CCF Transactions on Networking manuscript No.
(will be inserted by the editor)

Communication and Computation Cooperation in Cloud Radio Access Network with Mobile Edge Computing

Tong Li · Kezhi Wang · Ke Xu · Kun Yang · Chathura Sarathchandra Magurawalage · Haiyang Wang

Received: June 4, 2018 / Accepted: October 26, 2018

Abstract Cloud radio access network (C-RAN) and mobile edge computing (MEC) have emerged as promising candidates for the next generation access network techniques. Unfortunately, although MEC tries to utilize the highly distributed computing resources in close proximity to user equipments (UE), C-RAN suggests to centralize the baseband processing units (BBU) deployed in radio access networks. To better understand and address such a conflict, this paper closely investigates the MEC task offloading control in C-RAN environments.

Most prior work handling offloading control falls in the general category of resource allocation optimization. However in this paper, we focus on the perspective of matching problem. Our model smartly captures the unique features in both MEC and C-RAN with respect to communication and computation efficiency constraints. We divide the cross-layer optimization into the following three stages: (1) matching between remote radio heads (RRH) and UEs, (2) matching between BBUs and UEs, and (3) matching between mobile clones (MC) and UEs. By applying the Gale-Shapley Matching Theory in the duplex matching framework, we propose a multi-stage heuristic to minimize the refusal rate for user's task offloading requests. Trace-based simulation confirms that our solution can successfully achieve near-optimal performance in such a hybrid deployment.

Keywords Computation Offloading · Cloud Radio Access Network · Mobile Edge Computing · Offloading Control

Tong Li
2012 Labs, Huawei Technologies, Shenzhen, China
E-mail: litong12@tsinghua.org.cn

Kezhi Wang
Department of Computer Science and Technology, Northumbria University, Newcastle upon Tyne, United Kingdom
E-mail: kezhi.wang@northumbria.ac.uk

Ke Xu
Department of Computer Science and Technology, Tsinghua University, Beijing, China
E-mail: xuke@tsinghua.edu.cn

Kun Yang
School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom
E-mail: kunyang@essex.ac.uk

Chathura Sarathchandra Magurawalage
InterDigital Europe, United Kingdom. Part of his work was done at the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom
E-mail: chathura.sarathchandra@gmail.com

Haiyang Wang
Department of Computer Science, University of Minnesota at Duluth, Duluth, United States
E-mail: haiyang@d.umn.edu

Corresponding author: Ke Xu

1 Introduction

User equipment (UE), such as smartphone and wearable device, is playing an important role in new application scenarios including virtual reality (VR), augmented reality (AR) and cloud gaming etc. While resource-constrained UEs (CPU, GPU, memory, storage capacity, and battery lifetime) have driven a dramatic surge in developing new paradigms to handle computation intensive tasks [Kumar(2010)] (for example, computation intensive applications requiring huge computing capacity are not suitable to run in mobile or portable devices). As shown in Figure 1, Mobile cloud computing (MCC) [Dinh(2013)] provides a solution where UEs offload computation to the remote resourceful cloud (*e.g.*, EC2 [Amazon(2018)]), thereby saving processing power and energy. However, the cloud in MCC scenarios is usually in a wide area network (WAN), and it is difficult to control delays and jitters at the WAN scale. Therefore offloading tasks to the public cloud may suffer from high latency via the Internet [Safaei(2005)]. For example, AR requires low latency in order to provide correct information according to user location and orientation, while offloading tasks to remote cloud may incur information distortion due to delayed data transmission. To accomplish this, mobile edge computing (MEC) [Hu(2015)] [Beck(2014)] is proposed where UEs offload computation intensive tasks to a computing

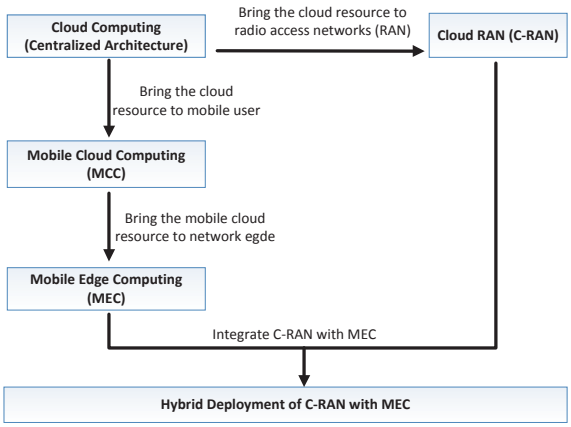

Fig. 1: Evolution progress

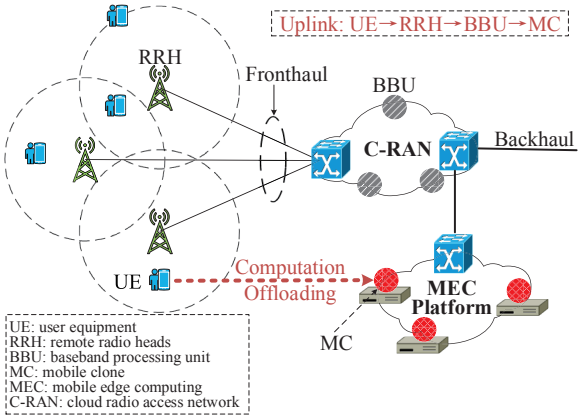

Fig. 2: Computation offloading architecture

resource-rich location, within radio access networks (RAN) and in close proximity to UEs.

On the other hand, task offloading generates data intensive workloads, which may become one of the main influential factors of the unprecedented mobile traffic growth [Ahmed(2015)]. It has been predicted that mobile traffic will increase exponentially to 100 times by the year 2020 [Andrews(2014)]. The dynamics of substantially increased data rates requires that cellular infrastructure must be flexible and reconfigurable, supporting simplified deployment and management of RANs. As conventional radio access network may incur high cost, latency and inefficient data exchange [Mobile(2011)], it lacks the efficiency to support centralized interference management and the flexibility to migrate services to network edges for computation intensive applications.

To ensure highly efficient network operation and flexible service delivery when handling mobile Internet traffic surging, cloud radio access network (C-RAN) [Mobile(2011)] [Wu(2012)] brings cloud computing technologies into mobile networks by centralizing baseband processing units (BBU) of RAN. It moves BBU from traditional base stations to the cloud and leaves remote radio heads (RRH) distributed geographically. RRHs are connected to BBU pool via high bandwidth and low-latency fronthaul. The BBU pool is realized by virtual machines (VM) in data centers, and the centralized processing enables BBU to be dynamically configured and shared [Tang(2015)]. In this case, with the transition from a conventional hardware based environment to a software based infrastructure, C-RAN can achieve flexible matching between RRHs and BBUs on demand.

To summarize, C-RAN has emerged as a replacement for the next generation access network. Prior work has proposed the hybrid deployment of C-RAN with MCC [Wang(2016b)] [Wang(2016a)], however, this integration still suffers the bottleneck introduced by MCC (*e.g.*, long latency). As mentioned above, MEC is a promising replacement of MCC in the latency-sensitive scenarios. It is worth mentioning that C-RAN uses centralized BBU to do baseband processing, while MEC handles distributed task offloading by shifting computation capacity from a public cloud to an edge cloud. Since MEC usually works with distributed base stations in conventional RANs, it is quite interesting to see if the MEC mobile offloading still works in C-RAN environments [Li(2017)].

Figure 2 shows the hybrid deployment of C-RAN with MEC[1] for computation offloading. Connected with geographically distributed RRHs and centralized BBUs, UEs get access to VMs, called mobile clones (MC), in a mobile cloud for computation offloading. For computation offloading requests, data is transmitted to MCs by base stations (composed of RRHs and BBUs) via uplinks. Once processed by an MC in the mobile cloud, the results will be returned to UEs via downlinks. As offloading control mainly focuses on uplink optimization, we calculate the completion time of task offloading as the sum of the data transmission latency via wireless communication and the task processing time on MCs.

Assume RRHs, BBUs and MCs are heterogeneous (*e.g.*, different loads and amount of resources), then the different matching among UEs, RRHs, BBUs and MCs results in different task offloading efficiencies. In particular, data transmission latency depends on the assignment of both RRHs and BBUs, and task processing time depends on the MC assignment. However, the UE interaction makes it challenging to directly assign a UE's most satisfied RRH, BBU or MC to them.

[1] The MEC platform is implemented by an edge cloud in close proximity to the BBU pool. The transmission latency from the MEC cloud to BBUs can be ignored compared to the latency from the MCC cloud to BBUs. Since the MCC cloud is usually in WANs far away from the radio access network, we call it MEC other than MCC in the hybrid deployment.

This interaction may affect the task offloading efficiency in two aspects: (1) the wireless transmission data rate will decrease with poor channel qualities between UEs and RRHs, (2) while the baseband processing speed of BBUs and task processing speed of MCs will be slowed down when overloaded. The former is called *communication efficiency*, and the latter is called *computation efficiency*.

For offloading control, we define *refusal ratio* as the proportion of offloading tasks that are not able to meet their deadlines. Then this paper is devoted to the efficient offloading control by addressing the assignment problem: how to assign RRHs, BBUs and MCs to UEs to minimize the refusal ratio among all the offloading requests? Different from the prior solutions of resource allocation [Wang(2016b)] [Wang(2016a)] [Sardellitti(2015)] [Tang(2015)] and admission control [Ha(2014a)] [Ha(2014b)], we focus on the matching problem. Moreover, we take into account the task offloading efficiency not only in wireless transmission but also in cloud computing, which is new and challenging in achieving efficient MEC task offloading control in C-RAN environments.

Motivated by these observations, we first formulate the joint assignment among UEs, RRHs, BBUs and MCs, which is unfortunately NP-Hard. By applying the duplex matching framework based on the classic Gale-Shapley Matching Theory, a multi-stage heuristic is finally given to minimize the refusal rate for UE's task offloading requests. Our major contributions are summarized as follows:

- We handle the offloading control with a new perspective that focuses on the joint RRH, BBU and MC matching problem, where a 0-1 programming model capturing the unique features in both MEC and C-RAN is proposed (Section 4).
- We divide the optimization problem into three stages including the UE-to-RRH stage, the UE-to-BBU stage, and the UE-to-MC stage, and a multi-stage heuristic for efficient offloading control is proposed(Section 6).
- We further conduct a trace-based evaluation to show that our solution can achieve the near-optimal performance for MEC task offloading control in C-RAN environments (Section 7).

2 Related Work

C-RAN is a cloud based, centralized, and collaborative radio access network, which was proposed by China mobile in 2009 and soon received a large amount of interests [Mobile(2011)]. Moreover, in 2015, another cloud-based technology, *i.e.*, mobile edge computing was launched by European Telecommunications Standards Institute (ETSI), which aims to bring cloud services closer to UEs [ETSI(2018)] such that users can enjoy high data rate, low latency and jitter services.

Cai *et al.* [Cai(2014)] enabled cloud services in the Internet, serving UEs by using a split-TCP proxy. However, the Internet may introduce large latency to the transmission, which may not be able to complete tasks within the required time limits. Wang *et al.* [Wang(2016b)] [Wang(2016a)] studied the joint resource allocation in C-RANs with MCC under the time constraints of the given tasks. Also, Sardellitti *et al.* [Sardellitti(2015)] studied joint optimization of radio and computational resources for MEC combined with cellular networks. Tang *et al.* [Tang(2015)] studied the cross-layer resource allocation with elastic service scaling in C-RANs. Nevertheless, all the above work [Wang(2016b)] [Wang(2016a)] [Sardellitti(2015)] [Tang(2015)] fell in the general category of resource allocation optimization, without considering the optimal matching between users (*e.g.*, UE), communication resource (*e.g.*, BBU) and computing resource (*e.g.*, MC).

Moreover, Ha *et al.* [Ha(2014a)] proposed cooperative transmission in C-RANs considering cloud processing constraints by allocating different BBUs and RRHs to different UEs. However, this paper did not consider the admission control scheme yet. Ha [Ha(2014b)] moved a step further by considering admission control in C-RANs under the fronthaul constraints. Both of the two papers only consider communication efficiency, other than considering cloud service computation efficiency as well.

Thus, to address the above challenges, we focus on the perspective of multi-stage RRH, BBU and MC assignments, and design a duplex matching framework based on the classic Gale-Shapley Matching Theory. To the best of our knowledge, no prior work has used the multi-stage matching algorithm to solve the offloading control problem in C-RANs with MEC, taking into account both communication efficiency and computation efficiency.

3 Offloading Control: Background and Framework

This section clarifies the computation offloading background and the offloading control framework in C-RANs with MEC.

3.1 Computation Offloading

Facing at the growing requirement for running resource-demanding applications, computation offloading expands the user base to the vast number of less powerful devices (*e.g.*, mobile phones and tablets). For example, the industrial pioneers such as Gaikai [Gaikai(2018)] and Onlive [Onlive(2018)] suggested a new generation of online gaming based on cloud computing platforms. For most 3D online games (*e.g.*, Battlefield 3, a highly popular first-person shooter game), the recommended system configuration is a quad-core CPU, 4 GB RAM, 20 GB storage space, and 1 GB video memory. However, the newest Samsung Galaxy or iPhone can only approach to the minimum system requirements, not to mention mobile devices whose hardware capability is limited. In this case, by utilizing the powerful and elastic service capacity offered by cloud computing, task offloading can meet the hardware/software requirements of user consoles. In particular, Gaikai and Onlive deploy cloud-based proxy to act as a game console/client and only stream game screen/interactions to end users.

Conventional cloud-based computation offloading offers great benefits for both users and service providers. However, offloading tasks to a public cloud may incur high latency due to multi-hop data transmission. For example, cloud gaming applications [Wang(2014)] firstly collect user actions, and then transmit them to the cloud proxy. During being processed in the cloud, the actions are rendered, encoded and compressed. Thereafter, the video (game scenes) will be streamed back to the player. All these serial operations must happen in milliseconds in order to ensure stable user's interactivity. The task offloading latency in MCC is thus essential even when the cloud capacity is not limited [Safaei(2005)].

To deal with the challenging issue of transmission delay, MEC, a new paradigm bringing the computation and storage to the close proximity of mobile subscribers, has attracted intensive attention from academia and industry [Hu(2015)] [Beck(2014)]. Figure 2 illustrates the overall architecture for task offloading in C-RAN with MEC. There are three basic components in the architecture: (1) Geographically distributed, RRHs are remote radio transceivers that bridge UEs and the operator radio control panel, performing lower layer analogue radio frequency (RF) functions. (2) Centralized in C-RANs, BBU is a unit for digital signal processing which can dynamically provision baseband processing for multiple distributed RRHs on demand. (3) MC is a VM (*e.g.*, an Android-x86 VM) deployed in a edge cloud near the BBU pool, hosting various mobile edge applications (*e.g.*, edge health care, smart tracking, automatic drive). For the scenarios of C-RANs with MEC, the MEC platform hosts computation and services at the edge of RANs, reducing network latency and bandwidth consumption for subscribers. Furthermore, network operators allow third-party partners to run the MEC platform, which will promote the rapid deployment of new applications and edge services to the mobile subscribers.

3.2 Computation and Communication Efficiency

Here we argue that not only communication efficiency but also computation efficiency should be considered in C-RANs with MEC scenarios, *i.e.*, there is interference among UEs both in wireless data transmission and cloud task processing. It is easy to understand that wireless channel quality will be influenced by user interaction. On the other hand, multiple tasks will compete for CPU time slices, which may lead to queueing delay. Moreover, based on the fact that the BBU processing during wireless communication can be regarded as computation intensive workload [Mobile(2011)], multiple UEs will also compete for the computing resource in the BBU. Table 1 summarizes some of the notations in our analysis.

Table 1: Summary of notations

Notation	Meaning
$\mathcal{U}, \mathcal{L}, \mathcal{B}, \mathcal{V}$	Set of the UEs, RRHs, BBUs and MCs, respectively ($u \in \mathcal{U}$, $l \in \mathcal{L}$, $b \in \mathcal{B}$, $v \in \mathcal{V}$);
d_u	Deadline of the offloading task for UE u;
c_u	Completion time of the offloading task for UE u;
D_u	Traffic size transmitted to the cloud for UE u;
F_u	Computing resource demand of the offloading task;
q	Number of UEs/Tasks on a node (BBU or MC);
f	Processing speed of a node (BBU or MC);
ρ_u	Transmission data rate (bit/second) for UE u;
α	Coefficient indicating the speed of full-loaded machine;
β	Coefficient controlling the skewness of the relationship between load and speed ($\beta \in (1, +\infty)$);
γ	VM service limitation;
θ_{ul}	MCS index value between UE u and RRH l.

Computation Efficiency. With regard to task processing, we use q_v to denote the number of tasks (load) being processed in MC v. Wang et. al [Wang(2014)] has conducted comprehensive experiments to demonstrate that the traffic load can significantly slow down processing speed of cloud VMs. Yet, such a problem is rarely seen on the non-virtualized local game consoles, or to a much lower degree. Since virtualization is applied in both C-RAN and MEC platforms, the Net Present

Value (NPV) function [Ross(1995)] applied by Wang can be borrowed to capture the relationship between processing speed and load (NPV has been widely used to quantify the relationship between cash and price/cost, which resembles our case when we try to purchase more computation resources to reduce the virtualization cost on VMs). We therefore calculate the task processing speed as follow:

$$f_{GOPS}^{v} = \frac{\beta^{\gamma - q_v}}{\alpha} \tag{1}$$

where f_{GOPS}^{v} refers to the computation frequency (CPU cycles per second) with the unit of *giga operations per second* (GOPS) in MC v. The parameter α indicates the speed when MC is fully loaded (reaching the VM service limitation γ ($\gamma > \max\{\lfloor \frac{n}{k} \rfloor, \lfloor \frac{n}{m} \rfloor\}$). Here the service limitation depends on the resource allocated to the VM, reflecting the budget of network operators. The parameter β controls the skewness of the relationship between load and speed where $\beta \in (1, +\infty)$. It is easy to see that different VMs may have different α, β and γ. For example in [Wang(2014)], the features of the EC2 large cloud instances is captured as follows: α is around 105, β is around 1.04, and γ represents the resource amount purchased from EC2. We have also investigated the parameters through our own testbed measurement, which is detailed in Section 7.1.

Communication Efficiency. On the other hand, the communication efficiency is influenced by multiple factors such as radio signal bandwidth and the modulation and coding scheme (MCS) index. Alyafawi *et al.* [Alyafawi(2015)] conducted a research to show that the decoding and encoding time for the LTE subframes grows with the increase of the MCS index. It is revealed that effective data rate over air interface (goodput) is mainly controlled by MCS. For heterogeneous UEs and RRHs in C-RANs, the MCS index varies from 0 to 31, deciding the number of bits per symbol and defining the amount of redundant information inserted into data stream [MCS(2018)]. Hence, the communication efficiency mainly depends on the MCS index between RRHs and UEs (in this paper, we do not consider user interference in wireless channels, whereas it can also be reflected by the MCS index). Based on the prior related work [Sigwele(2015)], we define the base station communication efficiency with the unit of *giga operations per bit*[2] (GOPB). We use $f_{GOPB} = g(\theta_{ul})$ to denote the communication efficiency, where $g(\theta)$ is defined as a function of the MCS index. θ_{ul} denotes the MCS index between UE u and RRH l.

Therefore, we derive the wireless transmission data rate (bit per second) as follow:

$$\rho_u = \frac{f_{GOPS}^{b}}{g(\theta_{ul})} \tag{2}$$

where f_{GOPS}^{b} refers to the computation frequency with the unit of GOPS in BBU b.

3.3 Offloading Control Framework

Real-time big data applications running on UEs have received considerable attention in the recent years [Ahmed(2015)]. These applications including automatic driving, health care, cloud gaming, mobile cloud governance etc. tend to offload their computation intensive functions to the cloud. Since it is not always smart to offload all tasks (small ones), UEs can make the decision according to the trade-offs between the overheads and benefits of offloading [Kumar(2010)]. Here we assume all UEs have offloading requests in our offloading control framework. For offloading tasks of hard real-time applications, their expected completion time varies due to the interdependence of tasks, *i.e.*, deadline of each task might be discrepant [Gardner(2015a)]. Meanwhile, the task will be invalid if it exceeds its deadline. Thus, offering an improved user experience and gaining higher operator profit mean maximizing the number of tasks that meet their deadlines across all offloading requests.

We illustrate the deadline-aware offloading control framework in Fig. 3. In terms of heterogeneous RRHs, BBUs and MCs, we consider the channel qualities between RRHs and UEs, the BBU load and the MC load as the inputs. At first, UE generates tasks with offloading requests, then the offloading control unit (*e.g.*, the mobile cloud controller) assigns RRHs, BBUs and MCs to UEs. The expected completion time of each offloading task is obtained as the output. By estimating whether a task may exceed its deadline, we decide to accept or reject UE's offloading request. Note that our objective is to maximize the number of tasks meeting their deadlines, the operator may gain a better profit while satisfying most subscribers.

4 Problem Formulation

In this section, we formulate the matching problem among UEs, RRHs, BBUs and MCs to achieve the optimal offloading control in C-RANs with MEC. Note that

[2] Based on the fact that poor channel quality aggravates packet loss and retransmission, resulting in more operations for baseband processing such as frequency domain (FD) processing and forward error correction (FEC), base station communication efficiency can be expressed in GOPB. Note that one operation here equals to π ($\pi \geq 1$) CPU cycles, the coefficient π is not explicitly reflected in Equation (2), but implicitly considered in $g(\theta)$.

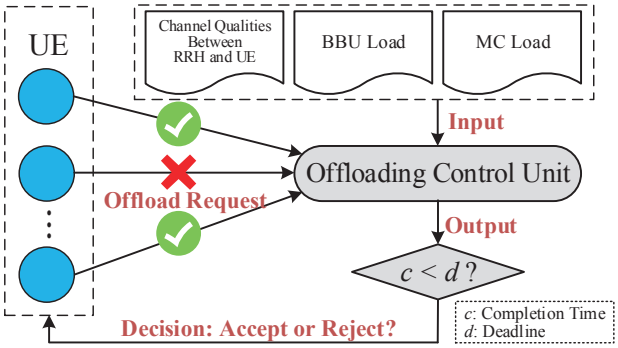

Fig. 3: Offloading control framework

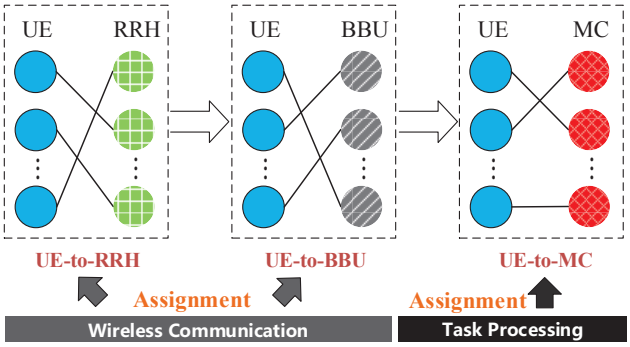

Fig. 4: Joint assignment among UEs, RRHs, BBUs and MCs

the problem we are solving can also be modeled into a non-matching problem, however, this paper proposes a different way, from the matching perspective, to achieve the maximum utility of both the operator and user.

As summarized in Table 1, $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, $\mathcal{L} = \{l_1, l_2, ..., l_o\}$, $\mathcal{B} = \{b_1, b_2, ..., b_k\}$ and $\mathcal{V} = \{v_1, v_2, ..., v_m\}$ denote the sets of UEs, RRHs, BBUs and MCs, respectively. n, o, k, and m denote the number of UEs, RRHs, BBUs and MCs, respectively. For a UE $u \in \mathcal{U}$ that requests task offloading, d_u refers to the deadline, and c_u refers to the completion time. According to Section 3.2, we consider the constraints of computation and communication efficiency. We model the task processing time and the wireless transmission latency, and then model the assignment optimization problem.

4.1 Task Processing Time

As mentioned above, computation efficiency depends on the loads in MCs. According to Equation (1), we therefore obtain the processing time of UE's offloading tasks as follow:

$$T_C(u, v) = \frac{F_u}{f^v_{GOPS}} = \frac{\alpha F_u}{\beta^{\gamma - q_v}} \tag{3}$$

where F_u refers to the computing resource of the offloading task, which is denoted by the number of CPU operations. F_u can be obtained by using the approaches provided in [Yang et al.(2013)Yang, Cao, Tang, Li, and Chan]. Note that although MCs do not usually have the same architecture as mobile devices in terms of hardware, we can calculate the cloud CPU cycles according to the mobile device ones [Kosta(2012)].

4.2 Wireless Transmission Latency

In C-RANs, user data is transmitted by wireless communication via base stations, in which the fibre links between RRHs and BBUs allow more flexibility in network planning and deployment. On the other hand, the BBU pool is also a cloud-based platform in C-RANs. Thus, the wireless transmission latency is related to both communication efficiency and computation efficiency. As mentioned above, we use different MCS indexes to estimate the communication efficiency. For the BBU baseband computation efficiency, we again use the NPV function to capture the relationship between baseband processing speed and the BBU load, *i.e.*, $f^b_{GOPS} = \frac{\beta^{\gamma - q_b}}{\alpha}$. Then based on Equation (2), we obtain the wireless transmission latency for UE u as follow:

$$T_N(u, l, b) = \frac{D_u}{\rho_u} = \frac{\alpha \cdot D_u \cdot g(\theta_{ul})}{\beta^{\gamma - q_b}} \tag{4}$$

where D_u refers to the traffic size to be transmitted to the cloud for UE u. D_u can also be obtained by using the approaches provided in [Yang et al.(2013)Yang, Cao, Tang, Li, and Chan].

4.3 Joint Assignment Optimization

Figure 4 illustrates the RRH, BBU and MC assignments. We define x_{uv}, z_{ul}, y_{ub} as the decision variables. In particular, $x_{uv}, z_{ul}, y_{ub} = 1$ if MC v, RRH l and BBU b are assigned to UE u, respectively, otherwise $x_{uv}, z_{ul}, y_{ub} = 0$. Since the offloading scheme depends on whether the task is able to meet its deadline, our objective becomes minimizing the refusal ratio for the UE's offloading requests, *i.e.*, maximizing the amount of UEs whose completion time is less than their deadlines. As mentioned before, we focus on the uplink completion time, which can be calculated by

$$c_u = T_C(u, v(u)) + T_N(u, l(u), b(u)) \tag{5}$$

where $v(u)$, $l(u)$, and $b(u)$ denote the MC, RRH and BBU assigned to UE u, respectively. Defining $\{x\}^+ =$

$\begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases}$, we therefore obtain the number of UEs that will miss their deadlines, *i.e.*, $Z = \sum_{u \in \mathcal{U}} \{c_u - d_u\}^+$ (refusal ratio is $\frac{Z}{n}$). Then the joint assignment optimization model is proposed as follows:

$$\min \quad \sum_{u \in \mathcal{U}} \{c_u - d_u\}^+ \tag{6}$$

$$\text{s.t.} \quad \sum_{v \in \mathcal{V}} x_{uv}, \sum_{b \in \mathcal{B}} y_{ub}, \sum_{l \in \mathcal{L}} z_{ul} = 1 \quad \forall u \in \mathcal{U} \tag{7}$$

$$\sum_{u \in \mathcal{U}} x_{uv} \leq \gamma_v - \gamma_v^0, \quad \forall v \in \mathcal{V} \tag{8}$$

$$\sum_{u \in \mathcal{U}} y_{ub} \leq \gamma_b - \gamma_b^0, \quad \forall b \in \mathcal{B} \tag{9}$$

$$x_{uv}, y_{ub}, z_{ul} = 0 \text{ or } 1, \quad \forall u \in \mathcal{U}, v \in \mathcal{V}, b \in \mathcal{B} \tag{10}$$

where the constraint (7) refers to that every UE only selects one MC, every UE only selects one BBU, and every UE only selects one RRH. Note that the load of MC v can be calculated as $q_v = \sum_{u \in \mathcal{U}} x_{uv} + \gamma_v^0$, and the load of BBU b can be calculated as $q_b = \sum_{u \in \mathcal{U}} y_{ub} + \gamma_b^0$, where γ^0 denotes the initial load. Then (8) and (9) refer to the service limitation constraints of MC and BBU, respectively.

Assuming the amount of computing resource is given, according to the Formulas (3)-(6) and (10), this matching problem can therefore be transformed into a 0-1 Multiple Knapsack problem with a non-linear objective function, which is known to be NP-hard [Li(2006)]. Thus we are devoted to seeking efficient heuristics towards the optimal solution, which will be detailed in the next sections.

5 Duplex Matching Framework

Our modeling focuses on the joint assignment optimization among UEs, RRHs, BBUs and MCs. By exhaustively searching all the possible combination of x_{uv}, y_{ub} and z_{ul}, the optimal solution can be achieved. However, the practical usefulness of this method is limited considering the real-time user demands. We thus propose a tri-level heuristic, which divides the optimization problem into three stages: the UE-to-RRH stage, the UE-to-BBU stage, and the UE-to-MC stage. As illustrated in Fig. 4, each stage is involved into the matching of a bipartite graph. The maximum matching of the bipartite graph can be achieved by other maximum flow algorithms (*e.g.*, the Edmonds-Karp algorithm [Edmonds(1972)]) or the Hungary algorithm [Kuhn(1955)]. Here we argue that none of these algorithms takes into account the interference between the

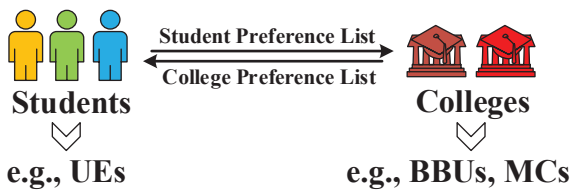

Fig. 5: Duplex matching in CAP

matching elements (*e.g.*, UEs), whereas both communication efficiency and computation efficiency result in dynamic utility of the matching elements. For example, for the two UEs (u_1, u_2) and two BBUs (b_1, b_2), the baseband processing speed matrix is set as $\begin{pmatrix} 10 & 15 \\ 20 & 30 \end{pmatrix}$ $(q = 1)$. When we assign the same BBU b_1 to both UEs, based on the fact that the BBU load affects computation efficiency, UE's baseband processing speed will decrease, i.e., the utility of UEs is lower than $\begin{pmatrix} 10 & 15 \end{pmatrix}$. However, the conventional bipartite graph algorithms failed to adapt to the utility dynamics of UEs. Under these circumstances, we aim to seek a novel duplex matching framework on account of the NP-Hardness of the joint assignment optimization model while adapting to the dynamics of computation and communication efficiency.

5.1 Deferred Acceptance (DA) Algorithm

First of all, we heuristically abstract a duplex matching framework which is inspired by the Gale-Shapley Matching Theory [Gale and Shapley(1962)], where Gale *et al.* discussed the real-life college admission problem (CAP). As shown in Fig. 5, regarding to the CAP, students are considered by a college which can admit a quota (denoted by φ). According to the applicant qualifications, the college decides which one to admit. Since students may apply multiple colleges according to their various preference lists, it is not generally satisfactory for the college to offer admissions to its φ best-qualified applicants. In this paper, UEs act the role of students, while RRHs, BBUs or MCs act as colleges.

The CAP can be solved by the classic Gale-Shapley Deferred Acceptance (DA) algorithm, which has already been proved to result in stable and the Pareto efficient match [Gale and Shapley(1962)]. DA is described as below.

Iteration i: All the non-admitted students apply to their i^{th} choice, and each college takes into account both the new applicants and the existing ones in its prospective admission list, assuming the total number of ap-

plicants is x. According to the college preference, every college puts top x ($x > \varphi$) students into its prospective admission list and rejects the other $x - \varphi$ applicants. If $x \leq \varphi$, put the x applicants into the prospective admission list.

Repeat *Iteration i* until every student is either in a prospective admission list of a certain college, or rejected by all colleges in its preference list.

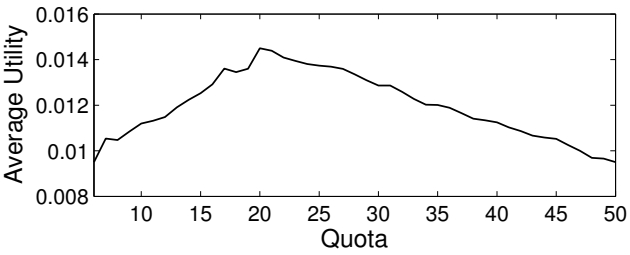

Fig. 6: Student utility vs. college quota

5.2 Applicability Consideration of DA

Inefficient Stability. It is worth mentioning that here the preference lists of both students and colleges are constant during each iteration, which means DA fails to consider the utility dynamics of UEs (as discussed above). Gale *et al.* demonstrated the stability and the Pareto efficiency of DA with regard to both colleges and students. Unfortunately, it is observed that this Pareto efficiency is inapplicable to our target duplex matching framework, in which the optimization objective is minimizing the refusal ratio. Since the refusal ratio is tightly coupled with the computation and communication latency, similar to the conventional bipartite graph algorithms, DA does not fit our problem without considering the dynamics of computation and communication efficiency. Therefore, despite the Pareto efficient, the stable match of DA is inefficient with regard to the duplex matching framework, which is called *inefficient stability*.

Quota Dependency. On the other hand, the quota of colleges is also fixed in DA. Here we argue that the match in DA has severe dependence on this quota. For instance, Fig. 6 shows the average utility varies with the quota φ. A random-generated instance with 10 colleges and 50 students are matched using DA. The student utility is quantified by Equation (1), where we set $\alpha = 105, \beta = 1.04$. For simplicity, we assume the student preference lists are the same and set $\varphi_i = \varphi = \gamma$ ($i = 1, 2, ..., 10$). We can find that the utility increases with the increase of small φ. However, it decreases with the increase of big φ. Intuitively enough, better colleges will always be popular among students, whereas full-loaded college results in utility decline according to Equation (1). It is observed that the performance of DA depends on the quota setting, called *quota dependency*.

5.3 Dynamic Duplex Matching Framework

To simultaneously mitigate inefficient stability and quota dependency, we finally propose the dynamic duplex matching framework, where we not only update the student preference list but also the college quota during each iteration.

In terms of inefficient stability, the student preference list is constant during all iterations, which ensures the matching stability of DA. Here we break the stability by taking into account the *expected utility*[3] that a student gets from a college. Since the expected utility depends on the interaction of all students, the student preference lists change dynamically during each iteration. This dynamics enables the students to close to the higher utility when selecting a college. The method to calculate the expected utility varies with the matching stage, which will be detailed in the next section.

On the other hand, to mitigate the challenge of quota dependency, we propose a technical solution that gives a dynamical quota to each college, i.e., gradually increase the quota of the *preferred college* in each iteration. Here we call it the preferred college whose applicant amount x meets $x > \varphi$.

It is easy to see that the dynamic duplex matching framework tends to result in unstable matches, as we do not care whether students have ever been rejected by colleges. In this case, by breaking the stability of DA we adapt our duplex matching framework to the dynamics of computation and communication efficiency. The worst case complexity of DA is $O(|S| * |C|)$, where $|S|$ and $|C|$ are the number of students and colleges, respectively [Iwama(2008)]. Since the quota gradually increases during each iteration, the worst case complexity of the dynamic duplex matching framework turns out to be $O(|S|^2 * |C|)$.

6 Multi-stage Duplex Matching

In this section, we aim to solve the proposed tri-level heuristic by applying the dynamic duplex matching framework. Table 2 summarizes some of the notations in our algorithm.

[3] Particularly, the expected utility can be the transmission latency for a UE assigning a BBU or the processing time for a UE assigning an MC.

Table 2: Summary of notations

Notation	Meaning
A_1	Matching from $\mathcal{U}$ to $\mathcal{L}$;
A_2	Matching from $\mathcal{U}$ to $\mathcal{B}$;
A_3	Matching from $\mathcal{U}$ to $\mathcal{V}$;
φ	Admission quota of a BBU or MC;
$\Delta\varphi$	Integer that denotes admission quota step size;
$\mathcal{Y}$	Prospective admission list of a BBU or MC;
$\mathcal{P}_u$	UE preference list for a UE to select BBUs ($\mathcal{P} \subseteq \mathcal{B}$);
$\mathcal{Q}_b$	BBU preference list for a BBU to select UEs ($\mathcal{Q} \subseteq \mathcal{U}$);
$\mathcal{P}'_u$	UE preference list for a UE to select MCs ($\mathcal{P}' \subseteq \mathcal{V}$);
$\mathcal{Q}'_v$	MC preference list for an MC to select UEs ($\mathcal{Q}' \subseteq \mathcal{U}$).

We define the assignment A_1, A_2 and A_3 as the matching from $\mathcal{U}$ to $\mathcal{L}$, $\mathcal{U}$ to $\mathcal{B}$ and $\mathcal{U}$ to $\mathcal{V}$, respectively. A_1 is the optimal assignment in the UE-to-RRH stage, while A_2 and A_3 are heuristic solutions obtained by applying the Matching Theory in the UE-to-BBU stage and UE-to-MC stage, respectively. It is worth mentioning that separately handling all stages is hard to close to the optimal assignment, and these three stages are correlative during multi-stage matching. In particular, A_2 is obtained according to A_1, and A_3 is obtained according to A_1 and A_2.

6.1 UE-to-RRH Stage

We assume that all the BBUs are the same and fully loaded, *i.e.*, $\alpha_b = \alpha$ and $q_b = \gamma_b$ ($b \in \mathcal{B}$). According to Equation (4), the expected transmission latency of UE u becomes $\alpha \cdot D_u \cdot g(\theta_{ul})$. In this case, each UE just selects the RRH with the minimal expected transmission latency. Thus, based on the MCS index θ_{ul}, we can get the optimal assignment A_1 between UEs and RRHs.

6.2 UE-to-BBU Stage

In this stage, different UEs have different deadlines d. Meanwhile, different BBUs have different loads q and service limitations γ. The BBU assignment problem to minimize the transmission latency can be transformed into a 0-1 Multiple Knapsack problem with the nonlinear objective function, which is known to be NP-Hard [Brucker(2009)]. Since it is hard to get the optimal assignment here, we apply the dynamic duplex matching framework. Particularly, we regard UEs as students and BBUs as colleges. Since both UEs and BBUs own diverse properties and their preference lists are variable within a large range, the challenge here is how to define the preference lists reasonably and efficiently [Li(2016)].

BBU Availability. We assume that all the MCs are the same and fully loaded, *i.e.*, $\alpha_v = \alpha$ and $q_v = r_v$ ($v \in \mathcal{V}$). According to Equation (3), the expected processing time of UE becomes αF_u. Since the BBU load affects transmission latency according to Equation (2), when we add a UE to the BBU, the performance of the UEs that are already assigned to this BBU will be affected. We therefore define that a BBU is available to a UE if those existing UEs can still meet their deadlines after the new UE is added in, *i.e.*, $T_N(u(b)) \leq d_{u(b)} - \alpha F_{u(b)}$, where $u(b)$ denotes the UE assigned to BBU b.

Algorithm 1 PreferenceListGeneration()

```
1: Get θ_ul according A_1;
2: for all u that u ∈ U do
3:     Get the set of the available BBU set B*_u;
4: end for;
5: for all u b that u ∈ U, b ∈ B do
6:     T_N(u,l,b) = GetExpectedTransLatency(q_u, θ_ul);
7:     Get P_u by sorting B*_u by ascending order of T_N(u,l,b);
8:     Get Q_b by sorting U by ascending order of d_u − αF_u;
9: end for;
10: return Q and P;
```

Preference List Generation. The preference for UE u to select the available BBUs results in preference list $\mathcal{P}_u$ ($\mathcal{P} \subseteq \mathcal{B}$). Also, every BBU owns a preference list $\mathcal{Q}_b$ ($\mathcal{Q} \subseteq \mathcal{U}$). As depicted in Algorithm 1, we calculate $\mathcal{P}_u$ and $\mathcal{Q}_b$ as follows. For UE u to select a BBU, $\mathcal{P}_u$ is obtained by sorting BBU set $\mathcal{B}$ by an ascending order of the expected transmission latency. To calculate the expected transmission latency for UE u assigning BBU b (Algorithm 1. *Step 6*), we add 1 to q_b and get θ_{ul} based on the assignment A_1 before calculating T_N according to Equation (2). Similarly, for BBU b to select a UE, $\mathcal{Q}_b$ is obtained by sorting $\mathcal{U}$ by an ascending order of $d_u - \alpha F_u$, where d_u denotes the deadline of UE u.

UE-to-BBU Duplex Matching. Defining $\mathcal{Y}_b$ as the prospective admission list of BBU b, and φ as the quota of a BBU. Based on the dynamic duplex matching framework, the UE-to-BBU duplex matching can be described as Algorithm 2, of which the iterations are outlined as below.

Iteration i: Every UE that is not assigned a BBU applies to its first choice of BBU in the preference list $\mathcal{P}$ (similar to the students applying to colleges in CAP), each BBU owns a set of x applicants (*Step* 5 of Algorithm 2). According to the preference list $\mathcal{Q}$, every BBU puts top φ ($x > \varphi$) UEs into its prospective admission list $\mathcal{Y}$ before rejecting the other $x - \varphi$ applicants (simi-

Algorithm 2 DuplexMatchingAlgorithm()

$i \leftarrow 1$, $\varphi_b \leftarrow 1$, $\mathcal{Y}_b \leftarrow \emptyset (b \in \mathcal{B})$;
while $(\bigcup_{b \in \mathcal{B}} \mathcal{Y}_b \neq \mathcal{U})$ **do**
  for all u b that $u \in (\mathcal{U} \setminus \cup_{b \in \mathcal{B}} \mathcal{Y}_b)$, $b \in \mathcal{B}$ **do**
    $(\mathcal{P}_u, \mathcal{Q}_b) = PreferenceListGeneration()$;
    $\mathcal{Y}_b = \mathcal{Y}_b \bigcup \{u | GetTopItem(\mathcal{P}_u) = b\}$;
  end for;
  for all b that $b \in \mathcal{B}$ **do**
    Sort $\mathcal{Y}_b$ according to UE ranking in $\mathcal{Q}_b$;
    if $GetElementCount(\mathcal{Y}_b) > \varphi_b$ **then**
      Define a temporary set $\mathcal{H}$ as the set of the bottom $(GetElementCount(\mathcal{Y}_b) - \varphi_b)$ UE(s) in $\mathcal{Y}_b$;
      $\mathcal{Y}_b = \mathcal{Y}_b \setminus \mathcal{H}$;
      $\varphi_b = \varphi_b + \triangle\varphi$;
    end if;
  end for;
  $i = i + 1$;
end while;
return A_2 according to $\mathcal{Y}_b$;

lar to the colleges rejecting students in CAP). Then add $\triangle\varphi$ $(1 \leq \triangle\varphi \leq n)$ to the φ of these preferred BBUs. If $x \leq \varphi$, put the x applicants into $\mathcal{Y}$.

Repeat *Iteration* i until every UE is in a prospective admission list of a certain BBU or all the BBUs are no longer available. Then the assignment A_2 is obtained according to $\mathcal{Y}$.

6.3 UE-to-MC Stage

After wireless transmission through base stations, offloading tasks are processed in the mobile cloud. The final UE-to-MC stage handles the MC assignment for offloading requests. Similar to the UE-to-BBU stage, it is also feasible to apply the dynamic duplex matching framework in this stage, however, to achieve better performance we have to modify some steps.

Firstly, the preference list for UE u to select the available MCs is denoted by $\mathcal{P}'_u$ $(\mathcal{P}' \subseteq \mathcal{V})$, which can be obtained by sorting $\mathcal{V}$ in an ascending order of the expected processing time. To calculate the expected processing time for UE u assigned MC v, we add 1 to q_v and calculate T_C according to Equation (3). Different from the UE-to-BBU stage, for MC v to select UEs, the preference list $\mathcal{Q}'_v$ $(\mathcal{Q}' \subseteq \mathcal{U})$ is obtained by sorting $\mathcal{U}$ in an ascending order of $d_u - T_N$, where T_N is calculated based on the assignment A_2. Secondly, the available MCs for a UE must meet the constraints of $T_C \leq d_u - T_N$ for all the UEs assigned to this MC. Note that during the UE-to-BBU stage, some of the offloading requests were probably not assigned to any BBU, due to task deadline constraints and BBU service limitations. As a result, in this stage we no longer assign MCs to the UEs that were not assigned a BBU.

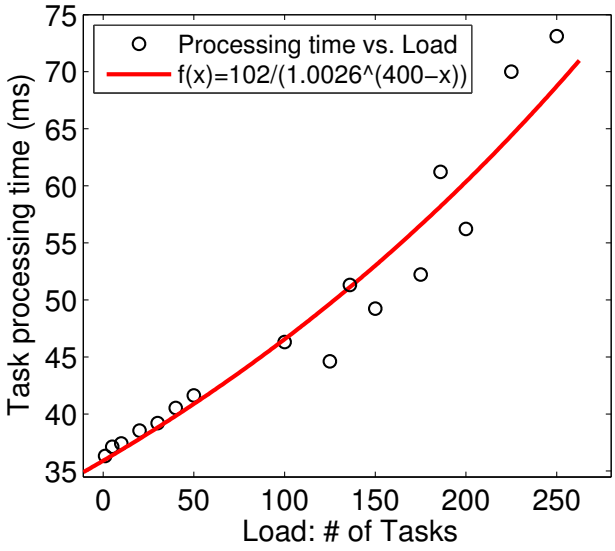

Fig. 7: Computation efficiency

By applying the modified Algorithms 1 and 2, we obtain the assignment A_3 through the duplex matching framework. The matchings between RRHs and BBUs as well as BBUs and MCs are also obtained by combining A_1, A_2 and A_3.

7 Performance Evaluation

In this section, based on the captured traces of an Openstack-enabled testbed, we start with the measurement of parameters in Equation (1), followed by deriving the function of MCS index θ_{ul}. Finally, we conduct the trace-based simulation to estimate our proposed multi-stage duplex matching algorithm.

7.1 Computation Efficiency Parameters

Firstly, we conduct measurements on our testbed to estimate the cloud computation efficiency. The standard CPU benchmark ($F = 1$ GHz) is created by the Rubis Task Generator [Rubis(2018)]. We adjust the number of tasks running on the Openstack instance and record the processing time of all tasks. Each case is tested 100 times to get the average processing time. As illustrated in Fig. 7, the task processing time increases with the load on the VM. In particular, a task processing with a standard CPU benchmark can be finished in 36.32 ms, whereas the completion time of all tasks increases to 73.11 ms when dealing with 250 concurrent task requests. Applying curve fitting, we therefore obtain the relationship between processing time and load,

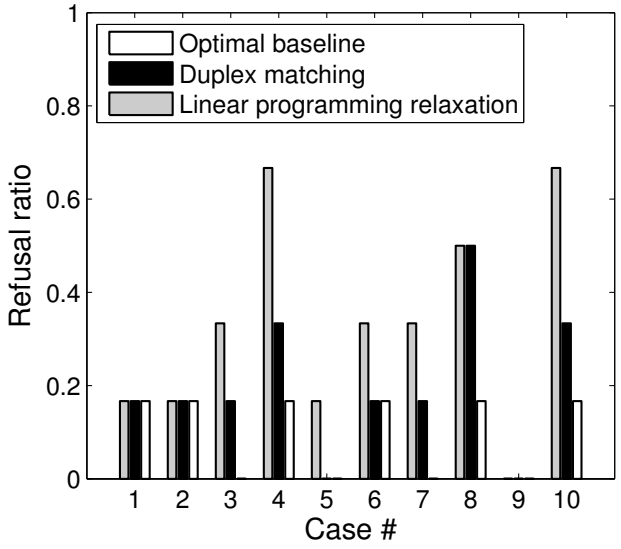

Fig. 8: Refusal ratio

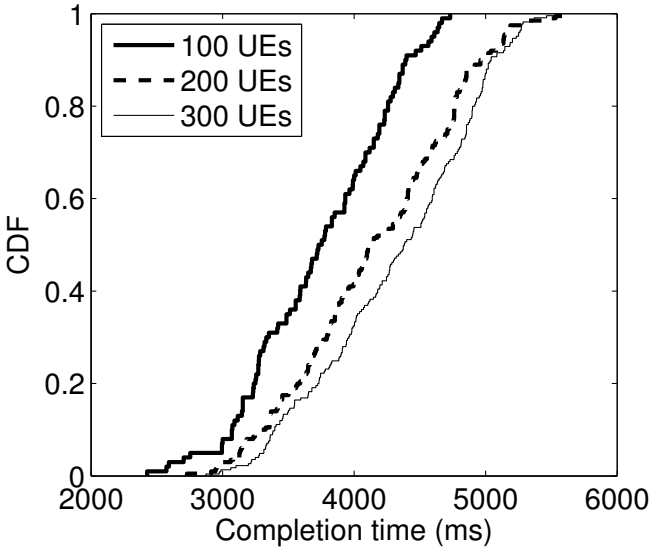

Fig. 10: Completion time

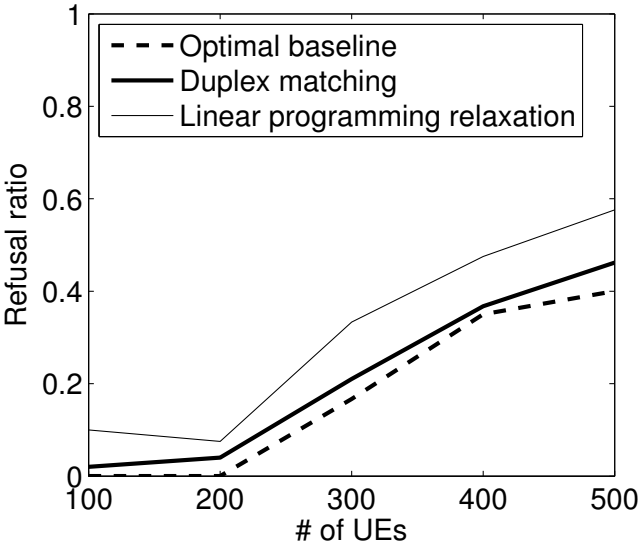

Fig. 9: Refusal ratio vs. # of UEs

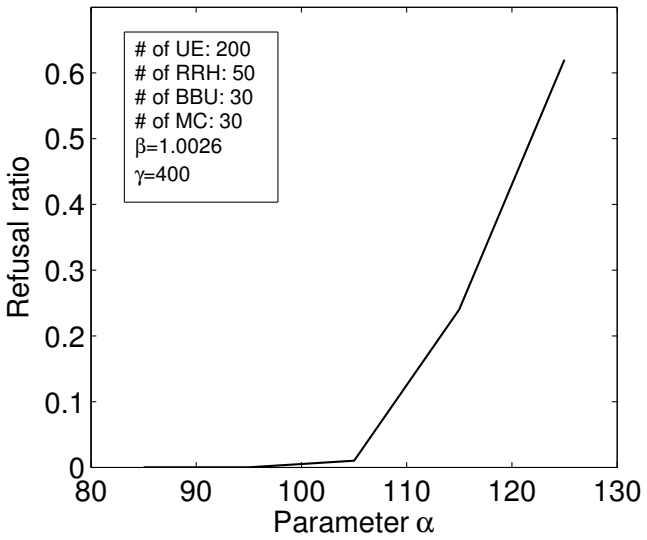

Fig. 11: Refusal ratio vs. α

i.e., $t(q) = \frac{102}{1.0026^{(400-q)}}$, where $t(q)$ denotes the task processing time and q denotes the load on the VM. With regard to the NPV function of Equation (1) in Section 3.2, we approximately calculate $\alpha = 102$, $\beta = 1.0026$, and $\gamma = 400$.

7.2 Communication Efficiency Parameters

Based on the analysis above, we calculate the baseband computation efficiency as $f_{GOPS} = 0.0276$ GOPS. According to the MCS index table [MCS(2018)], MCS index ($\theta \in [0 - 31]$) can be completely mapped into the combination value of $M \times C \times S$, *i.e.*, $M \times C \times S = \Phi(\theta)$, where M denotes the modulation type ($1, 2, 4, 6$, referring to BPSK, QPSK, 16-QAM, and 64-QAM, respectively), C denotes the coding rate (*e.g.*, $\frac{1}{2}$, $\frac{2}{3}$, $\frac{3}{4}$, and $\frac{5}{6}$), and S denotes the number of spatial streams (1 - 4). For example, $\theta = 0, 1, 15, 31$ refer to $M \times C \times S = 0.5, 1, 10, 20$, respectively. Note that the multiple MCS indexes may be mapped to the same $M \times C \times S$, for example, both of $\theta = 1$ and $\theta = 8$ refer to $M \times C \times S = 1$. [MCS(2018)] further illustrates that the data rate grows with the increase of $\Phi(\theta)$ ($\Phi(\theta) \in [0.5, 20]$). We therefore fit a function of the data rate for $\Phi(\theta)$, *i.e.*, $\rho = 1.768\Phi(\theta)$, where ρ denotes the data rate. Based on Equation (2), we obtain the function of MCS index,

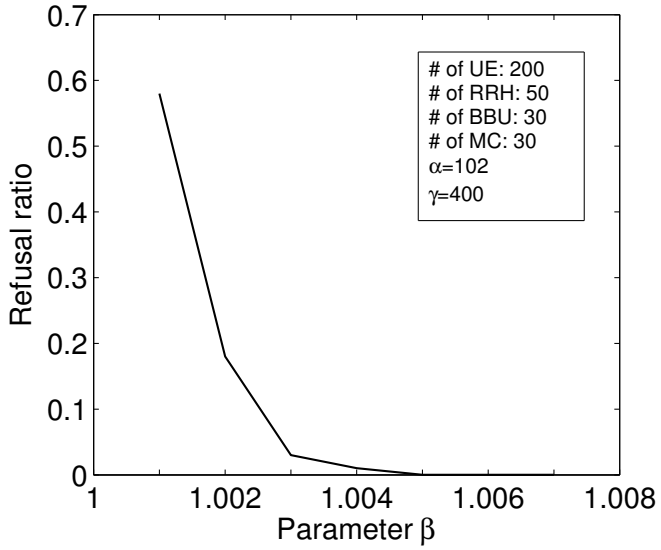

Fig. 12: Refusal ratio vs. β

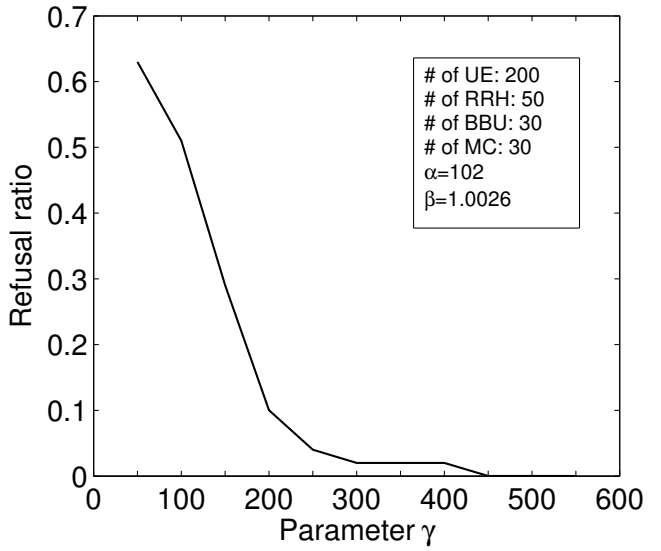

Fig. 13: Refusal ratio vs. γ

which is expressed by the following equation:

$$g(\theta) = \frac{W}{\Phi(\theta)} \tag{11}$$

where $W = 0.0156$ (W is related to radio signal bandwidth, which is set 20 MHz in this case), and the matching function of $\Phi(\theta)$ is obtained according to [MCS(2018)].

7.3 Algorithm Estimation

In this section, we conduct the matlab-based implementation to estimate the duplex matching framework, where trace-based offloading requests are fed to these

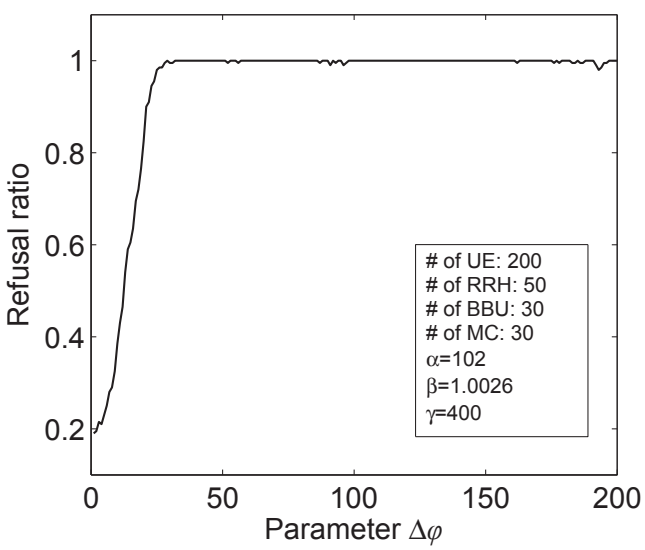

Fig. 14: Refusal ratio vs. $\Delta\varphi$

programs. In particular, task deadline, offloading traffic size and computation resource demand are randomly generated in a uniform distribution according to the prior works [Sigwele(2015)] [Gardner(2015b)] [Livelab(2018)], *i.e.*, we set $d_u \in [4000, 6000]$ (ms), $D_u \in [1, 100]$ (MB), $F_u \in [1, 20]$ (G Hz). We set parameters $\alpha = 102$, $\beta = 1.0026$, $\gamma = 400$, $\gamma^0 = 0$, $\Delta\varphi = 1$, $\theta_{ul} \in [0, 31]$, and $W = 0.0156$.

For optimality comparison, we summarize the algorithms as follows. *Optimal baseline* refers to the optimal solution obtained by brute-force searching. *Duplex matching* refers to our heuristic solution proposed in Section 6. *Linear programming relaxation* refers to the solution that converts the integer constraint (Formula (10)) into the continuous one, *i.e.*, $x_{uv}, y_{ub}, z_{ul} \in [0, 1], \forall u \in \mathcal{U}, v \in \mathcal{V}, b \in \mathcal{B}$. By solving the relaxed linear programming, we obtain the rounded decision variables as a feasible solution.

We start with a small offloading scenario with 6 offloading requests (each request is generated by an individual UE), 5 RRHs distributed geologically, 3 BBUs in the BBU pool, and 3 MCs in the MEC platform, *i.e.*, $n = 6$, $o = 3$, $k = 3$, and $m = 3$. Figure 8 presents the performance of our optimal assignment among UEs, RRHs, BBUs and MCs (we test 10 randomly generated cases). We can see that the multi-stage duplex matching reduces the refusal ratio compared to the linear programming relaxation solution (whose refusal ratio can be as high as 68%). Note that both the *Optimal baseline* and our solution can achieve 0 refusal ratio in Case 9. Eventually, this figure draws the conclusion that our approach can achieve the near-optimal performance in 90% of cases. The reason is our duplex matching frame-

work can adapt to the dynamics of computation and communication efficiency.

To avoid measurement bias, we also test the scenarios with a larger number of UEs, RRHs, BBUs and MCs, *i.e.*, $n \geq 100$, $o = 50$, $k = 30$, and $m = 30$. Figure 9 illustrates the case when the number of UEs varies from 100 to 500 (refusal ratio is the average value calculated by running the same case 100 times). We can see that even though refusal ratio increases with the number of UEs, our solution can always bound the optimal assignment. Figure 10 further explores the cumulative distribution function (CDF) of task completion time for the UEs admitted (UEs that can meet their deadline). It is easy to see that the completion time increases with the growing number of UEs, due to the increased load on both BBUs and MCs, which affects the computation efficiency. Moreover, 98% of the admitted tasks can be completed in 4500 ms when we have 100 UEs, and more than 91% of the tasks can be completed in 5000 ms with 200 or 300 UEs. This indicates that our solution can gain a small enough completion time (compared with deadline range) when minimizing the refusal ratio. Note that although the two cases (with 200 and 300 UEs) achieve similar completion time, the refusal ratio in 300 UEs is much higher than that in 200 UEs.

To better understand the computation efficiency in different types of VMs, we investigate the refusal ratio with different parameter inputs in Equation (1). Figure 11 presents the refusal ratio evolution when parameter α for the BBU or MC varies from 85 to 125. We can see that the refusal ratio increases with α. On the other hand, Figure 12 presents the refusal ratio evolution of the skewed relationship between processing speed and load, and Figure 13 presents the refusal ratio evolution of service limitation. Our figures reveal that a good VM should have a smaller α and larger β and γ. The minimum processing speed should be larger when the VM is fully loaded (smaller α). In other words, adding idle resources to the VM will significantly increase computation efficiency (larger β and γ).

8 Further Discussion

Our duplex matching framework is a bit different from the DA. In CAP, students are considered by a college which can admit a quota of only φ, which is fixed. According to the qualifications of applicants, the college decides which one to admit. However, in our solution we gradually increase the quota by $\Delta\varphi$ ($\Delta\varphi = 1$ by default) in each iteration. The intention to introduce quota is to bound the deadline of offloading requests (since we can hardly process all tasks on the same VM, this quota also contributes to load balance by avoiding overloading the most popular VM). Furthermore, we optimally assign BBUs or MCs to UEs by means of iteration.

Figure 14 further presents the case when $\Delta\varphi$ varies from 1 to 200. It is easy to see that the refusal ratio increases with incremental $\Delta\varphi$ when $\Delta\varphi < 30$. This reveals that the smaller $\Delta\varphi$ helps to achieve higher assignment performance with more iterations. On the other hand, when $\Delta\varphi \geq 30$, the algorithm performance turns out to be poor due to insufficient iterations.

9 Conclusion and Future Work

This paper focuses on the perspective of matching problem in the hybrid offloading architecture of the C-RAN with MEC. We design an efficient offloading control framework minimizing the refusal ratio of offloading requests. The joint assignment modeling shows the NP-Hardness of our problem, and a tri-level heuristic applying the duplex matching framework is therefore proposed, which divides the cross-layer optimization into three stages: the UE-to-RRH stage, the UE-to-BBU stage, and the UE-to-MC stage. Our evaluation shows that our solution can achieve the near-optimal performance.

Our ongoing work is to modify a simple framework version to work in real time and hopefully deploy a prototype system in C-RAN environments. Future work also includes the extension to environments with multi-resource management as well as further research on scenarios of mobility management [Beck(2014)].

References

[Ahmed(2015)] Ahmed E (2015) Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. JNCA 52:154–172

[Alyafawi(2015)] Alyafawi I (2015) Critical issues of centralized and cloudified lte-fdd radio access networks. In: Proc. of ICC, IEEE, pp 5523–5528

[Amazon(2018)] Amazon (2018) Amazon ec2. URL `http://aws.amazon.com/ec2/`

[Andrews(2014)] Andrews JG (2014) What will 5g be? JSAC 32(6):1065–1082

[Beck(2014)] Beck MT (2014) Mobile edge computing: A taxonomy. In: Proc. of AFIN, Citeseer

[Brucker(2009)] Brucker P (2009) Complexity results for scheduling problems. `http://www2.informatik.uni-osnabrueck.de/knust/class/`

[Cai(2014)] Cai Y (2014) Cloud radio access networks (c-ran) in mobile cloud computing systems. In: Proc. of INFOCOM Workshops, IEEE, pp 369–374

[Dinh(2013)] Dinh HT (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mob Comput 13(18):1587–1611

[Edmonds(1972)] Edmonds J (1972) Theoretical improvements in algorithmic efficiency for network flow problems. JACM 19(2):248–264

[ETSI(2018)] ETSI (2018) Etsi first meeting of new standardization group on mobile-edge computing. http://www.etsi.org/newsevents/

[Gaikai(2018)] Gaikai (2018) Gaikai. URL http://www.gaikai.com

[Gale and Shapley(1962)] Gale D, Shapley LS (1962) College admissions and the stability of marriage. American Mathematical Monthly pp 9–15

[Gardner(2015a)] Gardner K (2015a) Optimal scheduling for jobs with progressive deadlines. In: Proc. of INFOCOM, IEEE, pp 1113–1121

[Gardner(2015b)] Gardner K (2015b) Optimal scheduling for jobs with progressive deadlines. In: Proc. of INFOCOM, IEEE, pp 1113–1121

[Ha(2014a)] Ha VN (2014a) Cooperative transmission in cloud ran considering fronthaul capacity and cloud processing constraints. In: Proc. of WCNC, IEEE, pp 1862–1867

[Ha(2014b)] Ha VN (2014b) Joint coordinated beamforming and admission control for fronthaul constrained cloud-rans. In: Proc. of GLOBECOM, IEEE, pp 4054–4059

[Hu(2015)] Hu YC (2015) Mobile edge computing: A key technology towards 5g. ETSI White Paper 11

[Iwama(2008)] Iwama K (2008) A survey of the stable marriage problem and its variants. In: Proc. of ICKS, IEEE, pp 131–136

[Kosta(2012)] Kosta S (2012) Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Proc. of INFOCOM, IEEE, pp 945–953

[Kuhn(1955)] Kuhn HW (1955) The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2):83–97

[Kumar(2010)] Kumar K (2010) Cloud computing for mobile users: Can offloading computation save energy? Computer 43(4):51–56

[Li(2006)] Li D (2006) Nonlinear integer programming, vol 84. Springer Science & Business Media

[Li(2016)] Li T (2016) Towards minimal tardiness of data-intensive applications in heterogeneous networks. In: Proc. of ICCCN, IEEE, pp 1–9

[Li(2017)] Li T (2017) On efficient offloading control in cloud radio access network with mobile edge computing. In: Proc. of ICDCS, IEEE, pp 2258–2263

[Livelab(2018)] Livelab (2018) Measuring wireless networks and smartphone users in the field. URL http://livelab.recg.rice.edu/traces.html

[MCS(2018)] MCS (2018) Modulation and Coding Scheme (MCS) Index. URL http://mcsindex.com/

[Mobile(2011)] Mobile C (2011) C-ran: the road towards green ran. White Paper

[Onlive(2018)] Onlive (2018) Onlive. URL http://www.onlive.com

[Ross(1995)] Ross SA (1995) Uses, abuses, and alternatives to the net-present-value rule. Financial management 24(3):96–102

[Rubis(2018)] Rubis (2018) Rubis Task Generator. URL http://aiya.ms.mff.cuni.cz/been/documentation/javadoc/overview-summary.html

[Safaei(2005)] Safaei F (2005) Latency-driven distribution: infrastructure needs of participatory entertainment applications. Communications Magazine 43(5):106–112

[Sardellitti(2015)] Sardellitti S (2015) Joint optimization of radio and computational resources for multicell mobile-edge computing. TSIPN 1(2):89–103

[Sigwele(2015)] Sigwele T (2015) Evaluating energy-efficient cloud radio access networks for 5g. In: Proc. of DSDIS, IEEE, pp 362–367

[Tang(2015)] Tang J (2015) Cross-layer resource allocation with elastic service scaling in cloud radio access network. TWC 14(9):5068–5081

[Wang(2014)] Wang H (2014) On design and performance of cloud-based distributed interactive applications. In: Proc. of ICNP, IEEE, pp 37–46

[Wang(2016a)] Wang K (2016a) Cost-effective resource allocation in C-RAN with mobile cloud. In: Proc. of ICC, IEEE, pp 1–6

[Wang(2016b)] Wang X (2016b) Dynamic resource scheduling in cloud radio access network with mobile cloud computing. In: Proc. of IWQoS, IEEE, pp 1–6

[Wu(2012)] Wu J (2012) Green wireless communications: from concept to reality. Wireless Communications 19(4):4–5

[Yang et al.(2013)Yang, Cao, Tang, Li, and Chan] Yang L, Cao J, Tang S, Li T, Chan ATS (2013) A framework for partitioning and execution of data stream applications in mobile cloud computing. ACM Sigmetrics Performance Evaluation Review 40(4):23–32