Survivable Task Allocation in Cloud Radio Access Networks with Mobile Edge Computing

Song Yang, *Member, IEEE,* Nan He, Fan Li, *Member, IEEE,* Stojan Trajanovski, *Member, IEEE,* Xu Chen, *Member, IEEE,* Yu Wang, *Fellow, IEEE,* and Xiaoming Fu, *Senior Member, IEEE*

Abstract—**Cloud Radio Access Network (C-RAN) is a promising 5G network architecture by establishing Baseband Units (BBU) pools to perform baseband processing functionalities and deploying Remote Radio Heads (RRH) for wireless signal transmission and reception. Mobile Edge Computing (MEC) offers a way to shorten the service delay by building small-scale cloud infrastructures at the network edge. By co-locating BBU pool with edge cloud at the so-called BBU node, we can take full advantages of C-RAN and MEC for better spectrum utilization and delay-guaranteed services. In this paper, we first study how to allocate each user's task to BBU node and find the path from his/her accessing RRH node to the BBU node such that the maximum service delay among all the requests is minimized. We then consider this problem with survivability concerns, which is to use both primary and backup BBU nodes to issue the request such that the primary path and backup path are link-disjoint. We analyze the complexities of these two problems and prove they are NP-hard in general. Subsequently, we devise a randomized approximation algorithm and an efficient heuristic to solve the considered problems, respectively. The simulation results show that the proposed algorithms outperform two benchmark heuristics in terms of acceptance ratio and maximum service delay.**

Index Terms—**Survivability, Task Allocation, Delay, Cloud Radio Access Network, Mobile Edge Computing.**

I. Introduction

NOWADAYS, the increasing number of connected Internet of Things (IoT) devices such as smart phones has produced a large quantity of traffic, mostly coming from bandwidth-starving or time-critical applications of these devices such as Augmented Reality (AR) and online gaming. According to [1], the mobile data traffic will increase sevenfold between 2017 and 2022 globally. Based on the current cellular network and Radio Access Network technology, more Base Stations (BS) are needed in order to satisfy bandwidth requirement of ever-increasing mobile traffic with their respective applications and services. However, this accordingly incurs more capital expenditure (CAPEX) and operational expenditure (OPEX) for service providers. In this context, 5G mobile technology is currently underway to provide ubiquitous, massive capacity and high availability of services to more end users.

S. Yang, N. He and F. Li are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. Email: {S.Yang, henan, fli}@bit.edu.cn

S. Trajanovski is with Microsoft, W2 6BD London, UK. E-mail: strajan@microsoft.com

X. Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China. Email: chenxu35@mail.sysu.edu.cn.

Y. Wang is with Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania 19122, USA. Email: wangyu@temple.edu

X. Fu is with Institute of Computer Science, University of Göttingen, 37077 Göttingen, Germany. Email: Fu@cs.uni-goettingen.de

Cloud Radio Access Network (C-RAN) [2], [3] is a promising architecture for 5G mobile networks. It divides traditional Base Stations (BS) into 2 parts, namely (1) Remote Radio Head (RRH): its function is wireless signal transmission and reception and it interconnects mobile end users via radio access links, and (2) Baseband Unit (BBU): its responsibility is to perform baseband processing and computational functionalities. In this context, the fronthaul links are referred to connect RRHs and BBUs with high bandwidth and low latency, and backhaul links are used to connect mobile core network and BBUs. C-RAN deploys RRHs across all cell sites to receive tasks from end users for a full coverage, and places one or several BBU pools in the centralized locations to improve spectrum utilization efficiency and reduce CAPEX/OPEX. Therefore, C-RANs will become essential for 5G mobile networks for both cost reduction and performance improvement. According to [4], the worldwide market for C-RAN will reach 17.5 Billion dollars by 2023. For instance, Huawei has released OSN 810 product [5], which can be used for service transmission on metro C-RAN networks.

Moreover, although the current cloud computing paradigm [6] can provide a virtually unlimited capability service to end users, the requested tasks from end users may experience long-distance and congested transmission to the remote public cloud and incur higher service delay. Hence, cloud computing becomes unable to satisfy the ever-increasing stringent delay task demands [7], which remains a crucial drawback to tackle. The concept of Mobile Edge Computing (MEC) [8] has been proposed to bring the computing resources closer to end users by installing small resource-limited cloud infrastructure called edge cloud at the network edge. In this context, the end users' task workloads can be offloaded on edge clouds by achieving shorter service delay. MEC therefore embraces characteristics like location awareness, real-time task processing, agile network management and control.

However, in the current C-RAN architecture, it is suggested to use one centralized BBU pool to perform baseband processing and computational functionalities (process general tasks). On the one hand, since all the requested tasks which come from distributed geographically areas will be sent to one centralized BBU pool for processing, some of the tasks from remote area may experience longer routing delay and this will

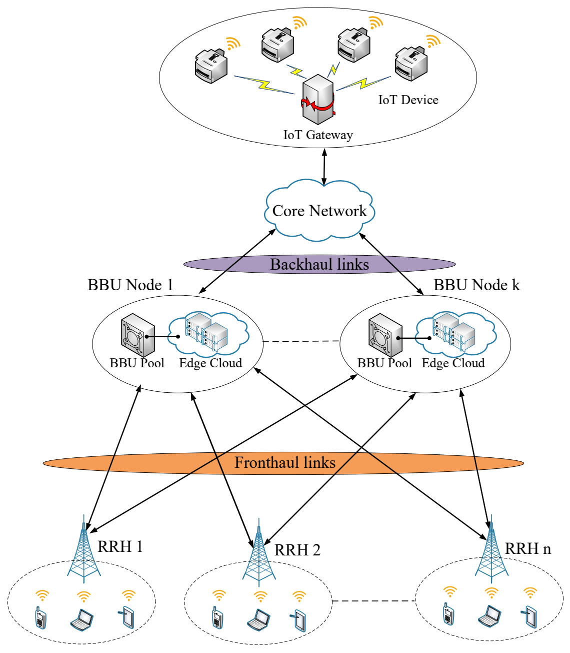


Fig. 1: An integrated system framework of C-RAN and MEC.

affect user's Quality of Service (QoS), especially for some delay-stringent task. Moreover, the failure of the centralized BBU node will trigger service breakdowns of all its associated tasks, which is not very robust. On the other hand, BBU pool is mainly used for baseband and some general task processing, but it cannot deal with different specific tasks. To tackle this issue, we propose to integrate mobile edge computing into C-RAN. More specifically, instead of applying one centralized "big" BBU pool, we could create several distributed "small" BBU pools for both delay and survivability concerns. Further, we deploy edge cloud consisting of a certain number of edge servers on each BBU pool to perform specific task functionalities. According to ETSI white paper [9], *collocating C-RAN and MEC can make the near term economics of deploying CRAN hubs based on generic processing components much more attractive, while positioning an Mobile Network Operator to support (and generate revenue from) some of the key 5G applications that it would not be able to support otherwise.*

Fig. 1 shows a possible framework by integrating MEC into C-RAN, where we call the node on which BBU pool and edge cloud are located and connected together as BBU node in this paper for brevity. Due to the limited resources (such as computing capability and battery) of IoT devices, the end users who use IoT devices have to send/offload their tasks to BBU node for processing, especially for complex and computation-intensive tasks. In this sense, each RRH node receives the task requests from its covering end users and sends the requests to the BBU node for signal processing and task processing. However, there are multiple BBU nodes that can process tasks, and there can be multiple paths from RRH node to the BBU node which traverse different backhaul links. Therefore, it is very important to solve the problem of how to allocate BBU node to process the task for each request and how to route each task traffic from its accessing RRH node to the corresponding BBU node such that the maximum service delay is minimized or guaranteed without affecting end users' quality of experience. Moreover, the BBU node or network link may fail due to man-made reasons or equipment malfunctioning, and this will produce service disruptions and degradation. In order to provide a reliable service [10], we need to first process the requested task on the primary BBU node and route the traffic via the primary path. Once a failure happens in the primary BBU node and/or primary path, we switch to process the task on the backup BBU node and route the traffic by the backup path which is link-disjoint with the primary path[1]. However, there is little existing work focusing on the aforementioned task allocation and routing problem in the integrated framework of C-RAN and MEC, not to mention about considering IoT service delay and survivability. In this paper, our main contributions are as follows:

- We study the Task Allocation and Survivable Task Allocation in C-RANs problems with MEC. We prove both problems are NP-hard in general and we identify under which conditions the Task Allocation in C-RANs problem is polynomial-time solvable.
- We present a randomized approximation algorithm and an efficient heuristic to solve the considered problems, respectively.
- We evaluate the proposed algorithms in terms of acceptance ratio, maximum service delay and running time.

As a result, the service provider can adopt our proposed algorithms to compute the solutions for IoT task requests in the integrated framework of C-RAN and MEC such that the maximum service delay is minimized with and without considering survivability issue.

The remainder of this paper is organized as follows. Section II presents the related work. Section III defines the Task Allocation and the Survivable Task Allocation in C-RANs problems with MEC and we show both problems are NP-hard. In Section IV, we propose an exact Integer Linear Programming (ILP) formulation to jointly solve the two problems. In Section V, we present an approximation algorithm and a heuristic for the considered problems, respectively. Section VI provides the simulation results and we conclude in Section VII.

II. Related Work

A survey about on throughput enhancement, interference management, energy efficiency, latency, security and system cost reduction of C-RAN based cellular networks can be found in [12].

A. Task/Resource allocation

Li *et al.* [13] target how to allocate Remote Radio Heads (RRH), BaseBand Unit (BBU) and Mobile Clones (MC) in C-RANs with MEC for each user requested task such that the number of requests which violate their respective deadlines is

[1]An overview of network survivability can be found in [11], where various link- or node-disjoint paths algorithms are proposed to find two link-disjoint paths.

minimized. They subsequently present a multi-stage matching theory-based heuristic to solve the problem. Different from [13], we assume there can be more than one BBU node that can process the users' requested tasks, and we also consider (survivable) routing issues. Pang *et al.* [14] study how to allocate tasks on different Fog-Radio Access Network (F-RAN[2]) nodes for multiple user requests such that the longest service delay among all the user requests is minimized. They present a polynomial-time dynamic programming-based algorithm for the special case of only one user request. They then devise an efficient heuristic for the general case of multiple user requests. Different from [14] in which the user's requested task can be divided into several blocks and conducted in multiple F-RAN nodes, we assume that the task can only be processed in one BBU node. Moreover, routing issue is also not considered in [14].

Wang *et al.* [16] propose an online approximation algorithm to jointly schedule network resources in C-RANs (switching on/off fronthaul links) and computation resources (dispatching requests to corresponding containers in different servers) in MEC to maximize profit of a service provider. Yu *et al.* [17] present an iterative algorithm to minimize the number of active BBU and RRH nodes such that each end user is accommodated by a RRH. Moreover, Yao and Ansari [18] consider the content placement problem in RRH and BBU nodes so as to minimize the network traffic costs based on data popularity. They devise an efficient heuristic for the problem. Wang *et al.* [19] formulate the joint energy minimization and resource allocation problem in C-RAN as a non-convex optimization by jointly considering the constraints of task executing time, transmitting power, computation capacity and fronthaul data rates. Subsequently, they reformulate this non-convex optimization into an equivalent convex problem based on weighted minimum mean square error (WMMSE), and derive an iterative algorithm. Liu *et al.* [20] study how to minimize the total energy consumption of C-RAN by jointly optimizing the cooperative beamforming, remote radio head (RRH) selection, and virtual baseband units (vBBUs) provisioning. To solve this problem, they first cluster RRHs into groups by using the hierarchical clustering analysis (HCA) algorithm and assign a vBBU to each RRH group for the baseband signal processing. After that, they determine the RRH selection by optimizing the cooperative beamforming. Hu *et al.* [21] present an approximation algorithm to solve the task allocation problem in C-RAN by randomized rounding without considering servicing delay and routing issues. Zhang *et al.* [22] propose an online framework for task offloading and radio resource allocation by considering both delay and energy consumption. Here, the task offloading problem refers to deciding the proportion of tasks to be executed on devices and edge servers. Zhang *et al.* [22] leverage Lyapunov technique to decompose this problem into four individual subproblems and apply convex decomposition methods and matching games to solve each subproblem accordingly. Chen *et al.* [23] first propose a deep-learning-based model to predict the traffic patterns and mobility trends. Subsequently, they propose a heuristic greedy algorithm to effectively find the robust approximation to the optimal schemes under resource constraints to solve the RRH-BBU mapping problem. Nevertheless, service delay and survivability issues are not considered in above literature.

[2]In F-RAN [15], one master F-RAN node is connected with multiple F-RAN nodes which receive user requests, and F-RAN node has the capability of processing tasks as well.

Moreover, [24], [25] deal with task offloading and scheduling problem for mobile edge computing, and [26], [27] tackle the microservice scheduling issue for mobile edge computing, but C-RAN is not considered. There are also some work dealing with task offloading [28], [29], [30], [31] in C-RAN with edge computing, which address how to divide tasks and decide which part of tasks are executed on devices, and which part of tasks are offloaded to edge servers. However, these work can only apply to the case when the task is divisible. In this paper, we deal with the indivisible task. For the divisible task, it is usually composed of several indivisible subtasks. In this sense, we could regard "subtask" as the "task" in the problem input and accordingly extend our work to solve the divisible tasks allocation problem in C-RAN with MEC.

B. BBU node placement and routing

The above literature assume that the BBU nodes are already placed in the network. [32], [33], [34], [35], [36], [37] jointly consider the BBU placement and (survivable) routing problem. The BBU node placement and routing problem of minimizing the overall network costs in Wavelength Division Multiplexing (WDM)-enabled fronthaul link are studied in [32], [33]. An exact Integer Linear Programming (ILP) is proposed in [32], [33] to solve this problem. Shehata *et al.* [34] further study how to place one primary and one backup BBU node for each RRH (request) and find respective wavelength path. An ILP is proposed for this problem. Khorsandi *et al.* [35] address the problem of placing primary and backup BBU nodes in order to cover all the RRHs. Khorsandi *et al.* [35] jointly consider the placement of a minimum number of BBU nodes and solve the routing and wavelength assignment problem for the fronthaul connections, ensuring that each RRH is connected to two different BBU nodes. They also devise a heuristic that tries to maximize the sharing of backup BBU ports while providing full coverage of all RRHs. Khorsandi and Raffaelli [36] additionally consider wavelength sharing and two enhanced placement strategies in their proposed heuristic. In addition, a facility location-based heuristic is presented in [37] in order to solve the survivable BBU location problem so as to minimize the overall costs.

Moreover, Mharsi and Hadji [38] present a Matroid-based algorithm to solve the assignment problem of RRHs to edge clouds. Younis *et al.* [39] consider the resource allocation problem as two subproblems, namely, (1) bandwidth power consumption problem and (2) BBU energy-aware resource allocation problem. To solve them, they respectively present a divide-and-conquer approach and a best fit decreasing method for bin-packing problem. However, the service delay is not considered in above work. Moreover, as we will show in this paper, even neglecting BBU placement problem, the respective task allocation and routing problem is NP-hard already and hence difficult to solve. This is the reason why only exact ILP and/or heuristic are proposed for jointly BBU placement and task allocation/routing problem in these work.

C. Cloud Computing

It is worthwhile to mention that our work is different from the VM placement problem and task scheduling problem in cloud computing, although they share some similarities. More specifically, in the Virtual Machine (VM) placement problem [40], [41], it is usually assumed that the "link" cost or delay metric connecting any two nodes in the network is known/given, i.e., the routing issue is not taken into account when dealing with VM placement. On the contrary, the addressed problem in this paper deals with both task allocation (decide which BBU node to perform task processing) and routing problem (find path from the accessing RRH node to the servicing BBU node) for each request such that the maximum servicing delay is minimized among all the requests. Furthermore, we also consider the task allocation problem in CRAN with survivability issue. In this sense, our problem definition additionally considers routing issue as well as survivability issue compared to the VM placement problem in cloud computing. Moreover, in the VM placement problem, the delay calculation only considers the routing delay between each VM pair, and the locations of the required VMs are not known. On the contrary, the servicing delay in our problem captures both routing delay and task processing delay, and the user's accessing RRH node is given to the problem. In terms of the task scheduling problem in cloud computing, it is usually assumed that a set of tasks have dynamically arrived with arrival time, task length/size, deadline, and finish time, and the problem is how to schedule these task requests without violating each task's deadline [42], [43], [44]. While the addressed problem in this paper is to decide which BBU node to process task requests (i.e., it is a task allocation problem but not task scheduling problem) and finding routing paths from accessing RRH node to the servicing BBU node, our work is therefore orthogonal to these works.

III. Problem Definition and Complexity Analysis

A network $G(\mathcal{N},\mathcal{L})$ is given, where $\mathcal{N}$ is a set of N nodes and $\mathcal{L}$ is a set of L links. Each link $l \in \mathcal{L}$ has a capacity of $c(l)$ and a delay value of $d(l)$. $\mathcal{N}_h \subseteq \mathcal{N}$ indicates a set of N_h RRH nodes which collect the task requests from end users. $\mathcal{N}_e \subseteq \mathcal{N}$ represents a set of N_e BBU nodes where BBU pool and edge cloud are placed, and each BBU node $n \in \mathcal{N}_e$ has a task processing capability of $\pi(n)$ (i.e., CPU cycles per second [45]). It is worthwhile to mention that one network node can be both the BBU node and RRH node, i.e., $\mathcal{N}_e \cap \mathcal{N}_h \neq \emptyset$. Moreover, apart from BBU node and RRH node, there also exist intermediate nodes (e.g., router nodes) in $\mathcal{N}$ to relay data, and different nodes are interconnected with links which depends on specific network topology. For each node pair (u,v) where $u,v \in \mathcal{N}$, we assume that the path set between them is given[3] and denoted by $P^{u,v}$ with a number of K paths. R represents a set of $|R|$ requests, and for each request $r(x,F,\delta) \in R$, x denotes the RRH node that the request accesses, F indicates required CPU cycles to complete the task, and δ stands for the transmitting rate. Without loss of generality, we assume that one end user only sends one task request in this paper. Moreover, we assume that each end user can transmit the task to the RRH node via an unique available channel. We therefore do not take the wireless transmission delay from users to RRH into account for brevity in this paper, since this part of value only depends on the channel bandwidth and task size [47] in this context and hence cannot be further optimized/minimized. The notations used in this paper are summarized in Table I.

TABLE I: Notations.

Notation	Description
$\mathcal{N},\mathcal{N}_h,\mathcal{N}_e$	The set of network nodes, RRH nodes and BBU nodes
$\mathcal{L},c(l),d(l)$	The set of L links, capacity and delay of link $l \in \mathcal{L}$
$\pi(n)$	Maximum processing capacity of BBU node $n \in \mathcal{N}_e$
$\mathcal{G}(\mathcal{N},\mathcal{L})$	A network with set of nodes $\mathcal{N}$ and links $\mathcal{L}$
$P^{u,v},K$	Path set between u and v, the number of paths
$T(p_k)$	Delay of path p_k
R	The set of requests. For each $r(x,F,\delta) \in R$, x indicates the RRH node it accesses, F denotes required CPU cycles to complete the task, δ represents the data transmitting rate
$H^{u,v}_{l,k}$	A given boolean array indicating whether link l is traversed by path p_k between u and v
D	A float variable meaning the maximum service delay
X^r_n, BX^r_n	A boolean variable. It is 1 (true) if r's requested task is processed by the primary/backup BBU node n; and 0 (false) otherwise
$Y^{r,k}_{u,v}, BY^{r,k}_{u,v}$	A boolean variable. It is 1 (true) if k-th primary/backup path between u and v is used for delivering the requested task of r; and 0 (false) otherwise.

The total service delay for serving a request $r(x,F,\delta)$ consists of traffic delivering delay and task processing delay. More specially, if r is processed by BBU node n, then the service delay Δ_r is equal to[4]

$$\Delta_r = T(x,n) + \Psi_n(F) \tag{1}$$

where $T(x,n)$ denotes the path delay from x to n if $n \neq r.x$ (otherwise $T(x,n)=0$), and $\Psi_n(F)$ indicates the processing delay on n for task F. According to [48], [45], $\Psi_n(F) = \frac{F}{\pi(n)}$.

Formally, the Task Allocation in C-RANs problem with MEC is defined as follows:

Definition 1: Given are a network $\mathcal{G}(\mathcal{N},\mathcal{L})$ and a set of requests R. The Task Allocation in C-RANs (TACR) problem with MEC is to allocate each user's requested task $r(x,F,\delta) \in R$ to corresponding BBU node and find the path from x to its servicing BBU node such that the maximum service delay among all the requests is minimized.

Accordingly, the Survivable Task Allocation in C-RANs problem with MEC is defined as follows:

Definition 2: Given are a network $\mathcal{G}(\mathcal{N},\mathcal{L})$ and a set of requests R. The Survivable Task Allocation in C-RANs (S-TACR) problem with MEC is to simultaneously allocate each user's task $r(x,F,\delta) \in R$ to two corresponding BBU nodes and find link-disjoint paths from x to its servicing BBU nodes

[3]According to [46], at most 6 paths in GÉANT network are enough for serving 11460 traffic matrices during the entire 4 month duration without violating Quality of Service (QoS). We therefore assume that a (small) set of paths is sufficient for calculating the optimal solution.

[4]In this paper, we assume that the task requests sent from end users' devices are to be processed only in BBU nodes (not executed locally on devices).

such that the maximum service delay among all the requests is minimized.

Theorem 1: The TACR problem is NP-hard when $|R| \geq 2$.

Proof 1: Let us first introduce the Min-Max routing problem [49]: Given is a network and each link is associated with a delay value, the Min-Max routing problem is to find a pair of link-disjoint paths from a source to a destination, such that the longer path delay is minimized.
In the TACR problem, suppose there is only one RRH node n_h and BBU node n_e such that $n_h \neq n_e$. Moreover, we assume $|R| = 2$ and each link in the network can only be used once for transporting either $r1$ or $r2$ because of the link capacity limit. Under this assumption, the TACR problem is equivalent to the Min-Max routing problem. Since the Min-Max routing problem is proved to be NP-hard according to [49], the TACR problem is NP-hard. The proof is therefore complete.

Remark 2: The TACR problem for $|R| = 1$ is polynomial-time solvable.

Proof 2: When $|R| = 1$, it indicates there is only one request (say $r(x, F, \delta)$). To solve the TACR problem, we apply the shortest path algorithm from $r.x$ to each node in $\mathcal{N}_e$ and store the result in Γ. Finally, we choose a feasible solution with the minimum service delay in Γ as the optimal solution.

Remark 3: The TACR problem for $N_e = 1$ is polynomial-time solvable when the link capacity is not considered.

Proof 3: When the link capacity is not considered, $\forall r \in R$ we can run shortest path from $r.x$ to $n \in \mathcal{N}_e$ and we can get the optimal solution (if there exists).

Theorem 4: The S-TACR problem is NP-hard, even when $|R| = 1$.

Proof 4: Suppose we do not consider link capacity and BBU node processing constraint, we first add a dummy node g_m and connect g_m with each node $n \in \mathcal{N}_e$ with link delay of $\Psi_n(F)$. Now, the S-TACR problem for $|R| = 1$ turns into the aforementioned NP-hard Min-Max routing problem [49], which is to find a pair of link-disjoint paths from $r.x$ to g_m, such that the longer path delay is minimized. The proof is therefore complete.

IV. Exact Integer Linear Programming

In this section, we present an exact Integer Linear Programming (ILP) model to jointly solve the TACR and the S-TACR problem. We first solve the S-TACR problem and start with some necessary notations and variables of the ILP.

ILP notations:

(u, v): Node pairs $u, v \in \mathcal{N}$ such that $u \neq v$.

$H_{l,k}^{u,v}$: A given boolean array. It is 1 (true) if link l is traversed by the path p_k between u and v, and 0 (false) otherwise.

$T(p_k)$: Delay of path p_k.

ILP variables:

D: A float variable indicating the maximum service delay value among all the requests.

X_n^r: A boolean variable. It is 1 (true) if the requested task of r is accommodated by primary BBU node n, and 0 (false) otherwise.

BX_n^r: A boolean variable. It is 1 (true) if the requested task of r is accommodated by backup BBU node n, and 0 (false) otherwise.

$Y_{u,v}^{r,k}$: A boolean variable. It is 1 (true) if k-th primary path between u and v is used for delivering the requested task of r, and 0 (false) otherwise.

$BY_{u,v}^{r,k}$: A boolean variable. It is 1 (true) if k-th backup path between u and v is used for delivering the requested task of r, and 0 (false) otherwise[5].

Objective:

$$\min \quad D \tag{2}$$

Delay Constraints:

$$\sum_{n \in \mathcal{N}_e} \sum_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \cdot (T(p_k) + \frac{r.F}{\pi(n)}) \leq D \quad \forall r \in R \tag{3}$$

$$\sum_{n \in \mathcal{N}_e} \sum_{p_k \in P^{r.x,n}} BY_{r.x,n}^{r,k} \cdot (T(p_k) + \frac{r.F}{\pi(n)}) \leq D \quad \forall r \in R \tag{4}$$

Placement Constraints:

$$\sum_{n \in \mathcal{N}_e} X_n^r = 1 \quad \forall r \in R \tag{5}$$

$$\sum_{n \in \mathcal{N}_e} BX_n^r = 1 \quad \forall r \in R \tag{6}$$

$$\sum_{u \in \mathcal{N}_h} \sum_{v \in \mathcal{N}_e} \sum_{p_k \in P^{u,v}} Y_{u,v}^{r,k} = 1 \quad \forall r \in R \tag{7}$$

$$\sum_{u \in \mathcal{N}_h} \sum_{v \in \mathcal{N}_e} \sum_{p_k \in P^{u,v}} BY_{u,v}^{r,k} = 1 \quad \forall r \in R \tag{8}$$

Path Selection Constraints:

$$X_n^r = \sum_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \quad \forall r \in R, n \in \mathcal{N}_e \tag{9}$$

$$BX_n^r = \sum_{p_k \in P^{r.x,n}} BY_{r.x,n}^{r,k} \quad \forall r \in R, n \in \mathcal{N}_e \tag{10}$$

Link Capacity Constraint:

$$\sum_{r \in R} \sum_{u \in \mathcal{N}_h} \sum_{v \in \mathcal{N}_e} \sum_{p_k \in P^{u,v}} r.\delta \cdot (Y_{u,v}^{r,k} + BY_{u,v}^{r,k}) \cdot H_{l,k}^{u,v} \leq c(l) \quad \forall l \in \mathcal{L} \tag{11}$$

BBU node processing capability Constraint:

$$\sum_{r \in R} (X_n^r + BX_n^r) \cdot r.F \leq \pi(n) \quad \forall n \in \mathcal{N}_e \tag{12}$$

Node-disjoint Constraint:

[5]For simplicity but without loss of generality, u always refers to the RRH node, and v always indicates the BBU node for $Y_{u,v}^{r,k}$ and $BY_{u,v}^{r,k}$.

$$X_n^r + BX_n^r \leq 1 \quad \forall n \in \mathcal{N}_e, r \in R \tag{13}$$

Link-disjoint Constraint:

$$\max_{u \in \mathcal{N}_h, v \in \mathcal{N}_e, p_k \in P^{u,v}} (Y_{u,v}^{r,k} \cdot H_{l,k}^{u,v} + BY_{u,v}^{r,k} \cdot H_{l,k}^{u,v}) \leq 1$$
$$\forall l \in \mathcal{L}, r \in R \tag{14}$$

Eq. (2) together with Eqs. (3) and (4) minimizes the maximum service delay value among all the requests. More specifically, for a certain request r in the primary task allocation of Eq. (3) (backup task allocation of Eq. (4) follows similarly), $\sum_{n \in \mathcal{N}_e} \sum_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \cdot T(p_k)$ calculates the path delay, and $\sum_{n \in \mathcal{N}_e} \sum_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \cdot \frac{r.F}{\pi(n)}$ calculates the task processing delay. Therefore, by imposing each request's service delay to be less than D for both primary and backup task allocation and minimizing D in the objective function, it is equivalently to ensure that the maximum service delay among all the requests is to be minimized. Eq. (5) (Eq. (6)) ensures that for each user's request, there must be one and only one primary (backup) BBU node to process its requested task. Eq. (7) (Eq. (8)) ensures that for each request, at most one primary (backup) path is used to transfer the task from its located RRH node to its hosted BBU node. The above placement constraints are set for variables X_n^r and $Y_{u,v}^{r,k}$. Eq. (9) (Eq. (10) follows similarly for BX_n^r and $BY_{u,v}^{r,k}$) establishes the equality relation between X_n^r and $Y_{u,v}^{r,k}$. More specifically, Eq. (9) indicates that when request r is accommodated on n (if $n \neq r.x$), then only one path between $r.x$ and n can be selected to use. Eq. (11) ensures that each link's capacity is not violated. Eq. (12) ensures that each BBU node's maximum processing capability is not exceeded. Eq. (13) ensures that the primary BBU node and backup BBU node for serving the same request cannot be the same node. Eq. (14) ensures that the primary path and backup path for delivering the same request traffic should be link-disjoint.

To solve the TACR problem exactly, we only need to keep the variables X_n^r and $Y_{u,v}^{r,k}$ in the above ILP and remove unnecessary variables and constraints containing BX_n^r and $BY_{u,v}^{r,k}$. The exact ILP for the TACR problem is shown in Eq. (15).

$$\min \quad D$$
$$s.t. \begin{cases} \sum\limits_{n \in \mathcal{N}_e} \sum\limits_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \cdot (T(p_k) + \frac{r.F}{\pi(n)}) \leq D \\ \forall r \in R \\ \sum\limits_{n \in \mathcal{N}_e} X_n^r = 1 \quad \forall r \in R \\ \sum\limits_{u \in \mathcal{N}_h} \sum\limits_{v \in \mathcal{N}_e} \sum\limits_{p_k \in P^{u,v}} Y_{u,v}^{r,k} = 1 \quad \forall r \in R \\ X_n^r = \sum\limits_{p_k \in P^{r.x,n}} Y_{r.x,n}^{r,k} \quad \forall r \in R, n \in \mathcal{N}_e \\ \sum\limits_{r \in R} \sum\limits_{u \in \mathcal{N}_h} \sum\limits_{v \in \mathcal{N}_e} \sum\limits_{p_k \in P^{u,v}} r.\delta \cdot Y_{u,v}^{r,k} \cdot H_{l,k}^{u,v} \leq c(l) \\ \forall l \in \mathcal{L} \\ \sum\limits_{r \in R} X_n^r \cdot r.F \leq \pi(n) \quad \forall n \in \mathcal{N}_e \end{cases} \tag{15}$$

V. Approximation Algorithm and Heuristic

Consider that the execution time of ILP solutions might increase with the complexity of the requests and the dimensions of C-RAN, in this section we first propose an approximation algorithm to solve the TACR problem. We subsequently devise an efficient heuristic to solve the S-TACR problem on the basis of the proposed approximation algorithm for the TACR problem.

A. Approximation algorithm for the TACR problem

The logic of the approximation algorithm is first to transform the original ILP to a Linear Programming (LP). By solving the LP, we subsequently achieve a solution via the randomized rounding method with a proved bounded approximation ratio. More specifically, we first obtain the relaxed LP for the TACR problem in Eq. (15) by setting X_n^r and $Y_{u,v}^{r,k}$ to be fractional variables ranging in $[0, 1]$, which provides a lower bound for the TACR problem. Subsequently, we present a Randomized Rounding Task Allocation (RRTA) algorithm in Algorithm 1 to solve the TACR problem. The rationale of RRTA in Algorithm 1 is that after solving linearized version in Eq. (15), we first get fractional solutions $\widetilde{Y_{u,v}^{r,k}}$ (and $\widetilde{X_n^r}$). Our aim is to derive an integer solution from $\widetilde{Y_{u,v}^{r,k}} > 0$ by randomized rounding. To that end, RRTA first uses $Ht[R]$ to store which BBU node to issue the requested task from $r \in R$, and uses $Rt[R][N][N]$ to store which path to route traffic of $r \in R$ from $u \in \mathcal{N}$ to $v \in \mathcal{N}$. After that, for each request r, RRTA finds the appropriate BBU node for processing its task request and the path to route the traffic from $r.x$ to its processing BBU node (if necessary). RRTA uses P to store the results where $\widetilde{Y_{r.x,n}^{r,k}} > 0$ in steps 6-7. If P is not empty, RRTA first tries to use $r.x$ to issue r if $\exists \widetilde{Y_{r.x,n}^{r,k}}$ such that $r.x == n$. If this step fails, RRTA randomly selects $\widetilde{Y_{r.x,n}^{r,k}} > 0$ from P and uses corresponding n to issue r. Accordingly, in step 14 $Ht[r] \leftarrow n$ means to record the BBU node for processing r and $Rt[r][r.x][n] \leftarrow k$ indicates to record path information. After that, in step 15 RRTA reduces the link capacity and node's processing capability accordingly. The above procedure continues until all the requests have been iterated.

The time complexity of RRTA can be analyzed like this: The time complexity of RRTA is dominated by the LP in Eq. (15). There exists an efficient polynomial-time algorithm to solve the LP with the current best worst-case complexity of $O([\frac{I^3}{\ln I}]\gamma)$ by an interior-point method according to [50], where I is the number of variables and γ is the bit size of the problem (related to the number of bits in its binary representation). There are in total $O(|R|KN^2 + |R|N + 1) = O(|R|KN^2)$ variables in Eq. (15), leading to a total time complexity of $O(\frac{\gamma |R|^3 K^3 N^6}{ln(|R|KN^2)})$ for RRTA.

B. Approximation Performance Analysis

The analysis leverages on the method of Upper Tail Chernoff bound [51] and Union Bound (Boole's inequality) [52, Chapter 4.7] — a technique often used in other works (see e.g., [53], [54], [55], [56]). For completeness, we first give a

Algorithm 1: RRTA $(\mathcal{G}(\mathcal{N},\mathcal{L}),R)$

```
1  Solve the LP in Eq. (15) where $X_n^r$ and $Y_{u,v}^{r,k}$ are set to
     be fractional variables to obtain $\widetilde{Y_{u,v}^{r,k}}$ and $\widetilde{X_n^r}$
2  $Ht[R] \leftarrow null$, $Rt[R][N][N] \leftarrow 0$
3  foreach request $r \in R$ do
4      $P \leftarrow \emptyset$
5      foreach $n \in \mathcal{N}_e$ do
6          if $\widetilde{Y_{r.x,n}^{r,k}} > 0$ then
7              P.Add($\widetilde{Y_{r.x,n}^{r,k}}$);
8      if $P \neq \emptyset$ then
9          if $\exists \widetilde{Y_{r.x,n}^{r,k}}$ such that $r.x == n$ then
10             $Ht[r] \leftarrow r.x$
11             Reduce the node's processing capability
12         else
13             Randomly select $\widetilde{Y_{r.x,n}^{r,k}} > 0$ from $P$
14             $Ht[r] \leftarrow n$, $Rt[r][r.x][n] \leftarrow k$
15             Reduce the link capacity and node's
                 processing capability
```

formal definition of the Upper Tail Chernoff bound and Union Bound inequality:

Theorem 5: [51] Let $x_1, x_2, \dots, x_n$ be n independent random variables, and $x_i \in [0,1]$ for $1 \le i \le n$. Denote $\mu = E\big[\sum_{i=1}^{n} x_i\big]$, then for an arbitrary positive ϵ we have:

$$\Pr\left[\sum_{i=1}^{n} x_i \ge (1+\epsilon)\mu\right] \le e^{\frac{-\epsilon^2\mu}{2+\epsilon}} \tag{16}$$

Theorem 6: Let $A_1, A_2, \dots, A_n$ be n events with happening probability $\Pr[A_1], \Pr[A_2], \dots, \Pr[A_n]$, then $\Pr[A_1 \cup A_2 \cup \dots \cup A_n] \le \sum_{i=1}^{n} \Pr[A_i]$.

Moreover, α is used to ensure that the following defined expected values are fractional number and defined as follows:

$$\alpha = \min\Big\{\min\{\frac{\min(c(l))}{r.\delta}\}, \min\{\frac{D_{\min}}{T(p_k)+\frac{r.F}{\pi(n)}}\}, \min\{\frac{\pi(n)}{r.F}\}\Big\}$$
$$\forall r \in R, l \in \mathcal{L}, n \in \mathcal{N}_e, p_k \in P^{u,v} : u \in \mathcal{N}_h, v \in \mathcal{N}_e \tag{17}$$

where $D_{\min}$ denotes the maximum service delay in the optimal solution. The proposed RRTA rounds the fractional $\widetilde{Y_{u,v}^{r,k}}$ solved from linearized version of Eq. (15) to integer value and then derive the task allocation and routing solution[6]. In the following, we will analyze the violating factor in terms of link capacity, maximum service delay and node processing capability of RRTA.

*1) **Link capacity violating factor**:*

Definition 3: For each request r and each link l, the traffic load z_l^r is defined as follows:

$$z_l^r = \begin{cases} r.\delta & \text{with prob.} \sum\limits_{u\in\mathcal{N}_h}\sum\limits_{v\in\mathcal{N}_e}\sum\limits_{p_k\in P^{u,v}} \widetilde{Y_{u,v}^{r,k}} \cdot H_{l,k}^{u,v} \\ 0 & \text{otherwise} \end{cases}$$

[6]According to Eq. (15), $X_n^r = \sum\limits_{p_k\in P^{n,r.x}} Y_{r.x,n}^{r,k}$, so the task allocation solution achieved by RRTA can also be regarded as rounded from X_n^r.

Since $z_l^{r_1}, z_l^{r_2}, \dots$ are mutually independent according to their definition, the expected load on link l is:

$$E\left[\sum_{r\in R} z_l^r\right] = \sum_{r\in R} E[z_l^r] \tag{18}$$
$$= \sum_{r\in R} r.\delta \sum_{u\in\mathcal{N}_h}\sum_{v\in\mathcal{N}_e}\sum_{p_k\in P^{u,v}} \widetilde{Y_{u,v,k}^{r,i,j}} \cdot H_{l,k}^{u,v} \le c(l)$$

According to the definition of α in Eq. (17), it holds that $0 \le \frac{z_l^r\cdot\alpha}{c(l)} \le 1$. Therefore, by dividing Eq. (18) with $\frac{c(l)}{\alpha}$ on both sides we have:

$$\mu_c = E\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{c(l)}\right] \le \alpha \tag{19}$$

Since $\frac{\alpha}{\mu_c} \ge 1$, we have:

$$\Pr\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{c(l)} \ge (1+\epsilon)\alpha\right] \le \Pr\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{c(l)} \ge (1+\epsilon)\alpha\right] \tag{20}$$

We cannot directly apply Theorem 5 for $\frac{z_l^r\cdot\alpha}{c(l)}$ with α, however the following holds:

$$\alpha = \frac{\alpha}{\mu_c}\mu_c = \frac{\alpha}{\mu_c}E\left[\sum_{r\in R}\frac{z_l^r\cdot\alpha}{c(l)}\right] = E\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{c(l)}\right] \tag{21}$$

By applying Theorem 5 for $\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{c(l)}$, based on Eq. (21):

$$\Pr\left[\sum_{r\in R}\frac{\alpha}{\mu_c}\frac{z_l^r\cdot\alpha}{c(l)} \ge (1+\epsilon)\alpha\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \tag{22}$$

where ϵ is an arbitrary positive value. Further, using inequalities from Eqs. (20) and (22) together and introducing Δ, we have:

$$\Pr\left[\sum_{r\in R}\frac{z_l^r}{c(l)} \ge (1+\epsilon)\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \le \frac{\Delta}{N^2} \tag{23}$$

where Δ is a network related variables and $\Delta \to 0$ when the network size grows. By solving Eq. (23), we have that

$$\epsilon \ge \frac{-\log\frac{\Delta}{N} + \sqrt{\log^2\frac{\Delta}{N^2} - 8\alpha\log\frac{\Delta}{N^2}}}{2\alpha} \tag{24}$$

Theorem 7: RRTA can achieve a link capacity violating factor of $\frac{4\log N}{\alpha} + 3$.

Proof: By setting $\Delta = \frac{1}{N^2}$, Eq. (23) becomes:

$$\Pr\left[\sum_{r\in R}\frac{z_l^r}{c(l)} \ge (1+\epsilon)\right] \le \frac{1}{N^4}, \quad \text{where } \epsilon \ge \frac{4\log N}{\alpha} + 2. \tag{25}$$

By using Union Bound inequality for all the links:

$$\Pr\left[\bigcup_{l\in\mathcal{L}}\sum_{r\in R}\frac{z_l^r}{c(l)} \ge (1+\epsilon)\right] \le \sum_{l\in\mathcal{L}}\Pr\left[\frac{z_l^r}{c(l)} \ge (1+\epsilon)\right]$$
$$\le N^2\cdot\frac{1}{N^4} = \frac{1}{N^2}, \quad \text{where } \epsilon \ge \frac{4\log N}{\alpha} + 2 \tag{26}$$

The last inequality holds since there are at most N^2 links in a network with N nodes. Finally, Eq. (26) indicates that the probability the expected load on any link violates $c(l)$ with a factor of $1+\epsilon = \frac{4\log N}{\alpha}+3$ approaches 0 when $N \to +\infty$. ■

*2) **Maximum service delay violating factor**:*

Theorem 8: RRTA can achieve a delay violating factor of $\frac{\log|R|+2\log N}{\alpha}+3$.

Proof 5: Similar to the proof of theorem 7, we have the following definition:

Definition 4: For each request $r \in R$, and each $n \in \mathcal{N}_e$:

$$w^r_{r.x,n} = \begin{cases} T(p_k) + \frac{r.F}{\pi(n)} \text{ with probability } \sum\limits_{p_k \in P^{r.x,n}} \widetilde{Y^{r,k}_{r.x,n}} \\ 0 \text{ otherwise} \end{cases}$$

Since $w^r_{r.x,n_1}, w^r_{r.x,n_2}, \ldots$ are mutually independent according to their definition, the expected delay for r follows:

$$E\left[\sum_{n\in\mathcal{N}_e} w^r_{r.x,n}\right] = \sum_{n\in\mathcal{N}_e} E[w^r_{r.x,n}]$$
$$= \sum_{n\in\mathcal{N}_e}\left(\sum_{p_k\in P^{r.x,n}} \widetilde{Y^{r,k}_{r.x,n}}\cdot(T(p_k)+\frac{r.F}{\pi(n)})\right) \le D \quad (27)$$

According to the definition of α in Eq. (17) it holds that $0 \le \frac{w^r_{r.x,n}\cdot\alpha}{D} \le 1$. By dividing Eq. (27) with $\frac{D}{\alpha}$ on both sides we have:

$$\mu_D = E\left[\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}\cdot\alpha}{D}\right] \le \alpha \quad (28)$$

In a similar way as proved from Eqs. (19) to (23), using $\frac{\alpha}{\mu_D} \ge 1$ (Eq. (28)) and applying Theorem 5 for $\frac{\alpha}{\mu_D}\frac{w^r_{r.x,n}\cdot\alpha}{D}$ whose expectation of their sum over $u, v \in \mathcal{N}$ is α, we arrive at:

$$\Pr\left[\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}\cdot\alpha}{D} \ge (1+\theta)\alpha\right] \le$$
$$\Pr\left[\sum_{n\in\mathcal{N}_e}\frac{\alpha}{\mu_D}\frac{w^r_{r.x,n}\cdot\alpha}{D} \ge (1+\theta)\alpha\right] \le e^{\frac{-\theta^2\alpha}{2+\theta}} \quad (29)$$

By letting $e^{\frac{-\theta^2\alpha}{2+\theta}}$ be less than $\frac{\Delta}{|R|}$, we have:

$$\Pr\left[\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}}{D} \ge (1+\theta)\right] \le e^{\frac{-\theta^2\alpha}{2+\theta}} \le \frac{\Delta}{|R|} \quad (30)$$

Eq. (30) is satisfied for:

$$\theta \ge \frac{-\log\frac{\Delta}{|R|}+\sqrt{\log^2\frac{\Delta}{|R|}-8\alpha\log\frac{\Delta}{|R|}}}{2\alpha} \quad (31)$$

By setting $\Delta = \frac{1}{N^2}$, Eq. (30) becomes:

$$\Pr\left[\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}}{D} \ge (1+\theta)\right] \le \frac{1}{|R|N^2} \quad (32)$$

Using Union Bound inequality for all the requests, we arrive at:

$$\Pr\left[\bigcup_{r\in R}\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}}{D} \ge (1+\theta)\right]$$
$$\le \sum_{r\in R}\Pr\left[\sum_{n\in\mathcal{N}_e}\frac{w^r_{r.x,n}}{D} \ge (1+\theta)\right]$$
$$\le |R|\cdot\frac{1}{|R|N^2} = \frac{1}{N^2}, \text{ where } \theta \ge \frac{\log|R|+2\log N}{\alpha}+2 \quad (33)$$

Eq. (33) indicates that for any request, the probability the expected delay violates the optimal maximum service delay with a factor of $1+\theta = \frac{\log|R|+2\log N}{\alpha}+3$ approaches 0 when $N \to +\infty$.

*3) **BBU node processing capability violating factor**:*

Theorem 9: The randomized approximation algorithm can achieve a node processing capability violating factor of $\frac{3\log N}{\alpha}+3$.

Proof 6:

Definition 5: For each request r and each BBU node n, the traffic load β^r_n is defined as follows:

$$\beta^r_n = \begin{cases} f \text{ with prob. } \widetilde{X^r_n} \quad \forall n \in \mathcal{N}_e, r \in R \\ 0 \text{ otherwise} \end{cases}$$

Since $\beta^{r_1}_n, \beta^{r_2}_n, \ldots$ are mutually independent according to their definition, the expected load on node n is:

$$E\left[\sum_{r\in R}\beta^r_n\right] = \sum_{r\in R}E[\beta^r_n] = \sum_{r\in R}\widetilde{X^r_n}\cdot f \le \pi(n)$$
$$\forall n \in \mathcal{N}_e \quad (34)$$

According to the definition of α in Eq. (17) and Eq. (34), it holds that $0 \le \frac{\beta^r_n\cdot\alpha}{\pi(n)} \le 1$. By dividing Eq. (34) with $\frac{\pi(n)}{\alpha}$ on both sides we have:

$$\mu_\pi = E\left[\sum_{r\in R}\frac{\beta^r_n\cdot\alpha}{\pi(n)}\right] \le \alpha \quad (35)$$

In a similar way as proved from Eqs. (19) to (23), using $\frac{\alpha}{\mu_\pi} \ge 1$ (Eq. (35)) and applying Theorem 5 for $\frac{\alpha}{\mu_\pi}\frac{\beta^r_n\cdot\alpha}{\pi(n)}$ whose expectation of their sum over $r \in R$ is α, we arrive at:

$$\Pr\left[\sum_{r\in R}\frac{\beta^r_n\cdot\alpha}{\pi(n)} \ge (1+\rho)\alpha\right] \le \quad (36)$$
$$\Pr\left[\sum_{r\in R}\frac{\alpha}{\mu_\pi}\frac{\beta^r_n\cdot\alpha}{\pi(n)} \ge (1+\rho)\alpha\right] \le e^{\frac{-\rho^2\alpha}{2+\rho}}$$

where ρ is an arbitrary positive value. Further, by letting $e^{\frac{-\rho^2\alpha}{2+\rho}}$ be less than $\frac{\Delta}{N}$, we arrive at:

$$\Pr\left[\sum_{r\in R}\frac{\beta^r_n}{\pi(n)} \ge (1+\rho)\right] \le e^{\frac{-\rho^2\alpha}{2+\rho}} \le \frac{\Delta}{N} \quad (37)$$

By solving Eq. (37), we have that

$$\rho \geq \frac{-\log\frac{\Delta}{N} + \sqrt{\log^2\frac{\Delta}{N^2} - 8\alpha\log\frac{\Delta}{N^2}}}{2\alpha} \tag{38}$$

By setting $\Delta = \frac{1}{N^2}$, Eq. (37) becomes
$\Pr\left[\sum_{r\in R}\frac{\beta_n^r\cdot\alpha}{\pi(n)} \geq (1+\rho)\alpha\right] \leq \frac{1}{N^3}$
As a result, for all the BBU nodes, we have:

$$\begin{aligned}&\Pr\left[\bigcup_{n\in\mathcal{N}_e}\sum_{r\in R}\frac{\beta_n^r}{\pi(n)} \geq (1+\rho)\right]\\ &\leq \sum_{n\in\mathcal{N}_e}\Pr\left[\sum_{r\in R}\frac{\beta_n^r}{\pi(n)} \geq (1+\rho)\right]\\ &\leq N_e\cdot\frac{1}{N^3} \leq \frac{1}{N^2},\ \text{where } \rho \geq \frac{3\log N}{\alpha}+2\end{aligned} \tag{39}$$

Therefore, Eq. (39) implies that for any BBU node in the network, the probability that its maximum processing capability is violated by a factor of $1+\rho = \frac{3\log N}{\alpha}+3$ will approach 0 when N grows $+\infty$.

Theorem 10: The randomized approximation algorithm can achieve a link capacity violating factor of $\frac{4\log N}{\alpha}+3$, maximum service delay violating factor of $\frac{\log|R|+2\log N}{\alpha}+3$, and node processing capability violating factor of $\frac{3\log N}{\alpha}+3$.

Proof: The proof follows from Theorems 7, 8 and 9. ■

Remark: Although theoretically RRTA has performance violating factors as analyzed in Theorem 10, we found that during extensive simulations in Section VI RRTA can achieve close-to-optimal performance, which indicates that it can hardly violate the threshold values. Furthermore, we could also accordingly adjust the "available" values of link capacity and node processing capability in advance, and then run RRVA. By doing this, we can guarantee there is no violation of the considered network metrics.

C. Heuristic for the S-TACR problem

In this subsection, we propose a heuristic called Survivable Sequential Task Allocation (SSTA) algorithm to solve the S-TACR problem. As shown in Algorithm 2, SSTA first calls RRTA to find the primary solution for all the requests. Subsequently, for each request r that has a feasible primary solution in Step 1, SSTA first removes the links and BBU node used by the primary solution in G. After that, SSTA runs the shortest path from $r.x$ to each remaining BBU node. SSTA then selects a BBU node as the backup solution with the minimum service delay. The link bandwidth and node capacity are accordingly reduced. Because the incurred RRTA is an approximation algorithm, it happens that some of the requests violate the requirements such as link capacity. For those requests (denoted by R'), Steps 8-11 apply Suurballe's algorithm [57] to find both primary and backup solution for each $r \in R'$. More specifically, in Step 9 SSTA first creates a dummy node and connects it with each BBU node. Subsequently, by applying Suurballe's algorithm from $r'.x$ to g_m, we can get the solution of two BBU nodes to process $r'.x$ and a pair of link-disjoint paths from $r'.x$ to two of BBU nodes. Finally, SSTA accordingly reduces the consumed link bandwidth and feasible processing capability in Step 11. Since SSTA calls for RRTA, the time complexity of SSTA is still dominated by the LP in Eq. (15). As a result, we can deduce that the time complexity of SSTA is $O(\frac{\gamma|R|^3K^3N^6}{ln(|R|KN^2)})$, which is the same with RRTA.

Algorithm 2: SSTA $(\mathcal{G}(\mathcal{N},\mathcal{L}), R)$

```
1  Call RRTA($\mathcal{G}, R$) in Algorithm 1 to achieve the
     primary solution.
2  Reduce the link capacity and node processing
     capability in $\mathcal{G}$ consumed by the primary solution.
3  foreach $r$ that has a feasible primary solution in Step
     1 do
4      $\mathcal{G}' \leftarrow \mathcal{G}$
5      Prune the links and BBU node in $\mathcal{G}'$ used by the
         primary solution of $r$ obtained in step 1.
6      Run shortest path from $r.x$ to each remaining BBU
         node and select one BBU node with the least
         total delay as the backup solution.
7      Reduce the link capacity and node processing
         capability in $\mathcal{G}$ consumed by the backup solution.
8  foreach $r'$ that violates the link or node capacity in
     Step 1 do
9      Create a dummy node $g_m$ to connect it with each
         BBU node with link weight as the processing
         delay.
10     Apply Suurballe's algorithm [57] from $r'.x$ to $g_m$
         to find link-disjoint paths.
11     Reduce the link capacity and node processing
         capability in $\mathcal{G}$ consumed by the solution.
```

Remark: In this paper, the goal of the TACR problem and S-TACR problem is to minimize the maximum service delay. However, it is worthwhile to mention that our proposed algorithms can also be adjusted to solve the problems for the requests with different delay requirements. For example, we can rewrite Eqs. (3) and (4) to ensure that each request's delay requirement is satisfied, and the approximation ratio proof goes similarly with Theorem 8. Moreover, our proposed algorithms (RRTA and SSTA) assume that a set of requests R is known/given and they calculate the solutions for these requests. This can be understood as an offline scenario. More than that, RRTA and SSTA can also deal with the requests which arrive in an online fashion. In the online scenario, the requests are assumed to arrive one by one dynamically, and RRTA and/or SSTA can take each arriving request as the input and return the task allocation and routing solution. That is to say, our tackled problem in this paper is general and our proposed algorithms can deal with both the online and offline scenarios.

VI. Simulations

A. Simulation Setup

We use the tool GT-ITM [58] to generate two MEC-enabled CRAN networks with the number of nodes 25 and 40, respectively. More specifically, we let (and round up) that

25% of the nodes are BBU nodes and these BBU nodes have at least 2 hops (a path with at least two links to connect them) from each other representing for different coverage regions. We also set (and round up) 70% of the nodes to be RRH nodes that can collect task requests from end users (one node can be both BBU node and RRH node). Moreover, network parameters, computing capabilities and traffic requests are randomly generated in a uniform distribution which is consistent with the existing work [13], [36], [59]. In both networks, the maximum processing capabilities of BBU nodes range from 800 to 1000. The link capacity is randomly distributed in $[0.5, 1]$ *Gb/s* and link delay takes values in $[10, 50]$ *ms*. We first randomly generate $|R| = 100, 200, 300, 400, 500$ requests for the TACR problem, and these requests are uniformly distributed in all the RRH nodes. For each request $r(x, F, \delta) \in R$, F is between 5 and 20 and δ is between 20 *Mb/s* and 50 *Mb/s*.

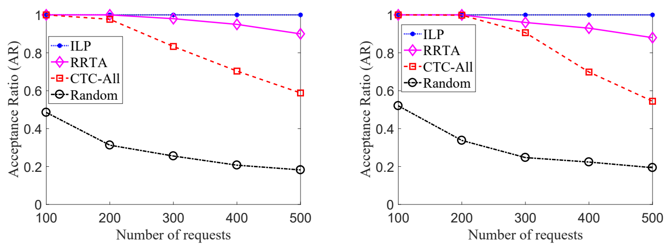
To solve the TACR problem, we compare the exact ILP in Eq. (15) and our proposed approximation algorithm RRTA with cooperative task computing one-for-all (CTC-All) which is derived from [14] and Random algorithm as follows:

1) CTC-All: We first create a dummy node g and connect it with each node $n \in \mathcal{N}_e$ with link weight $\frac{d(gn)}{\lambda(gn)} \cdot \tau$, where $d(gn)$ denotes the delay of link (g, n), $\lambda(gn)$ indicates the residual available processing capability, and τ represents a penalty factor or coefficient. For the others links $l \in \mathcal{L}$ in the network, its weight is assigned as $\frac{d(l)}{\eta(l)}$, where $\eta(l)$ stands for the free link capacity. By assigning the above link weights, CTC-All tries to reduce maximum service delay through distributing the requests throughout the network by avoiding bottleneck. Subsequently, for each request $r(x, F, \delta)$, CTC-All finds the shortest path from x to g without violating the link capacity and BBU node processing capability. This procedure continues until all the requests in R have been iterated.
2) Random Algorithm: For each request $r(x, F, \delta)$, Random algorithm keeps randomly selecting one BBU node $n \in \mathcal{N}_e$ until n has sufficient residual processing capability (in at most W rounds). If this step succeeds, Random algorithm randomly selects one path from x to n. This procedure continues until all the requests in R have been iterated.

To solve the S-TACR problem, the routine of CTC-All and Random algorithm is like this: First run CTC-All or Random algorithm in the network to obtain the primary solution. Prune all the links and BBU node used by the primary solution and applies CTC-All or Random algorithm again to achieve the backup solution.

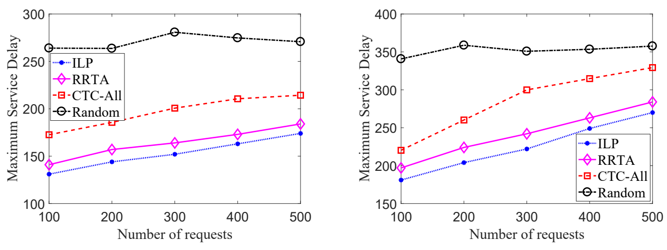
The number of paths K between each node pair is set to be 10 for all the algorithms. We let $\tau = 1$ for CTC-All and $W = 20$ for Random algorithm. The simulations are run on a high-performance desktop PC with 3.4 GHz and 16 GB memory. We use IBM ILOG CPLEX 12.8 to implement ILP, IBM ILOG CPLEX 12.8 (CPLEX Callable Library interface) together with C# to implement RRTA, and C# to implement all the heuristics.

B. Simulation Results for the TACR problem



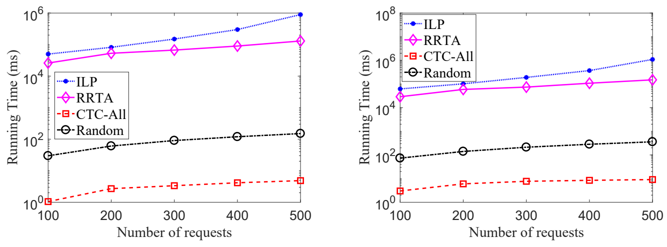
(a) 25-node network (b) 40-node network

Fig. 2: Acceptance Ratio for the TACR problem.



(a) 25-node network (b) 40-node network

Fig. 3: Maximum Service Delay for the TACR problem.



(a) 25-node network (b) 40-node network

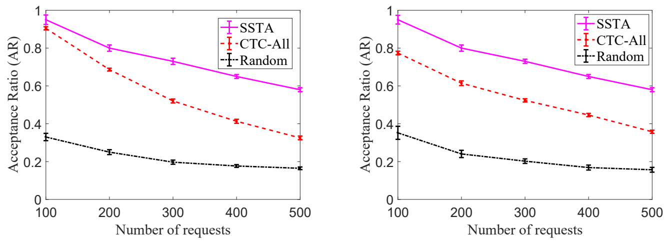
Fig. 4: Running Time for the TACR problem.

We first compare the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted/accommodated requests divided by the total number of requests. From Fig. 2 we can see that ILP can always accept all the requests and our approximation algorithm can accept almost all the requests. CTC-All algorithm also performs well in accepting more requests, which may benefit from its link weight allocation to evenly distribute task requests across the network, but at the expense of higher maximum service delay as we will show later in Fig. 3. The Random algorithm has the worst performance because of its randomness, and it can only accept around 20% of the total requests when $|R| = 500$.

Next, we evaluate the maximum service delay of all the algorithms in Fig. 3. We find that ILP can always have the lowest maximum service delay value, and the approximation algorithm achieves a very close performance with the ILP. Random algorithm performs poorly by having the highest maximum service delay value. Except for the Random algorithm, the maximum service delay value increases with $|R|$ increasing. The reason is that with more requests to serve, the available capacities of critical links are decreased rapidly, so longer delay path has to be used to route the request.

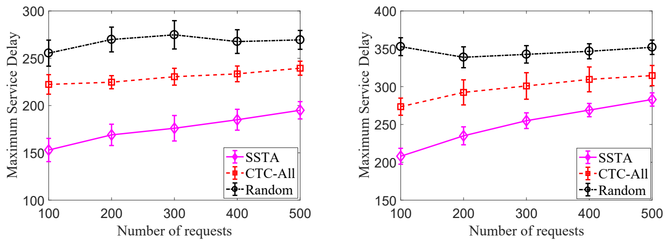
Finally, Fig. 4 presents the total running time of all the algorithms (in log scale). We observe that the CTC-All has the lowest running time, and Random algorithm consumes a bit higher running time than the CTC-All because of its iteration rounds W. The running time of RRTA is less than the value of ILP, but it is higher than the values of benchmark heuristics. Nevertheless, it is still acceptable since RRTA achieves better performance in terms of AR (Fig. 2) and maximum service delay (Fig. 3). In practice, in many cases the traffic requests can be known or predicted [60] in advance, therefore the service provider can apply the proposed approximation algorithm to calculate the solutions in an offline fashion (in acceptably short time as implied in Fig. 4) for the incoming traffic requests. Another observation is that when $|R| = 1$, we found our proposed approximation algorithm can return the solution in less than 1 second (We omit the figure for brevity). In practice, in many cases the traffic requests can be known or predicted [60] in advance, therefore the service provider can apply the proposed approximation/heuristic algorithm to calculate the solutions in an offline fashion (in acceptably short time as implied in Fig. 4) for the (delay-sensitive) incoming traffic requests. In addition, our proposed approximation algorithm can also be applied only periodically or as a complementary solution for existing methods.

C. Simulation Results for the S-TACR problem



(a) 25-node network (b) 40-node network

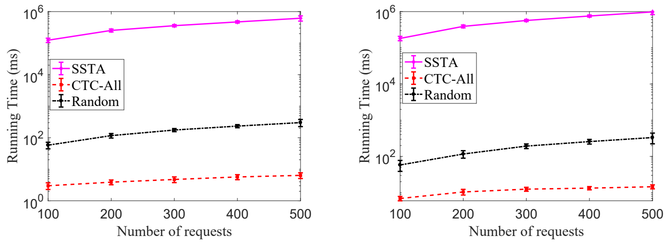
Fig. 5: Acceptance Ratio for the S-TACR problem (95 percent confidence interval).



(a) 25-node network (b) 40-node network

Fig. 6: Maximum Service Delay for the S-TACR problem (95 percent confidence interval).

In this subsection, we show the performance of the algorithms to solve the S-TACR problem. In particular, we found that the ILP in Eqs. (2)-(14) for the S-TACR problem cannot



(a) 25-node network (b) 40-node network

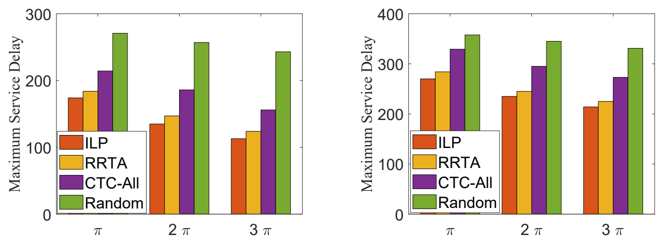
Fig. 7: Running Time for the S-TACR problem (95 percent confidence interval).

return a feasible solution even if it is running for more than one day. Besides, even though we set the variables to be fractional variables in the ILP of Eqs. (2)-(14), we still cannot get the feasible lower bound solution for more than one day's running. This indicates that with more variables and constraints in Eqs. (2)-(14), we cannot achieve a feasible exact solution or even lower bound in a reasonable time, and heuristics with faster running time are therefore needed. Due to the lack of the exact solution, we randomly generate 50 sets of different traffic requests and show the simulation results of the heuristics with the confidence interval of 95%.

Fig. 5 shows the AR value returned by all the heuristics. Similar to Fig. 2, SSTA achieves the highest AR, followed by CTC-All, and Random algorithm has the worst performance. Fig. 6 depicts the maximum service delay returned by all the heuristics. We see that SSTA always obtains lowest maximum service delay value among all the heuristics, and Random algorithm behaves worst. Moreover, although CTC-All has better performance in AR by leveraging its link weight allocation as shown in Fig. 5, it also leads to a higher maximum service delay as shown in Fig. 6. Finally, Fig. 7 shows the total running time of all the heuristics. Similar to Fig. 4, CTC-All and Random algorithm have much quicker running time than SSTA.

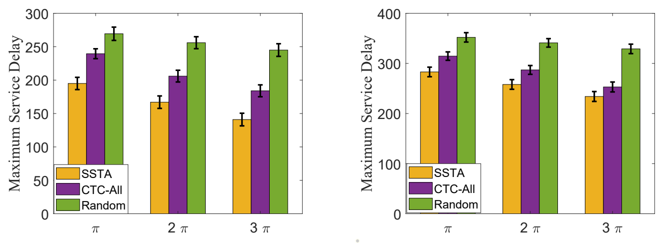
D. Impacts of node processing capacity and link capacity

In this subsection, we investigate the impact of processing capacity $\pi(n)$ and link capacity $c(l)$ on the maximum service delay. To that end, we respectively increase the value of $\pi(n)$ to $2\pi(n)$ and $3\pi(n)$ for each node $n \in \mathcal{N}_e$, and increase the value of $c(l)$ to $2c(l)$ and $3c(l)$ for each link $l \in \mathcal{L}$. Accordingly, the network throughput has also been increased with the increase of link capacity. Fig. 8 and Fig. 9 illustrate the maximum service delay value returned by all the algorithms with different π value for the TACR and S-TACR problem. We can see that when π increases, the maximum service values decrease for all the algorithms. The reason is that according to Eq. (1), the maximum service delay consists of both routing delay and processing delay. Since the processing delay is inversely proportional to processing capacity, an increase of processing capacity leads to a shorter processing delay and hence a shorter maximum service delay. With all the other simulation parameters unchanged, Fig. 10 and Fig. 11 depict the maximum service delay value with different value of link capacities. We see that with link capacity increasing, the maximum service delay decreases accordingly. This can be explained that since routing delay is one part of the service delay according to Eq. (1), when link capacity is enlarged, more requests can be routed by shorter delayed paths without violating the link capacity. Hence, the overall maximum service delay is decreased by increasing the link capacity. In all, we conclude that both node processing capacity and link capacity can affect the maximum service delay, which also validates Eq. (1).



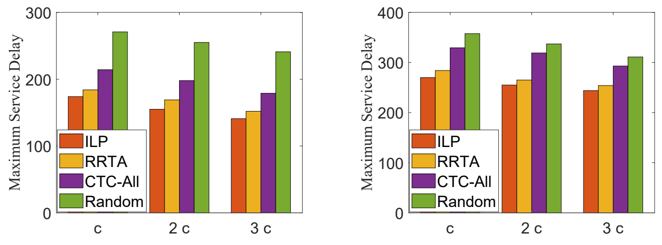
(a) 25-node network (b) 40-node network

Fig. 8: Maximum Service Delay with different value of processing capacity in the TACR problem.



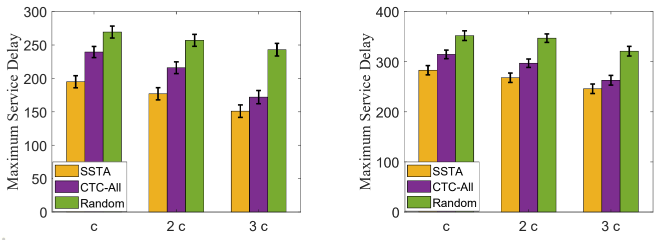
(a) 25-node network (b) 40-node network

Fig. 9: Maximum Service Delay with different value of processing capacity in the S-TACR problem.



(a) 25-node network (b) 40-node network

Fig. 10: Maximum Service Delay with different value of link capacity in the TACR problem.



(a) 25-node network (b) 40-node network

Fig. 11: Maximum Service Delay with different value of link capacity in the S-TACR problem.

VII. Conclusion

In this paper, we study the Task Allocation and Survivable Task Allocation in C-RANs problems with MEC. We first analyze the complexities of these two problems, and show they are NP-hard in general. We subsequently present an exact ILP formulation to jointly solve these two problems. After that, we devise a randomized approximation algorithm to solve the Task allocation in C-RANs problem with MEC. Based on the proposed approximation algorithm, we propose an efficient heuristic to solve the Survivable Task allocation in C-RANs problem with MEC. Via extensive simulations, we find that our proposed algorithms can outperform two benchmark heuristics in terms of acceptance ratio and maximum service delay, although this comes at the expense of higher running time. In our future work, we plan to implement our proposed algorithms in real testbeds [39] and extend the work to jointly consider network management and control related metrics.

Acknowledgment

The work of Song Yang is partially supported by the National Natural Science Foundation of China (NSFC, No. 61802018) and Beijing Institute of Technology Research Fund Program for Young Scholars. The work of Fan Li is partially supported by the NSFC (No. 61772077), and the Beijing Natural Science Foundation (No. 4192051). The work of Xu Chen is partially supported by the National Key Research and Development Program of China (No. 2017YFB1001703), NSFC (No. U1711265) and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No. 2017ZT07X355). The work of Xiaoming Fu is partially supported by the EU H2020 RISE COSAFE project (No. 824019). Song Yang is the corresponding author.

References

[1] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," November 2018, white paper.
[2] K. Chen and R. Duan, "C-RAN: the road towards green RAN," *China Mobile Research Institute, white paper*, vol. 2, 2011.
[3] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, no. 1, pp. 35–41, 2015.
[4] "C-RAN deployment: Market opportunity analysis - 2018 and beyond," 2018. [Online]. Available: https://www.researchandmarkets.com/research/w8ds5f/global_cran?w=12
[5] "5G-oriented C-RAN bearer solution." [Online]. Available: https://carrier.huawei.com/en/solutions/all-cloud-network-towards-5g/cloudran
[6] P. Mell and T. Grance, "The NIST definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.
[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[8] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI White paper*, 2015.
[9] A. Reznik, L. M. C. Murillo, Y. Fang, W. Featherstone, M. Filippou, F. Fontes, F. Giust, Q. Huang, A. Li, C. Turyagyenda *et al.*, "Cloud RAN and MEC: a perfect pairing," *ETSI MEC*, no. 23, p. 25, 2018.

[10] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2244–2281, 2016.

[11] F. A. Kuipers, "An overview of algorithms for network survivability," *ISRN Communications and Networking*, vol. 2012, no. 932456, 2012.

[12] M. F. Hossain, A. U. Mahin, T. Debnath, F. B. Mosharrof, and K. Z. Islam, "Recent research in cloud radio access network (c-ran) for 5g cellular systems - a survey," *Journal of Network and Computer Applications*, vol. 139, pp. 31 – 48, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804519301432

[13] T. Li, C. S. Magurawalage, K. Wang, K. Xu, K. Yang, and H. Wang, "On efficient offloading control in cloud radio access network with mobile edge computing," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2258–2263.

[14] A. Pang, W. Chung, T. Chiu, and J. Zhang, "Latency-driven cooperative task computing in multi-user fog-radio access networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 615–624.

[15] Y. Shih, W. Chung, A. Pang, T. Chiu, and H. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Network*, vol. 31, no. 1, pp. 52–58, 2017.

[16] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429–2445, 2018.

[17] N. Yu, Z. Song, H. Du, H. Huang, and X. Jia, "Multi-resource allocation in cloud radio access networks," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[18] J. Yao and N. Ansari, "Joint content placement and storage allocation in C-RANs for IoT sensing service," *IEEE Internet of Things Journal*, 2018.

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

[20] Q. Liu, T. Han, and N. Ansari, "Energy-efficient on-demand resource provisioning in cloud radio access networks," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 4, pp. 1142–1151, 2019.

[21] C.-C. Hu, "Minimizing executing and transmitting time of task scheduling and resource allocation in c-rans," *Future Generation Computer Systems*, vol. 108, pp. 349 – 360, 2020.

[22] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, 2020.

[23] L. Chen, T. Nguyen, D. Yang, M. Nogueira, C. Wang, and D. Zhang, "Data-driven c-ran optimization exploiting traffic and mobility dynamics of mobile users," *IEEE Transactions on Mobile Computing*, 2020.

[24] A. Samanta, Z. Chang, and Z. Han, "Latency-oblivious distributed task scheduling for mobile edge computing," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[25] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3864–3872, 2019.

[26] A. Samanta, L. Jiao, M. Mühlhäuser, and L. Wang, "Incentivizing microservices for online resource sharing in edge clouds," in *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 420–430.

[27] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled iot," *IEEE Internet of Things Journal*, 2020.

[28] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[29] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for d2d-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

[30] E. El Haber, T. M. Nguyen, and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3407–3421, 2019.

[31] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1524–1537, 2020.

[32] Y. Li, M. Bhopalwala, S. Das, J. Yu, W. Mo, M. Ruffini, and D. C. Kilper, "Joint optimization of BBU pool allocation and selection for C-RAN networks," in *Optical Fiber Communications Conference and Exposition (OFC)*, March 2018, pp. 1–3.

[33] F. Musumeci, C. Bellanzon, N. Carapellese, M. Tornatore, A. Pattavina, and S. Gosselin, "Optimal BBU placement for 5G C-RAN deployment over WDM aggregation networks," *Journal of Lightwave Technology*, vol. 34, no. 8, pp. 1963–1970, 2016.

[34] M. Shehata, O. Ayoub, F. Musumeci, and M. Tornatore, "Survivable BBU placement for C-RAN over optical aggregation networks," in *IEEE International Conference on Transparent Optical Networks (ICTON)*, July 2018, pp. 1–4.

[35] B. M. Khorsandi, C. Raffaelli, M. Fiorani, L. Wosinska, and P. Monti, "Survivable BBU hotel placement in a C-RAN with an optical WDM transport," in *IEEE International Conference Design of Reliable Communication Networks (DRCN)*, March 2017, pp. 1–6.

[36] B. M. Khorsandi and C. Raffaelli, "BBU location algorithms for survivable 5G C-RAN over WDM," *Computer Networks*, vol. 144, pp. 53 – 63, 2018.

[37] B. M. Khorsandi, F. Tonini, and C. Raffaelli, "Design methodologies and algorithms for survivable C-RAN," in *IEEE International Conference on Optical Network Design and Modeling (ONDM)*, 2018, pp. 106–111.

[38] N. Mharsi and M. Hadji, "Edge computing optimization for efficient rrh-bbu assignment in cloud radio access networks," *Computer Networks*, vol. 164, p. 106901, 2019.

[39] A. Younis, T. X. Tran, and D. Pompili, "Bandwidth and energy-aware resource allocation for cloud radio access networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6487–6500, 2018.

[40] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *IEEE INFOCOM 2014-IEEE conference on computer communications*. IEEE, 2014, pp. 1842–1850.

[41] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106 – 127, 2016.

[42] Y. Zhu, Y. Liang, Q. Zhang, X. Wang, P. Palacharla, and M. Sekiya, "Reliable resource allocation for optically interconnected distributed clouds," in *Proc. of IEEE International Conference on Communications (ICC)*, 2014, pp. 3301–3306.

[43] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2014.

[44] Z. Cai, X. Li, and R. Ruiz, "Resource provisioning for task-batch based workflows with deadlines in public clouds," *IEEE Transactions on Cloud Computing*, 2017.

[45] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[46] M. Leconte, A. Destounis, and G. Paschos, "Traffic engineering with precomputed pathbooks," in *IEEE INFOCOM*, 2018.

[47] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

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

[49] C.-L. Li, S. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105 – 115, 1990.

[50] K. M. Anstreicher, "Linear programming in o ([n3/ln n] l) operations," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 803–812, 1999.

[51] R. Tarjan, *Course: Advanced Algorithm Design. Lecture: Chernoff, Probability and Computing*. Princeton University, 2009.

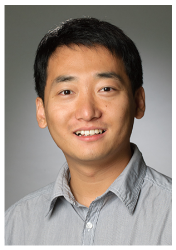
[52] L. Comtet, *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. Springer Netherlands, 1974.

[53] H. Xu, Z. Yu, X. Y. Li, L. Huang, C. Qian, and T. Jung, "Joint route selection and update scheduling for low-latency update in SDNs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3073–3087, 2017.

[54] S.-C. Lin, P. Wang, I. Akyildiz, and M. Luo, "Towards optimal network planning for software-defined networks," *IEEE Transactions on Mobile Computing*, 2018.

[55] H. Wang, X. Yu, H. Xu, J. Fan, C. Qiao, and L. Huang, "Integrating coflow and circuit scheduling for optical networks," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[56] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, “Joint service placement and request routing in multi-cell mobile edge computing networks,” in *IEEE INFOCOM*, 2019, pp. 10–18.
[57] J. W. Suurballe and R. E. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, no. 2, pp. 325–336, 1984.
[58] K. Calvert and E. Zegura, “Gt-itm: Georgia tech internetwork topology models,” 1996.
[59] K. Gardner, S. Borst, and M. Harchol-Balter, “Optimal scheduling for jobs with progressive deadlines,” in *INFOCOM*, 2015, pp. 1113–1121.
[60] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, “Improving traffic forecasting for 5G core network scalability: A machine learning approach,” *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018.



Song Yang is currently an associate professor at School of Computer Science in Beijing Institute of Technology, China. Song Yang received the B.S. degree in software engineering and the M.S. degree in computer science from Dalian University of Technology, Dalian, Liaoning, China, in 2008 and 2010, respectively, and the Ph.D. degree from Delft University of Technology, The Netherlands, in 2015. From August 2015 to August 2017, he worked as postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. His research interests focus data communication networks, cloud/edge computing and network function virtualization. He is a member of IEEE and ACM.

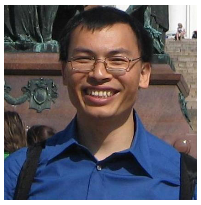
He Nan is currently a Ph.D. student in the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include network function virtualization and network resource optimization management based on reinforcement learning.

Fan Li received the PhD degree in computer science from the University of North Carolina at Charlotte in 2008, MEng degree in electrical engineering from the University of Delaware in 2004, MEng and BEng degrees in communications and information system from Huazhong University of Science and Technology, China in 2001 and 1998, respectively. She is currently a professor at School of Computer Science in Beijing Institute of Technology, China. Her current research focuses on wireless networks, ad hoc and sensor networks, and mobile computing. Her papers won Best Paper Awards from IEEE MASS (2013), IEEE IPCCC (2013), ACM MobiHoc (2014), and Tsinghua Science and Technology (2015). She is a member of IEEE and ACM.

Stojan Trajanovski received his PhD degree (cum laude, 2014) from Delft University of Technology, The Netherlands and his master degree in Advanced Computer Science (with distinction, 2011) from the University of Cambridge, United Kingdom. He is currently an applied scientist in Microsoft, working in London, UK and Bellevue, WA, USA. He was in a similar role with Philips Research in Eindhoven, The Netherlands from 2016 to 2019. Before that, he spent some time as a postdoctoral researcher at the University of Amsterdam and at Delft University of Technology. He successfully participated at international science olympiads, winning a bronze medal at the International Mathematical Olympiad (IMO) in 2003. His main research interests include network science & complex networks, machine learning, game theory, and optimization algorithms.

Xu Chen is a Full Professor with Sun Yat-sen University, Guangzhou, China, and the vice director of National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012, and worked as a Postdoctoral Research Associate at Arizona State University, Tempe, USA from 2012 to 2014, and a Humboldt Scholar Fellow at Institute of Computer Science of University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by Alexander von Humboldt Foundation of Germany, 2016 Thousand Talents Plan Award for Young Professionals of China, and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an Associate Editor of IEEE Internet of Things Journal and IEEE Journal on Selected Areas in Communications (JSAC) Series on Network Softwarization and Enablers.

Yu Wang is currently a Professor in the Department of Computer and Information Sciences at Temple University. Prior to joining Temple University, he was a Professor of Computer Science at the University of North Carolina at Charlotte (UNC Charlotte). He holds a Ph.D. from Illinois Institute of Technology, an MEng and a BEng from Tsinghua University, all in Computer Science. His research interest includes wireless networks, smart sensing, and mobile computing. He has served as Editorial Board Member of several international journals, including IEEE Transactions on Parallel and Distributed Systems. He is a recipient of Ralph E. Powe Junior Faculty Enhancement Awards from Oak Ridge Associated Universities (2006), Outstanding Faculty Research Award from College of Computing and Informatics at UNC Charlotte (2008), and Fellow of IEEE (2018). He is also a senior member of ACM and a member of AAAS.



Xiaoming Fu received his Ph.D. in computer science from Tsinghua University, Beijing, China in 2000. He was then a research staff at the Technical University Berlin until joining the University of Göttingen, Germany in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. Prof. Fu’s research interests include network architectures, protocols, and applications. He is currently an editorial board member of IEEE Communications Magazine, IEEE Transactions on Network and Service Management, and Elsevier Computer Communications, and has served on the organization or program committees of leading conferences such as INFOCOM, ICNP, ICDCS, MOBICOM, MOBIHOC, CoNEXT, ICN and COSN. He is an IEEE Senior Member, an IEEE Communications Society Distinguished Lecturer, a fellow of IET and member of the Academia Europaea.