{G}_s, \forall s \in \mathcal{S}$
        Each user in $\mathcal{G}_s$ finds $\tilde{p}^{s,*}_n$ and $\tilde{I}^s_{nm}$.
        Solve (23) with $I^{s,0}_{nm} = \tilde{I}^s_{nm}$ for the user $n$.
    Computation resource allocation at MEC servers
        Computation resource is allocated via (28).
    Post-computation offloading decision
        For $n \in \mathcal{N}_{\text{pof}}, m \in \Psi^*(n), s \in \Omega^*(n)$, compute $\Upsilon^*_n$
            $$\Upsilon^*_n = \frac{\lambda^t_n \alpha_n + \lambda^e_n p^*_n \alpha_n \zeta^{-1}_n}{R^{s*}_{nm^*}} + \frac{\lambda^t_n \beta_n}{f_{nm^*}} - Z^l_n.$$
        Each user checks the offloading decision $\Upsilon^*_n > 0$ to decide to offload ($a_n = 1$) or not offload ($a_n = 0$).
    MEC servers and users update the preference lists for the next iteration.
until $\Upsilon^*_n \leq 0, \forall n \in \mathcal{N}_{\text{pof}}$ for two consecutive times or $a^s_{nm} = 0, \forall n \in \mathcal{N}, m \in \mathcal{M}, s \in \mathcal{S}$.
Output: the optimal solution $(\boldsymbol{A}^*, \boldsymbol{P}^*, \boldsymbol{F}^*)$.
```

The second phase is further divided into four steps: user-server association, subchannel allocation, transmit power control, and computation resource allocation.

- *User-server association*: The users and MEC servers join a one-to-many matching via Alg. 1. The preference of a user over potential servers and the preference of a server over the $|\mathcal{N}_{\text{pof}}|$ users are calculated according to (16) and (17), respectively. The optimal user association is obtained via Alg. 1, where a user sends the computation offloading proposal to the most preferred server and a server accepts a number of preferred users based on its quota. Alg. 1 stops when every user is either accepted by one server or rejected by all preferred servers.
- *Subchannel allocation*: After all users know their associated MEC servers, users offloading to the same MEC

server join a one-to-one matching game. Each user in $\mathcal{N}_m$ calculates the preference over its preferred subchannels according to (19) and the MEC server m computes its preference on all subchannels over its associated users according to (20). A user sends the proposal to the most preferred subchannel and a SeNB assigns a subchannel to the most preferred user, who has the highest preference among requested users, and rejects the proposals of other users on that subchannel. Alg. 2 terminates when there is no bidding between users and subchannels.

- *Transmit power control*: Once two matching algorithms for user association and subchannel allocation terminate, the transmit power of offloading users is allocated. Note that there may be S groups $\mathcal{G}_s$ and the transmit power of users in the group $\mathcal{G}_s$ is achieved by solving the individual problem (23). The approximate transmit power of a user in $\mathcal{G}_s$ is found via Alg. 3 by fixing the inter-cell interference at the maximum transmit power of the other users. After that, the inter-cell interference of users can be well approximated, and these approximation problems are solved to find the transmit power of users in $\mathcal{G}_s$.
- *Computation resource allocation*: Computation resource allocation at MEC servers is executed when Alg. 2 terminates. Each MEC server m allocates the computation resources to its associated users in $\mathcal{N}_m$ according to (28).

The third phase acts as the second filter since we assume that all of the users in $\mathcal{N}_{\text{pof}}$ offload their tasks to the MEC servers. After the second phase in each iteration, it is necessary to determine whether or not the users benefit from computation offloading with resource allocation from the second phase. There are two scenarios for a user to not offload the computation task. The first is that the user is not accepted by any server and/or subchannel and the other is that the user does not benefit from computation offloading with the preferred server and subchannel. In these cases, rejected users in the previous iterations will be considered as new users in the next iterations. A user will not join the next matchings in the UA and CA steps if it is rejected in the previous iterations. Moreover, users that are served by an MEC server and benefit from computation offloading in the previous iterations will not be involved in the next matchings. However, the CCRA subproblem will consider both previously accepted users and currently proposed users.

To start a new iteration, at the end of the previous iteration, each user is required to update its potential servers and subchannels and calculate the preferences for the next matchings. Similarly, each server is required to remove the rejected users from the preference list, update the remaining quota and subchannel availability, and calculate the preferences for the next matching processes. The system-wide computation overhead decreases as more iterations are processed. The proposed algorithm converges and terminates when either the matching of two consecutive iterations remains unchanged or all users do not benefit from computation offloading, thus executing their tasks locally (line 22).

E. Convergence and Stability

In order to analyze the convergence and stability of the proposed algorithm, let $\mathcal{G}_s, s \in \mathcal{S}$ denote the group formed by Alg. 4 at the end of each iteration t and introduce the definition of group stable [16], [36]. Then, we show that matching games have group stability and the proposed algorithm converges to above group-stable points.

Definition 5 (Group Stable)**.** *The group $\mathcal{G}_s, \forall s \in \mathcal{S}$ is blocked by another group $\mathcal{G}'_{s'}$ if two following conditions hold:*

1) *No user inside $\mathcal{G}_s$ can leave it to join $\mathcal{G}'_{s'}$.*
2) *No user inside $\mathcal{G}'_{s'}$ can leave it to join $\mathcal{G}_s$,*

The group $\mathcal{G}_s$ is said to be group stable if it is not blocked by any group. In addition, matchings in the proposed algorithm are stable if and only if all groups $\mathcal{G}_s, \forall s \in \mathcal{S}$ are group stable.

We prove that the group $\mathcal{G}_s$ created by Alg. 4 at the end of an iteration t is group stable by contradiction. Now assume that there is a user n inside $\mathcal{G}_s$ can leave it to join another group $\mathcal{G}'_{s'}$. This implies that $\mathcal{G}_s$ formed at the end of the CA phase is not stable. However, according to Theorem 2, $\mathcal{G}_s$ is stable and there is no another matching $\mathcal{G}'_{s'}$ that can provide a higher preference for user n, i.e., $\phi^{m}_{s,\text{CA}}(n) > \phi^{m'}_{s',\text{CA}}(n)$. Therefore, no user inside $\mathcal{G}_s$ can leave it to join another group. Similarly, assume that there is a user n' outside $\mathcal{G}_s$ can join it. We consider two scenarios for this condition. First, if $(n, n') \in \mathcal{N}_m$, $n' \succ^{m}_{s,\text{CA}} n$. Nevertheless, since $\mathcal{G}_s$ is stable at the CA phase, $n \in \Omega_m(s)$ and the server m assigns the subchannel s to n instead of n'. Thus the user n' cannot join $\mathcal{G}_s$. Second, if $n \in \mathcal{N}_m, n' \in \mathcal{N}_{m'}$, and $m \neq m'$, we have $n \in \Omega_m(s)$ and $n' \in \Omega_{m'}(s)$. Then, given that the user n' has positive computation offloading gain, the post-computation offloading phase will assign n' to $\mathcal{N}_{m'}$ and add (n', m') to $\mathcal{G}_s$. However, because $\mathcal{G}_s$ is stable at the CA phase, the post-computation offloading phase will not accept n' to join $\mathcal{G}_s$. For those two scenarios, no user outside $\mathcal{G}_s$ can join it. As a consequence, we can conclude that each group $\mathcal{G}_s, s \in \mathcal{S}$ is not blocked by any other groups, thus making the matchings of the proposed algorithms are stable.

Theorem 5. *Matchings from the UA and CA games are all stable in each iteration of the proposed Alg. 4.*

Proof. The proof is similar to that in Appendix A in [16]. It is therefore omitted. □

Given the stable outcomes from the UA and CA matching games in each iteration, transmit power and computation resource for accepted users are optimized. Then, users are required to check the offloading conditions to decide to offload or not. This post-computation offloading process in the third phase of Alg. 4 can be considered as the matching between subchannels and pairs of users-servers. To observe the optimality property of this matching, we realize the definition of *weak Pareto optimality* (PO) [42]. Denote by $Z(\mathcal{G})$ the system-wide computation overhead obtained by the matching $\mathcal{G}$. The matching $\mathcal{G}$ is weak PO if there is no other matching $\mathcal{G}'$ with $Z(\mathcal{G}') \preceq Z(\mathcal{G})$, which is strict for one user [43].

Theorem 6. *The post-computation offloading phase achieves a weak PO solution for the CCRA subproblem in each iteration.*

Proof. We consider a matching $\mathcal{G}$ obtained by the post-computation offloading phase in any iteration t and assume

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2882800, IEEE Access

that there is a unstable matching $\mathcal{G}'$ that is better than $\mathcal{G}$, i.e., $Z(\mathcal{G}') < Z(\mathcal{G})$. Since $\mathcal{G}'$ is unstable, there exists at least one blocking pair. Let the matching between the subchannel s and the user-server (n, m) be a blocking pair. The reasons behind instability of $\mathcal{G}'$ are:

1) $Z_n > Z_n^l$, $a_n = 1$, $n \in \Psi(m)$, and $n \in \Omega_m(s)$,
2) $Z_n \leq Z_n^l$, $a_n = 0$, $n \in \Psi(m)$, and $n \in \Omega_m(s)$.

For case 1, the post-computation offloading phase will construct a new stable matching $\mathcal{G}$ by setting $a_n = 0$, i.e., the user n chooses to execute the task locally instead of offloading the computation task to the MEC server over the subchannel s. This decreases the computation overhead of the user n from Z_n^r to Z_n^l (i.e., a quantity of $\left(Z_n^r - Z_n^l\right) > 0$), thus resulting in $Z(\mathcal{G}) < Z(\mathcal{G}')$ since the computation overhead of other users is left unchanged in $\mathcal{G}'$. For case 2, the post-computation offloading phase will construct a new stable matching $\mathcal{G}$ by removing the user n from the set of local users ($a_n = 0$) and then assigning to the set of offloading users with the server m ($a_n = 1, n \in \mathcal{N}_m$). Since for the user n, the local execution with negative computation offloading gain is replaced by the remote execution with positive computation offloading gain, removing the user n from the set of local users can decrease the system-wide computation overhead or $Z(\mathcal{G}) < Z(\mathcal{G}')$.

Hence, under both cases, the unstable matching $\mathcal{G}'$ is replaced by the new stable matching $\mathcal{G}$. Accordingly, there is no unstable matching $\mathcal{G}'$ that can lower the system-wide computation overhead. Based on the definition of weak PO, we can conclude that the post-computation offloading phase produces a stable outcome and a weak PO solution for the CCRA subproblem in each iteration. □

Theorem 7. *The JCORAMS algorithm outputs a group stable $\mathcal{G}_s$ after a finite number of iterations and is guaranteed to converge.*

Proof. The numbers of preference relations of the users, MEC servers, and subchannels, i.e., $\succ_{n,\text{UA}}$, $\succ_{m,\text{UA}}$, $\succ_{n,\text{CA}}$, and $\succ_{s,\text{CA}}$, in each iteration are finite since the numbers of users, MEC servers, and subchannels are finite. According to Theorems 5 and 6, matchings generated by the JCORAMS algorithm in each iteration are proved to be all stable. Additionally, since only rejected users in the previous iterations are processed in the next iterations and no user is rejected by the same server and subchannel more than once in the UA and CA phases, respectively, the number of preference relations reduces after each iteration. The gradual finiteness of the preference relations ensures the convergence of the proposed algorithm after a finite number of iterations. □

F. Complexity Analysis

The optimal offloading decision can be obtained by the exhaustive search. All possible combinations of users, servers, and subchannels are searched and the one that minimizes the system-wide computation overhead is selected as the optimal solution. The exhaustive search for possible combinations has a complexity $\mathcal{O}((MS)^N)$. For each combination, the bisection method is run to find the approximated interference for each user and then to find the solutions for the approximation problems. Here, the complexity of achieving the transmit power for all users is $\mathcal{O}(2N \log_2 (p_n^{\max})/\varepsilon)$. The complexity of allocating computation resources for users at M servers is $\mathcal{O}(M)$. Since each user needs to check the offloading condition and decides to offload if it benefits from computation offloading, the complexity of finding the computation offloading decisions is $\mathcal{O}(N)$. Therefore, the complexity of the exhaustive search is $\mathcal{O}((MS)^N (M + N + 2N \log_2 (p_n^{\max})/\varepsilon))$. It can be seen that the complexity of the exhaustive search increases exponentially with N, which would not be affordable for MEC systems with a massive number of users.

To analyze the computational complexity of the proposed algorithm, we consider the worst case when the preferences of all users for all servers as well as the preferences of users offloading to the same server for all subchannels are the same. At the end of each iteration, the preference relations used in the next iteration are updated. The computational complexities to sort the preference list of a user and a server using a standard sorting algorithm are $\mathcal{O}(M \log(M))$ and $\mathcal{O}(N \log(N))$, respectively. Similarly, the complexity to sort the preferences for all users $\mathcal{N}_m$ and subchannels is $\mathcal{O}(N_m S \log(N_m S))$. The total lengths of the input preferences in the UA phase and CA phase are, respectively, $2NM$ and $2N_m S$. From [44], it can be shown that the stable matchings in the UA and CA phases can be obtained with the computational complexities of $\mathcal{O}(NM)$ and $\mathcal{O}(N_m S)$ respectively, which are both linear in the input size. Given the stable outputs from two matching games (i.e., computation offloading decisions are given), as aforementioned, the complexities of the JCCRA phase and the post-computation offloading phase are also linear. Since the proposed algorithm terminates after a finite number of iterations, the complexity of the proposed algorithm is finally linear in the numbers of users, servers, and subchannels. Compared with the exhaustive search, the proposed algorithm has a reasonable complexity and appears efficient in dense MEC systems.

IV. Numerical Simulation

A. Simulation Settings

In order to evaluate our proposed algorithm, we use the following simulation settings for all simulations. We first consider the scenario where 9 SeNBs are randomly deployed in a small indoor area of 250×250 m^2 to serve 36 users. Each SeNB consists of 4 subchannels and has a quota of 4 users ($q_m = 4, \forall m \in \mathcal{M}$) . The bandwidth of each subchannel is $B_s = 5$ MHz, each user has the maximum transmit power $p_n^{\max} = 100$ mW, and $n_0 = -100$ dBm. The path-loss model is $-140.7 - 36.7 \log_{10}(d)$, where d (kilometers) is the distance from the user to the serving SeNB. For the computation task, we adopt the face recognition application in [10], [15], [45], where the computation input data size is $\alpha_n = 420$ KB and the total required number of CPU cycles is $\beta_n = 1000$ Megacycles. The CPU computation capability f_n^l of the user n is randomly assigned from the set $\{0.5, 0.8, 1.0\}$ GHz [9], [10] and the computation capability of each MEC server $f_m^{\max} = 4.0$ GHz, $\forall m \in \mathcal{M}$. The weighted parameters of the computational time and energy consumption are both

0.5, i.e., $\lambda_n^t = \lambda_n^e = 0.5, \forall n \in \mathcal{N}$. Finally, we set the values of φ_{UA}, ε_{UA}, φ_{CA}, and δ_m^s ($\forall m \in \mathcal{M}, s \in \mathcal{S}$) to 8×10^6, 0.2, 1, and 0.1, respectively. For all the results, each plot is the average of 100 channel realizations and in each realization, the user and MEC server locations are uniformly distributed randomly.

B. Simulation Results

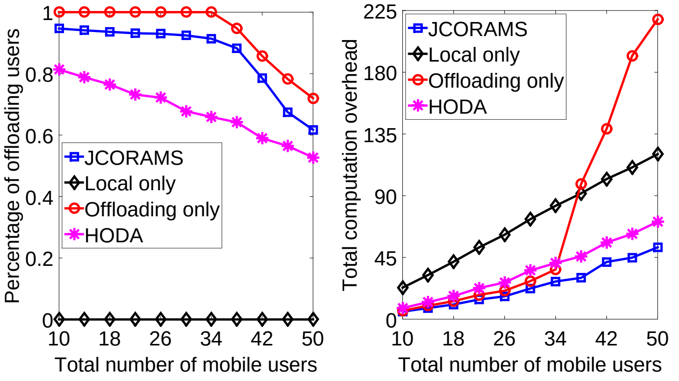
In the following, we present the performance of our proposed approach compared with several representative benchmark methods. For existing frameworks, the following solutions are considered:

1) *Local computing only*: there is no computation offloading. All users perform computations locally, i.e., $a_n = 0, \forall n \in \mathcal{N}$.
2) *Offloading only*: all users offload their computation tasks to the MEC servers i.e., $a_n = 1, \forall n \in \mathcal{N}$. This is achieved by running our algorithm without the pre-computation offloading decision and post-computation offloading decision steps. Note that when the number of users exceeds the system capacity, some requested users are rejected by the algorithm, i.e., $\exists n \in \mathcal{N} | a_n = 0$.
3) *HODA* [15]: the offloading decision, transmit power, and computation resources are determined in each cell independently.

To allow fair comparison between algorithms for single MEC server and multiple MEC servers, the final results, i.e., percentage of offloading users and system-wide computation overhead, do not take into account the inter-cell interference among offloading users.

In the first experiment, we vary the number of users from 10 to 50 with a step deviation of 4 and examine the percentage of offloading users. From Fig. 3a, the percentage of offloading users is relatively high[4] when the number of user is small. However, the percentage of offloading users gradually decreases when the number of users increases. This is reasonable since 1) each user might have a high probability to associate with its preferred MEC server and offload its computation task over a good subchannel and 2) small intercell interference makes users profit more from computation offloading. In addition, when the number of users keeps increasing, each user needs to compete with the others for using radio and computation resources, thus lowering the probabilities for each user to connect with its preferred MEC server and subchannel. Due to the limited number of MEC servers, number of subchannels in each cell, and quota of each SeNB, a portion of requested users must be rejected by the proposed algorithm and they must execute their tasks locally.

We also examine the performances in terms of the percentage of offloading users and system-wide computation overhead of our proposed approach and three compared frameworks. It is observed from Fig. 3a that the percentage of offloading users in the local computing only method is 0 while that of the offloading only approach is 1 and starts to decrease as N becomes smaller and greater than 36, respectively. This is due to the quota of each SeNB, the number of subchannels of each cell, and the number of MEC servers, and hence the system capacity in terms of the number of admitted offloading users is limited by $M \times \min\{q, S\}$. Fig. 3b also reveals that the performance of the offloading only algorithm becomes worse than that for the local computing only method when N gets larger. This is due to competition among users for the limited radio and computation resources. Compared with three baseline schemes, i.e., offloading only, local only, and HODA, our proposed algorithm can achieve better performance in terms of the percentage of offloading users and yield a lower computation overhead.



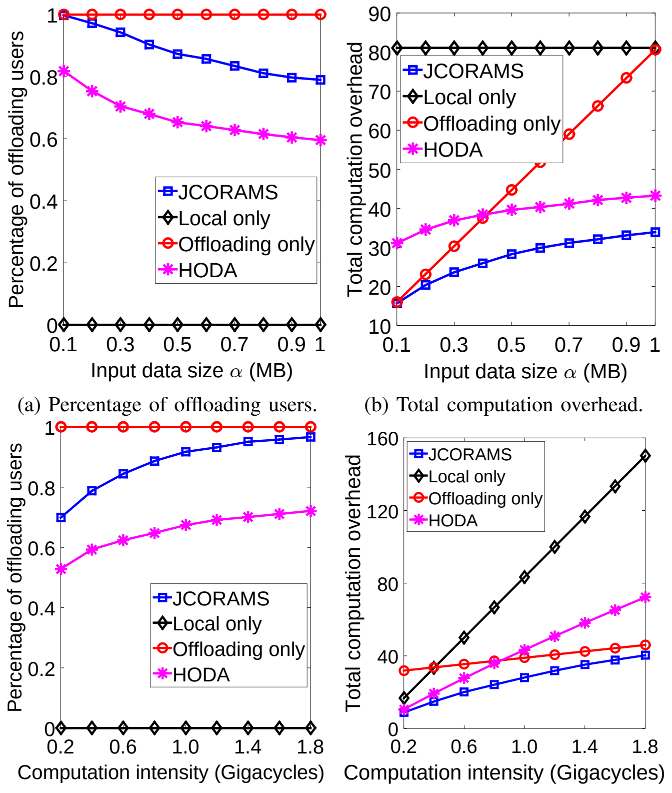
(a) Percentage of offloading users. (b) Total computation overhead.

Fig. 3: Comparison of JCORAMS and three baseline frameworks under different numbers of users.

The second experiment compares the performances of our proposed algorithm with the existing frameworks under different computation task profiles. It is shown in Figs. 4a-4b that when the input data size α is large enough (1 MB in this case), the computation overhead of the offloading only method can reach that of the local computing only scheme. It is therefore better to offload less computation tasks as α increases, i.e., a computation task with small data size is more preferable to computation offloading than one with high data size. The reason for this is that by increasing the input data size, the time cost and energy cost for offloading computation tasks become higher, as seen from Eqs. (5) and (6). This observation agrees with the performance lines of JCORAMS and HODA, where fewer users benefit from computation offloading and the system-wide computation overhead increases as α increases. From Figs. 4c-4d, we can observe that the percentage of offloading users and the system-wide computation overhead increase with the computation workload β. This is reasonable since both the local completion time and remote execution time increase as β increases; however, the computation capability of a user is often limited and an MEC server can offer offloading users with higher computation capability, i.e., users therefore benefit from computation offloading if their computation tasks are executed by the MEC servers. From the above reasons, it is better to offload a computation task with small input data size and large computation intensity rather

[4] The percentage of offloading users should be 1 when the number of users is relatively small. However, user and MEC server locations are both randomly generated in each simulation realization, so a user may not offload its computation task due to bad channel connections with MEC servers.

than one with large input data size and small computation intensity. Obviously, the proposed algorithm achieves the better performance than the baseline solutions.

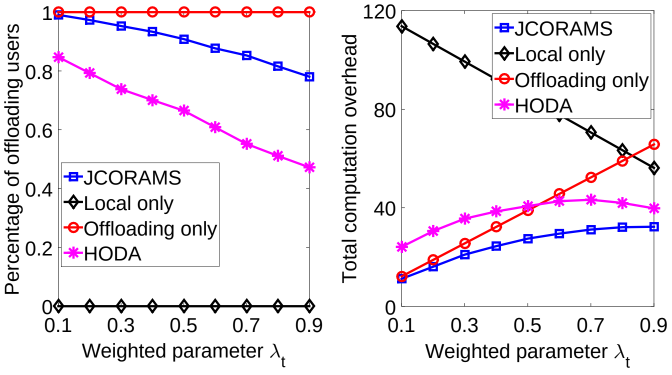


(a) Percentage of offloading users. (b) Total computation overhead.

(c) Percentage of offloading users. (d) Total computation overhead.

Fig. 4: Comparison of our proposed approach and three alternatives under different computation task profiles.

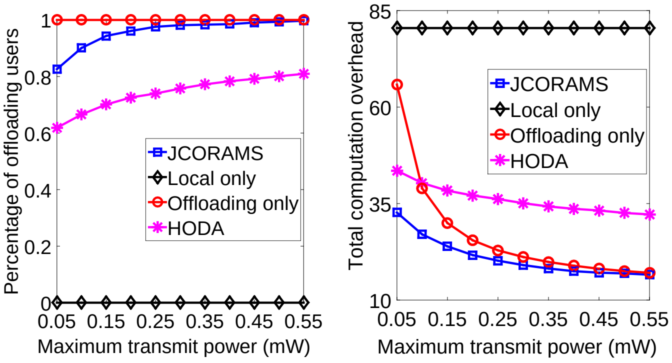
Next, by varying the weighted parameter of the computational time λ_t from 0.1 to 0.9 with a deviation step of 0.1 and setting the weighted parameter of the energy consumption λ_e to $1 - \lambda_t$, we further explore the performance comparison between our proposed algorithm and existing ones. It is worth noting that λ_t and λ_e are the same for all users; however, the extension to different λ_t and λ_e for different users does not affect the comparison among algorithms. Selecting a user with $f_l = 0.8$ GHz as an example, we have $t^l = 1000 \times 10^6/0.8 \times 10^9 = 1.25$ (seconds) and $E^l = 5 \times 10^{-27} \times 1000 \times 10^6 \times \left(0.8 \times 10^9\right)^2 = 3.20$ (Joules). It is obvious that for local computing, the energy consumption is nearly three times larger than the completion time. As a result, with the increment of λ_t, the system-wide computation overhead by the local only and offloading only schemes decreases and increases, respectively, as observed from Fig. 5b. In addition, there are fewer users that tend to offload their computation tasks to the MEC servers due to the lower local computation overhead, and the percentage of offloading users reduces, as shown in Fig. 5a. Here, the system-wide computation overheads by JCORAMS and HODA still increase and only start to decline when λ_t is large enough. The main reason for this is the dominance of the offloading and execution time ($t^{\text{off}} + t^{\text{exe}}$) over the offloading energy consumption (E^{off}). Therefore, selection of the weighted parameters plays an important role in the achieved performances. Again, our proposed algorithm is superior to the baseline solutions.



(a) Percentage of offloading users. (b) Total computation overhead.

Fig. 5: Comparison of our proposed approach and three alternative frameworks under different weighted parameters.

In the fourth experiment, we discuss the impacts of the maximum transmit power $p_n^{\max}$ on the performances of the considered approaches. From the Fig. 6, it is seen that when $p_n^{\max}$ increases, the percentage of offloading users and system-wide computation overhead increases and decreases, respectively, and all become saturated when $p_n^{\max}$ is sufficiently large. For example, with $N = 36$ and $p_n^{\max} = 0.55$ (W) for all users, the percentage of offloading users is 100%, i.e., the performances of our proposed and the offloading only schemes are the same, and the computation overhead is 17.9. This is due to the fact that increasing the offloading rates makes the time and energy costs for offloading the computation tasks smaller and as a consequence, there are more users that tend to offload their computation tasks to the MEC servers.

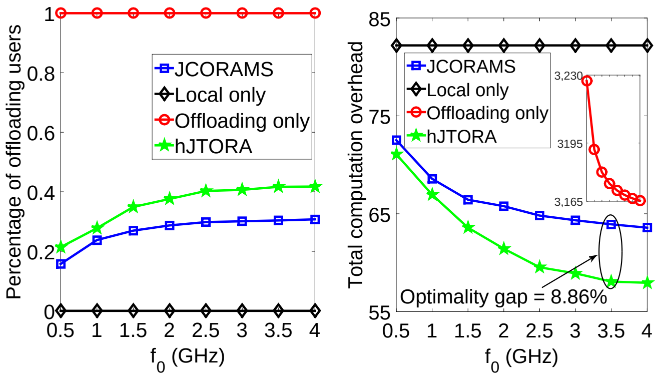


(a) Percentage of offloading users. (b) Total computation overhead.

Fig. 6: Comparison of our proposed approach and three existing frameworks under variant maximum transmit power.

The final experiment represents the number of offloading users and system-wide computation overhead for the different algorithms when the maximum computation capability f_0 of MEC servers varies from 0.5 GHz to 4 GHz. It is shown in Fig. 7a that the percentage of offloading users monotonically increases with f_0, and the increasing rate gradually decreases, i.e., increasing f_0 from 0.5 GHz to 1.0 GHz makes more users benefit from computation offloading than that from 1.0

GHz to 1.5 GHz. The reason is that when the computation capability of MEC servers is small, the execution time is high and so the remote computation overhead becomes higher than the local computation overhead. At the same time, because more users benefit from computation offloading, the system-wide computation overhead decreases. In order to evaluate the optimality of the proposed algorithm, we compare JCORAMS with the hJTORA algorithm proposed in [25], where at each step, MEC server and subchannel selections are heuristically found by solving all possible resource allocation problems, and the algorithm ends when there is no feasible way to increase the objective value. Observe from Fig. 7b that at $f_0 = 3.5$ GHz the proposed algorithm generates the total computation overhead of 63.579, which is close to that of hJTORA with the gap of 8.86%. Fig. 7b also depicts that when the inter-cell interference is taken into consideration, offloading all computation tasks to the MEC servers is very inefficient. This is due to the fact that (i) the locations of users and MEC servers are randomly distributed in each simulation realization, so some users may have very bad connections to the MEC servers, and (ii) when the inter-cell interference exists and become severe, the offloading rates of offloading users are relatively low and the offloading time becomes much higher. In this scenario users, with the bad connections and severe interference, should locally handle their computations, while the other send requests to the MEC servers for computation offloading. As a result, a joint optimization of offloading decision, resource allocation, and interference management is highly needed to improve the network performance, which is clearly demonstrated by the comparison between our proposed algorithm and hJTORA with the local and offloading only schemes in Fig. 7.



(a) Percentage of offloading users.

(b) Total computation overhead.

Fig. 7: Performance of the proposed algorithm under different computation capabilities of MEC servers.

V. Conclusion and Future Work

In this paper, we have proposed an optimization problem for jointly determining the computation offloading decision and allocating the transmit power of users and computation resources at the MEC servers. Our proposed framework is different from existing ones in that 1) we have considered a HetNet with multiple MEC servers, where a user is possibly under the service coverage of multiple MEC servers and 2) we have proposed a decentralized computation offloading scheme based on the matching theory. The simulation results validated that the proposed algorithm can achieve better performances than three alternative frameworks. In addition, our proposed algorithm could perform close to that of the centralized exhaustive search with the small optimality gap.

A joint framework of resource allocation and server selection in collocation edge computing systems is currently under investigation of our ongoing work. Moreover, we will take into account the effects of computation offloading to the quality of service of macrocell users. Finally, we will consider of hierarchical MEC systems for differentiated applications of users where users with latency-sensitive applications offload their tasks to the first tier at small cells while users with latency-tolerant applications offload their tasks to the second MEC server tier at macrocells.

References

[1] 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. PP, no. 99, pp. 1–1, 2017.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[3] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[4] 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.

[5] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[6] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[7] Q.-V. Pham, L. B. Le, S.-H. Chung, and W.-J. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, 2018.

[8] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *CoRR*, vol. abs/1708.08810, 2017. [Online]. Available: http://arxiv.org/abs/1708.08810

[9] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

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

[11] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[12] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[13] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[14] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *CoRR*, vol. abs/1704.00163, 2017. [Online]. Available: http://arxiv.org/abs/1704.00163

[15] 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.

[16] T. LeAnh, N. H. Tran, W. Saad, L. Le, D. Niyato, T. Ho, and C. S. Hong, "Matching theory for distributed user association and resource allocation in cognitive femtocell network," *IEEE Trans. Veh. Technol.*, vol. PP, no. 99, pp. 1–1, 2017.

[17] Q.-V. Pham and W.-J. Hwang, "Energy-efficient power control in uplink spectrum-sharing heterogeneous networks," *International Journal of Communication Systems*, vol. 31, no. 14, p. e3717, July 2018.

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

[19] X. Ge, H. Cheng, M. Guizani, and T. Han, "5g wireless backhaul networks: challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6–11, Nov. 2014.

[20] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19 324–19 337, 2018.

[21] K. Sato and T. Fujii, "Radio environment aware computation offloading with multiple mobile edge computing servers," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Mar. 2017, pp. 1–5.

[22] M. Emara, M. C. Filippou, and D. Sabella, "MEC-aware Cell Association for 5G Heterogeneous Networks," *ArXiv e-prints*, Nov. 2017.

[23] S. Ranadheera, S. Maghsudi, and E. Hossain, "Computation offloading and activation of mobile edge computing servers: A minority game," *IEEE Wireless Communications Letters*, pp. 1–1, 2018.

[24] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, Sept. 2017.

[25] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *CoRR*, vol. abs/1705.00704, 2017. [Online]. Available: http://arxiv.org/abs/1705.00704

[26] N. Li, J. Martínez-Ortega, and G. Rubio, "Distributed joint offloading decision and resource allocation for multi-user mobile edge computing: A game theory approach," *CoRR*, vol. abs/1805.02182, 2018. [Online]. Available: http://arxiv.org/abs/1805.02182

[27] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[28] Z. Han, Y. Gu, and W. Saad, *Matching Theory for Wireless Networks*. Springer, 2017.

[29] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Commun. Mag.*, vol. 53, no. 5, pp. 52–59, May 2015.

[30] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 103–122, Nov. 2016.

[31] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, Aug. 2018.

[32] V. Chandrasekhar, J. G. Andrews, and A. Gatherer, "Femtocell networks: a survey," *IEEE Commun. Mag.*, vol. 46, no. 9, pp. 59–67, Sep. 2008.

[33] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[34] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.

[35] E. Yaacoub and Z. Dawy, "Proportional fair scheduling with probabilistic interference avoidance in the uplink of multicell ofdma systems," in *2010 IEEE Globecom Workshops*, Dec. 2010, pp. 1202–1206.

[36] S. Bayat, R. H. Y. Louie, Z. Han, B. Vucetic, and Y. Li, "Distributed user association and femtocell allocation in heterogeneous wireless networks," *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 3027–3043, Aug. 2014.

[37] D. Liu, L. Wang, Y. Chen, M. Elkashlan, K. K. Wong, R. Schober, and L. Hanzo, "User association in 5g networks: A survey and an outlook," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1018–1044, Second Quarter 2016.

[38] X. Li, W. Xu, Z. Feng, X. Lin, and J. Lin, "Matching-theory-based spectrum utilization in cognitive noma-ofdm systems," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2017, pp. 1–6.

[39] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[40] Q.-V. Pham and W.-J. Hwang, "Resource allocation for heterogeneous traffic in complex communication networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 10, pp. 959–963, Oct. 2016.

[41] ——, "Network utility maximization-based congestion control over wireless networks: A survey and potential directives," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1173–1200, Second Quarter 2017.

[42] D. F. Manlove, *Algorithmics of matching under preferences*. World Scientific, 2013.

[43] E. A. Jorswieck, "Stable matchings for resource allocation in wireless networks," in *2011 17th International Conference on Digital Signal Processing (DSP)*, Jul. 2011, pp. 1–8.

[44] Z. Zhou, K. Ota, M. Dong, and C. Xu, "Energy-efficient matching for resource allocation in d2d enabled cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5256–5268, June 2017.

[45] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *2012 IEEE Symposium on Computers and Communications (ISCC)*, Jul. 2012, pp. 000 059–000 066.