Placement, Routing and Scheduling Optimization in C-RAN

Makhlouf Hadji, Hatem Ibn-Khedher, Ahmed E.Kamal

Placement, Routing and Scheduling Optimizations in Cloud-RAN

Hatem Ibn-khedher [§], Makhlouf Hadji [¶], and Ahmed E. Kamal[‡]
[§]ALTRAN Labs,78140 Velizy-Villacoublay, France.
Email: hatem.ibnkhedher@altran.com
[¶]Technological Research Institute - IRT SystemX, 8 avenue de la vauve, 91120, Palaiseau, France.
Email: makhlouf.hadji@irt-systemx.fr
[‡]Department of Electrical Computer Engineering, Iowa State University, Ames, IA 50011-3060, USA.
Email: see http://www.ece.iastate.edu/ kamal/

***Abstract*—The density increasing in Radio Access Networks (RAN) caused the migration of traditional base stations to the cloud to meet huge traffic of end-users' demands. In this context, virtualization techniques can add more flexibility and programmability to scale in/out virtual storage, network and computing resources. However, Cloud-RAN (C-RAN) requires real-time processing and scheduling of its demands represented as service chains. In this paper, we formulate the joint assignment and scheduling problem in C-RAN using linear programming approach. Placement and scheduling algorithms allowing to allocate efficiently computing resources for C-RAN Virtual Network Functions (VNFs) with respect to the RAN services chaining are introduced and their behavior is quantified through real traces. We illustrate and highlight the feasibility and efficiency of our proposed algorithms through different scenarios in various considered network instances. Metrics such as cpu cores occupancy, network throughput, and successful subframe decoding rate are used to illustrate our algorithms' efficiency.**

***Index Terms*—Cloud-Radio Access Network; radio parameters; BBU processing time; Optimization; Resource Allocation**

I. Introduction

Cloud Radio Access Networks (C-RAN) is a promising paradigm that aims to centralize (i.e., cloudify) certain Base Band Unit (BBU) processing functions while adding more flexibility and increasing the overall network performance [1] [2]. European Telecommunications Standards Institute (ETSI) has highlighted C-RAN among the major use cases [3]. The main purpose of C-RAN is to allow the operator to dynamically deploy/process on demand virtual base stations (BBU) to deal with the massive growing amount of cellular network traffic [4] [5] [6]. Scaling in/out, BBU as a service (BBUaaS), network slicing and security etc.. are also among the key benefits of C-RAN [7] [8] [9] [10].

ETSI-MANO standardizes a framework for deploying different Virtual Network Functions (VNFs) and in our case, RAN is the target VNF. In this paper, we follow this standard and propose our specific design and architecture for RAN that can be used in 5G deployment. This latter (C-RAN or vRAN) extends the virtual network functions (VNFs) to FFT (or Inverse FFT (IFFT)), demodulation (or modulation) and decoding (or coding). The VNFs are controlled by a Virtual Network Function Manager (VNFM).

NFV Infrastructure (NFVI) is composed of three domains: i) virtual computing domain, ii) virtual storage domain, and iii) virtual networking domain. NFVI is managed by cloud management platform (e.g., OpenStack) which corresponds to the Virtual Infrastructure Manager (VIM) that manages VMs or dockers used in the virtualization process. The proposed C-RAN architecture has a global orchestration that manages and orchestrates the C-RAN VNFMs (if there are many of them), OpenStack, and OSS/BSS. This latter manages QoS, network failure, and security in 5G radio access.

Currently, despite the importance of C-RAN optimization tasks (i.e., the C-RAN scheduler module), such functions are missing including the global architecture itself. We therefore, propose in this paper to contribute with placement, routing and scheduling optimizations (OSA and OTA) in Cloud-RAN and illustrate how it can be integrated by network operators.

The proposed optimization algorithms in this paper consider C-RAN chaining, placement, routing and scheduling problems in a multi-core platform (i.e., the BBU pool) which is a part of future operator network. Moreover, the major objective from the proposed algorithms is to guarantee that the BBU pool has met their real-time deadlines related to the frame/subframe timing while minimizing the additional extra-costs needed for processing virtual base station instances and chaining the micro instances. Through this optimization, we investigate exact algorithms based on linear programming techniques for deciding about the optimal locations to place gNB subframes on available cpu cores to satisfy user's quality requirements and minimize the total number of deadline misses[1].

In the context of C-RAN, subframes are considered as service function chains of three main subfunctions given by Fast Fourier Transform (FFT), demodulation, and decoding. In this work, we propose linear programming algorithms that can ensure the precedence constraint (i.e., chaining and sequencing) of the subframe subfunctions inside the network operator. Moreover, our objective in the proposed algorithm is to maximise the total number of decoded subframes while minimizing the additional extra-costs needed for chaining, real-time processing, and handling starting and completion times. Through the first optimization (i.e. OSA: Optimal Subframe Allocation in C-RAN), we propose to formulate an exact algorithm based on a mathematical model for deciding about the optimal chaining, scheduling, and assignment of subframe

[1]It results from the compute node not being able to decode a subframe within a deadline.

subfunctions starting and completion time. We identify cpu cores that have to process subfunctions. Further, to cope with network diversity and loads, we adapt another optimization algorithm (i.e., OTA: Optimal Throughput Allocation) that maximises the overall network throughput to deal with our constraints when the network operator prioritizes the network throughput over the number of successfully decoded subframes. In these algorithms, we formulate the precedence constraints in real time C-RAN scheduling context. As the optimization algorithms deal with many heteroclite parameters, we propose two different scenarios depending on the number of cpu cores available in the pool and RRH size.

The rest of this paper is organized as follows: Section II highlights recent C-RAN architectures and the state-of-the art of the scheduling approaches. Section III describes the context and the optimisation algorithms. Section IV introduces the real traffic model used as an input to the proposed algorithms. Section V assesses the performance of the proposed optimization algorithms using different metrics. We conclude the paper in Section VI.

II. Related Work

In [4], authors study the constrained resource allocation (or RRH-BBU assignment) problem in C-RAN. They proposed novel approaches based on matroids theory to deal with large network scale. Further, authors proved that their algorithms are polynomial time under some "light" assumptions.

In [11], authors explore the design of C-RAN architecture. They study the problem of determining the required computing capacity for C-RAN functions placement. They model the BBU execution time and evaluate different scheduling strategies that can be proposed such as scheduling per LTE subframe as proposed in [12]. Further, some design consideration in C-RAN scheduler are proposed as follows.

In [13] authors affirm that an optimal scheduling algorithm can be mathematically formulated as follows $R_{opt} = argmax_{R \in R'} \sum_{r_k \in R} r_k$ subject to: $\sum_{r_k \in R} c_k \leq c_{server}$ where r_k is the rate of the user k, c_k is its offered computational load, and c_{server} is the total available computing power. Optimal solution results in a water-filling algorithm. Heuristic alternative solution is to simply pick the user with highest complexity and back off its rate until the complexity constraint is satisfied. In their work, three approaches are proposed: max-rate scheduling (MRS), scheduling with complexity cutoff (SCC) and scheduling with water filling (SWF).

In [14], the authors propose an algorithm that optimizes the number of allocated BBU virtual resources by considering both the computational requirements of the RRH and the Quality of Service of users. In this [15], these resources are allocated in the Coordinated Multi-Point (CoMP) use case, where a user equipment (UE) is connected to multiple base stations simultaneously.

III. VNF Chaining and Scheduling Algorithms

Consider a BBU pool composed of homogeneous cpu cores with the same execution speeds that execute the BBU functions of the RRHs. Each RRH sends a subframe every Transmission Time Interval (TTI), where each subframe can be seen as a sequence of the three following BBU subfunctions: FFT, demodulation and decoding. It is worth mentioning that these subfunctions should be executed in series (i.e., one function can start only when the output of the previous one is available). Moreover at a given instant of time, a cpu core can process at most one subframe sub-function that can not be split among different cores (e.g., a subframe subfunction can be processed by at most one cpu core at a time). Our main objective is to maximize the total number of correctly processed subframes.

TABLE I: Summary of the general mathematical notation

Parameters	***Definition***
$\mathcal{N}$	The set of RRHs
$\mathcal{C}$	The set of CPU cores in a shared BBU pool (muti-core data center).
$\mathcal{K}$	The set of subframe subfunctions (i.e., FFT, ULSCH[2] demodulation, and ULSCH decoding)
$t_{i,k}$	The processing time of the subfunction $k \in K$ of the subframe $i \in \mathcal{N}$
M	a large number equivalent to infinity
d	The subframe processing time deadline
Decision Variables	***Definition***
$x^c_{i,k}$	A binary variable that assigns the subfunction $k \in \mathcal{K}$of subframe $i \in \mathcal{N}$ to the core $c \in \mathcal{C}$.
$s^c_{i,k}$	A real variable that indicates the starting time of the subfunction $k \in \mathcal{K}$ of subframe $i \in \mathcal{N}$ on core $c \in \mathcal{C}$
$h^c_{i,k}$	A real variable that indicates the completion time of the subfunction $k \in \mathcal{K}$ of subframe $i \in \mathcal{N}$ on core $c \in \mathcal{C}$
$y^c_{(i,k),(i',k')}$	A binary variable that assures that the subframe subfunction (i, k) precedes (i', k') on a specific cpu core $c \in \mathcal{C}$
C_i	the completion time of the subframe $i \in \mathcal{N}$
C_{max}	the makespan that represents the maximum completion time over all the subframes

A. Notations

Before providing a deep mathematical formulation of the addressed problem, and for sake of clarity, we propose a notation of the used parameters and variables in our modelling. Let $\mathcal{C}$ be the set of homogeneous cpu cores, $\mathcal{N}$ is the set of RRHs where each one of them has, each TTI, a subframe to be executed. In this paper, we focus on only one TTI so that the number of RRHs is equal to the number of subframes to be executed. Let $\mathcal{K}$ be the set of subframe subfunctions: FFT, Uplink Shared Channel (ULSCH) demodulation, and ULSCH decoding. The processing time of the subfunction $k \in \mathcal{K}$ of subframe $i \in \mathcal{N}$ is denoted by $t_{i,k}$. Furthermore, each subframe has a processing time deadline d that should not be missed, otherwise the three subfunctions of it have to be re-executed. We summarize optimization parameters and variables in Table I.

In this subsection, we formulate the scheduling problem of computing resources (cpu cores) in C-RAN using Integer Linear Programming (ILP) technique. A centralized single decision-maker optimization problem whose objective is to maximize the total number of correctly processed subframes is proposed. This centralized setting would correspond to the case where a centralized management entity ensures the global scheduling of cpu cores among the different subframes.

In our approach, we consider only one TTI assuming even and odd TTIs are assigned to separate cpu cores, so that tasks corresponding to different TTIs do not overlap. Moreover, we propose two mathematical formulations. In the first, our methods processes all subfunctions forming a particular VNF upon the same cpu core. In the second, the method processes each VNF component upon an optimal cpu core. We describe hereafter these two methods.

B. Entire subframe allocation to a dedicated core

In this optimization model, the complete processing function of a subframe is performed on a single cpu core. Therefore, we introduce the binary variable x_i^c that indicates the allocation of the subframe $i \in N$ on the optimal CPU core $c \in C$. It is defined as:

$$x_i^c = \begin{cases} 1 & \text{if the subframe } i \text{ is allocated on the CPU core } c \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

The constraints proposed in this algorithm are:

1) Subframes must be processed by at most one cpu core

$$\sum_{c \in \mathcal{C}} x_i^c \leq 1, \ \forall i \in \mathcal{N} \tag{2}$$

2) CPU cores have a budget of 2 ms after which they should drop the unfinished processed subframes. This budget represents the hard deadline to decode the subframes

$$\sum_{i \in \mathcal{N}} x_i^c \left(\sum_{k \in \mathcal{K}} t_{i,k} \right) \leq d, \ \forall c \in \mathcal{C} \tag{3}$$

3) Subframes must be processed before the hard deadline 2 ms in the uplink direction.

$$\sum_{c \in \mathcal{C}} x_i^c \left(\sum_{k \in \mathcal{K}} t_{i,k} \right) \leq d, \ \forall i \in \mathcal{N} \tag{4}$$

In order to maximize the average quality of experience (QoE) of User Equipment's (UEs), subframes allocation should be maximized. Therefore, the proposed objective function has to include this network indicator. Hence, it is formulated by (5) (gain function):

$$\sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} x_i^c \tag{5}$$

C. OSA: Optimal Subframe subfunctions Allocation

In the following, we leverage the decomposition of subframes into subfunctions in order to improve the allocation of compute resources at the central entity. Decision variables and OSA algorithm constraints are introduced in the following subsections.

1) Decision variables of OSA:

- Binary variables $x_{i,k}^c$ indicate the placement (allocation) of the subframe subfunction on the optimal cpu core c. It is defined as:

$$x_{i,k}^c = \begin{cases} 1 & \text{if the sub-function } k \in \mathcal{K} \text{ of the subframe} \\ & i \in \mathcal{N} \text{ is allocated on the CPU core } c \in \mathcal{C} \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

- $s_{i,k}^c$ indicates the starting time of the subfunction k of subframe i on cpu core c. It is defined as follows:

$$s_{i,k}^c = \begin{cases} \geq 0 & if \ \sum_{c \in \mathcal{C}} x_{i,k}^c = 1 \\ & \big(\text{i.e., if the subframe sub-function } (i,k) \\ & \text{is assigned to a CPU core } \big) \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

- Binary variables $y_{(i,k),(i',k')}^c$ indicate that subframe subfunctions (i,k) and (i',k') cannot be processed at the same time on any cpu core $c \in \mathcal{C}$. It is defined by:

$$y_{(i,k),(i',k')}^c = \begin{cases} 1 & \text{if subframe subfunction } (i,k) \\ & \text{precedes subfunction } (i',k') \\ & \text{on core } c \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

2) Linear constraints of OSA: It is worth mentioning that the proposed Optimal Subframe Allocation (OSA) aims to chain and schedule C-RAN VNFs rather than a complete VNF. This algorithms consider the following constraints:

- Constraint (9) defines the single core assignment constraint. It implies that each of the three sub-functions of each subframe should be assigned to at most one CPU core c

$$\sum_{c \in \mathcal{C}} x_{i,k}^c \leq 1, \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{9}$$

- If the subframe subfunction is not assigned to a cpu core, constraint (10) sets the starting and completion times of the current task on cpu core c equal to zero

$$s_{i,k}^c + h_{i,k}^c \leq x_{i,k}^c M, \forall i \in \mathcal{N}, k \in \mathcal{K}, c \in \mathcal{C} \tag{10}$$

- If it exists an available cpu core that can process such a subframe subfunction, then constraint 11 assures that the difference between its starting and completion times is equal to the processing time.

$$h_{i,k}^c \geq s_{i,k}^c + t_{i,k} - (1 - x_{i,k}^c) M, \forall i \in \mathcal{N}, k \in \mathcal{K}, c \in \mathcal{C} \tag{11}$$

- Constraints (12) and (13) consider the requirement that subframe subfunctions cannot be done at the same time.

$$s_{i,k}^c \geq h_{i',k'}^c - y_{(i,k),(i',k')}^c M, \forall i < i' \in \mathcal{N}, k, k' \in \mathcal{K}, c \in \mathcal{C} \tag{12}$$

$$s_{i',k'}^c \geq h_{i,k}^c - (1 - y_{(i,k),(i',k')}^c) M, \forall i < i' \in \mathcal{N}, k, k' \in \mathcal{K} \tag{13}$$

- Further, constraint (14) ensures that the precedence relationships between the subfunctions of a subframe is not violated. In fact, for each subframe i, the starting time of the subfunction k must start after the completion time of the subfunction $k-1$.

$$\sum_{c \in \mathcal{C}} s_{i,k}^c \geq \sum_{c \in \mathcal{C}} h_{i,k-1}^c, \forall i \in \mathcal{N}, k \in \mathcal{K} - \{1\} \tag{14}$$

- Constraint (15) assures that the decoding time of each subframe, running on a single cpu core, must finish before the hard deadline d.

$$C_i = \sum_{c \in \mathcal{C}} h_{i,3}^c, \forall i \in \mathcal{N} \tag{15}$$

- Constraint (16) defines the core deadline constraint. Recall that the amount of processing time a subframe i would require, if run entirely on core c, is $t_i = \sum_k t_{i,k}$. Therefore, for a set of fractional assignments $x_{i,k}^c$, we can determine the amount of time core c will work. It is given by $\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} x_{i,k}^c t_{i,k}$ and it should be at most equal to d. In other words, each core c must be able to finish the sub-functions assigned to it before the deadline

d (otherwise the sub-function should not be assigned to core c). This ensures also that each subframe should be processed in a time lower than the deadline d.

$$s^c_{i,k} + \sum_{c \in \mathcal{C}} x^c_{i,k} t_{i,k} \leq d, \forall i \in \mathcal{N}, k \in \mathcal{K} \quad (16)$$

- Constraint (17) ensures that three sub-functions of each subframe must be executed in series; for a given subframe, the allocation of one sub-function to a core should not be done if the previous sub-functions have not been allocated. This guarantees that, at the end, if a sub-function is allocated, then the previous sub-functions are also allocated.

$$\sum_{c \in \mathcal{C}} x^c_{i,k} \geq \sum_{c \in \mathcal{C}} x^c_{i,k+1}, \forall i \in \mathcal{N}, k \in \mathcal{K} - \{3\} \quad (17)$$

- Constraints (18, 19, and 20) ensure the non-negativity of the proposed decision variables.

$$s^c_{i,k} \geq 0, \forall i \in \mathcal{N}, k \in \mathcal{K}, c \in \mathcal{C} \quad (18)$$

$$h^c_{i,k} \geq 0, \forall i \in \mathcal{N}, k \in \mathcal{K}, c \in \mathcal{C} \quad (19)$$

$$C_i \geq 0, \forall i \in \mathcal{N} \quad (20)$$

3) The mono-objective resolution: As our objective is to maximize the total number of correctly processed subframes that is equivalent to minimizing the total number of deadline misses; it turns out that the objective function of our optimization problem as defined in Eq. (21) is to maximize the allocation of the last sub-function (i.e., third one that corresponds to the decoding subfunction in our reference split.

$$\textit{OSA:} \ \max \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} x^c_{i,3} \quad (21)$$

D. OTA: Optimal Throughput Allocation

From network operator perspective, maximizing the overall traffic throughput of all the allocated subframes may be prioritized in some cases. Then, maximizing $\max \sum_{i \in N, c \in C} x^c_{i,3} b_i$ quantity may offer higher throughput than Eq. (21). We propose then a slight modification of OSA in order to better quantify the behavior of the proposed algorithm when network operator has to maximize the throughput rather than bit error rate. Optimal Throughput Allocation (OTA) algorithm may solve this issue. Its general ILP formulation will have the following objective function:

$$\textit{OTA:} \ \max \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} x^c_{i,3} b_i \quad (22)$$

The above constraints used in OSA are remained the same in OTA optimization.

IV. Real traffic modeling and analysis

The performance of different resource allocation algorithms strongly depends on the cellular traffic pattern. Thus, it is important to evaluate optimization techniques with real (representative) traffic models. Generally, a precise description and characterization of the offered traffic would require to know the following parameters: i) the number of RRHs, ii) the number of users, iii) the number of resources block (RBs) per user, iv) the Modulation and Coding Scheme (MCS) per user, v) the Signal to Noise Ratio (SNR) per user on each RRH, and vi) the processing times per MCS.

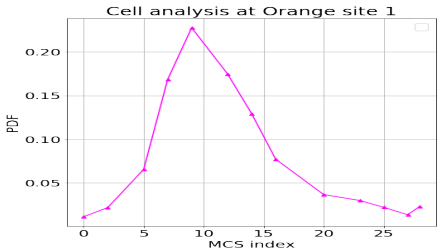


Figure 1: Probability density function of MCS

For sake of clarity, we consider real traces provided by a network operator. These traces are cell statistics over 7 days at a scale of 15 minutes per sample. They are multi-variate time series data used to analyze and obtain the previous mentioned parameters (MCS, SNR, PRB, etc.). Then, the input traffic features can lead to deduct MCS (and CQI/SNR) distribution and therefore better optimise and manage computing resources.

We derive the probability density function (PDF) of a subframe to carry traffic of a given MCS from the real traces. The PDF is represented in Fig. 1. It is used for selecting MCS indexes for different subframes and feeding the optimization algorithms described in the previous section.

We assume that user subframes are composed of 100 PRBs (i.e., RRHs operate in the 20 MHz bandwidth). Further, since that our proposed algorithm OTA is interested in maximizing the total throughput (b_i), we need to quantify the amount of data that resides in each MCS as in our previous work [16].

Moreover, we have modeled the processing time for each subframe subfunction $t_{i,k}$ $\forall i \in N, k \in k$ using real simulations according to the OpenAirInterface, a 5G network simulator [17].

V. Performance Evaluation

We evaluate our algorithms in a C-RAN network architecture that processes the VNF components (vFFT, vDEM, and vDEC) of a base station in a central server. We note that RRH may be assigned to any cpu core in the BBU pool. Recall that a pool is considered as a cloud of homogeneous computing resources (i.e., a set of cpu cores). For sake of simplicity we can suppose that a BBU is represented as a central processing unit. To better quantify the behavior of the above algorithms, we propose two different scenarios according to the network loads. The first scenario considers a BBU pool having a single cpu core that can host 25 RRH at maximum while the second scenario considers two cpu cores. Both scenarios consider 100 runs (instances) of the above MCS distribution. Then, average values of different metrics will be presented in the sequel. In the first scenario, we assess the trade-off between the number of non-decoded subframes and the offered throughput. The second scenario is proposed to show the fairness behavior of the proposed algorithms.

Further, we have used real traces from a real network operator. In Fig. 2, we show the total traffic demand that represents the requested bandwidth. It increases linearly with growing values of RRHs. Moreover, for the interest of assessing the efficiency of OSA and OTA algorithms, we propose different Key Performance Indicators (KPI) as follows:

- **Average undecoded subframes:** It represents the number of subframes that can not be processed before its hard

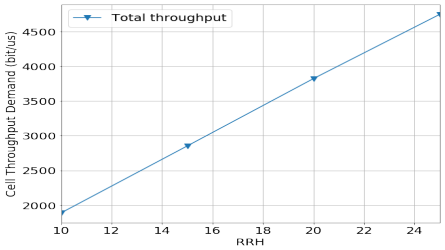


Figure 2: Total traffic demand as a function of the number of RRHs assigned to a BBU pool

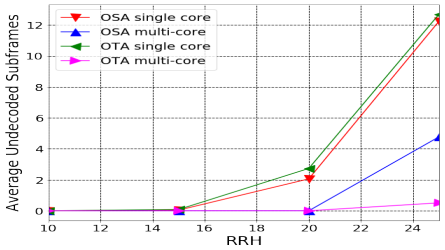


Figure 3: Average undecoded subframes

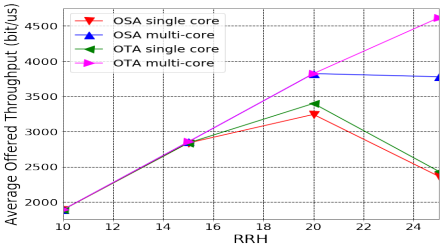


Figure 4: Average offered throughput

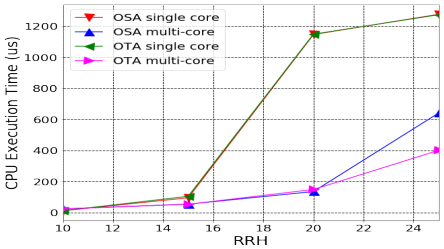


Figure 5: CPU execution times

deadline.

- **Average offered throughput:** It represents the optimal throughput allocation according to the VNF chaining and scheduling linear programming technique.
- **Average unused CPU time:** It represents the remaining time slots at CPU cores.
- **Average CPU execution time:** It represents the average computational load.
- **Average makespan:** It represents the overall completion time of all the subframes.

A. Average undecoded subframes

When the processing time of a subframe subfunction exceeds the deadline, the ILP scheduler decides to drop the current subframe. So we have an interest in minimizing the undecoded number of subframes. In Fig. 3, We measured the average number of undecoded subframes against the number of RRH. Result shows the efficiency and the feasibility of the proposed ILP algorithms. In fact, when we consider the 2 cpu cores configuration, we notice that the proposed models do not show losses at the networking layer from RRH 10 to RRH 18. Then, we receive non significant number of RRH's subframes that need retransmission. We mention here that the HARQ mechanism is not considered in the above formulation. Introducing this mechanism can enhance the packet loss and the end-to-end consumer delay. Further, when we consider the first scenario of 1 cpu core, we can see that the number of undecoded subframes is significant. This is due to the selection of high MCS indices which require high processing time. the behavior of the two algorithms in this scenario is almost the same. In summary, OSA outperforms OTA in terms of the number of undecoded subframes. Further, adding more cpu cores can enhance the computational load and minimize the subframe loss. However, the behavior of algorithms is inverted due to the chaining of subframe subfunctions.

B. Average offered throughput

In C-RAN context, user equipment demands (in terms of bandwidth) can exceed the available bandwidth (Fig. 2). Therefore, offering the optimal throughput to users is highly recommended to deal with the demand. In Fig. 4, we show the average offered (allocated) throughput against the number of RRH. Result shows that the OTA algorithm outperforms OSA technique in terms of overall throughput allocation when RRH number exceeds 20 nodes. This peak represents the average number of nodes from which the system begins to lose the incoming subframes. The result recommends also the use of multi-core systems to increase the overall network throughput.

C. CPU execution time

In Fig. 5, we show the CPU execution time according to different RRH sizes. Results show that CPU execution time does not increase exponentially with the growing of RRH numbers. This is due to the feasability and efficiency of the proposed ILPs. Moreover, OTA outperforms OSA in terms of this KPI (i.e. CPU time).

D. Average unused CPU time

In Fig. 6, we show that unused CPU time decreases with RRH size. Both algorithms gives the same behavior when using a single cpu core, however, using 2 cpu cores we may add some flexibility to the subframe throughput allocation and we can modify the behavior of the algorithm. Therefore, we show in Fig. 7 the percentage of the unused CPU time at each CPU core. we clearly see that OTA outperforms OSA in terms of fairness. Indeed cpu resources are fairly shared under its allocation strategy.

E. Average makespan

We have defined the makespan as the maximum completion time over all the subframes that share a central BBU pool. In our context that targets real-time applications, it is well envisaged that our C-RAN completes the scheduling of all

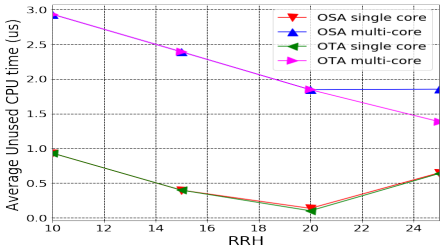


Figure 6: Average unused CPU time

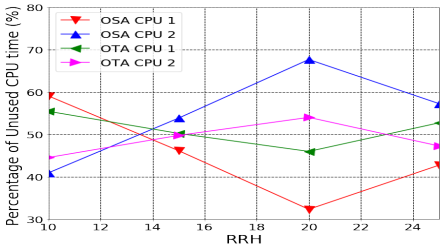
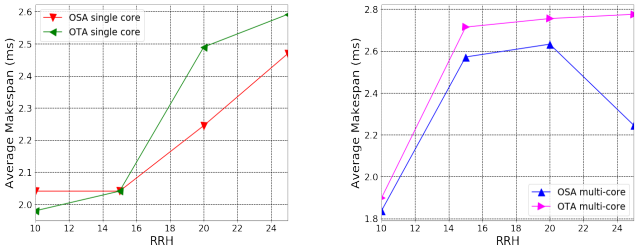


Figure 7: Average unused CPU time at each CPU core



(a) Serial processing (b) Parallel processing

Figure 8: Average makespan according to 2 main processing scenarios

subframes in 2 TTI (e.g., before the soft deadline). In Fig. 8, we plot the average makespan when RRH number increases from 10 to 25. Results show that OSA algorithm that prioritizes short subframes (i.e., with small MCS indices) outperforms OTA algorithm that prioritizes long subframes (i.e., with big MCS indices). We can deduce also that both algorithms work better with a single cpu core.

VI. Discussion and Conclusion

This paper presents two optimization solutions either for the chaining or the scheduling problem of cloud RAN. Multiple constraints that are strongly related to the RAN virtualization are taken into account such as subframe subfunctions processing time, subframe processing deadline, and precedence constraints and requirements. In addition, the two optimization algorithms OSA and OTA target respectively different objectives (average number of decoded subframes and average offered throughput). Then, they are formulated, implemented, and analysed. In small BBU sizes, results show that $|\mathcal{N}|$ have more significant impact on the average number of undecoded subframe and offered throughput in the case of using OTA rather than OSA. Nevertheless, it is noticeable that both algorithms have almost the same behavior. In large BBU size, we have observed that OTA works better in terms of the previous key metrics. Further, OTA gives a short execution time which illustrates its efficiency and scalability. Moreover, the two approaches provide significant unused cpu time and less makespan in different scenarios (single VNF and VNF chaining).

References

[1] L. Gavrilovska, V. Rakovic, and D. Denkovski, "From cloud ran to open ran," *Wireless Personal Communications*, pp. 1–17, 2020.

[2] A. G. Dalla-Costa, L. Bondan, J. A. Wickboldt, C. B. Both, and L. Z. Granville, "Orchestra: A customizable split-aware nfv orchestrator for dynamic c-ran," *IEEE Journal on Selected Areas in Communications*, 2020.

[3] E. G. N. V1.1.1. (2013) Network functions virtualization (nfv); use cases.

[4] N. MHARSI and M. Hadji, "Joint Optimization of Communication Latency and Resource Allocation in Cloud Radio Access Networks," in *7th IEEE International Conference on Smart Communications in Network Technologies (SaCoNeT 2018)*, EL Oued, Algeria, Oct. 2018.

[5] W. Xia, T. Q. Quek, J. Zhang, S. Jin, and H. Zhu, "Programmable hierarchical c-ran: From task scheduling to resource allocation," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 2003–2016, 2019.

[6] H. Khedher, H. Afifi, and H. Moustafa, "Optimal placement algorithm (opa) for iot over icn," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 372–377.

[7] Z. Luo and C. Wu, "An online algorithm for vnf service chain scaling in datacenters," *IEEE/ACM Transactions on Networking*, pp. 1–13, 2020.

[8] H. Touati, H. Castel-Taleb, B. Jouaber, and S. Akbarzadeh, "Split analysis and fronthaul dimensioning in 5g c-ran to guarantee ultra low latency," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–4.

[9] H. Zhang and V. W. Wong, "A two-timescale approach for network slicing in c-ran," *IEEE Transactions on Vehicular Technology*, 2020.

[10] C.-C. Hu, "Minimizing executing and transmitting time of task scheduling and resource allocation in c-rans," *Future Generation Computer Systems*, 2020.

[11] V. Q. Rodriguez and F. Guillemin, "Performance analysis of vnfs for sizing cloud-ran infrastructures," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–6.

[12] S. Bhaumik, S. Preeth Chandrabose, M. Kashyap Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan, and T. Woo, "Cloudiq: A framework for processing base stations in a data center," 08 2012.

[13] M. C. Valenti, S. Talarico, and P. Rost, "The role of computational outage in dense cloud-based centralized radio access networks," *CoRR*, vol. abs/1407.1830, 2014.

[14] A. Okic and A. E. C. Redondi, "Optimal resource allocation in c-ran through dsp computational load forecasting," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–7.

[15] J. Khan and L. Jacob, "Resource allocation for comp enabled urllc in 5g c-ran architecture," *IEEE Systems Journal*, pp. 1–12, 2020.

[16] H. Kheder, S. Hoteit, P. Brown, R. Krishnaswamy, W. Diego, and V. Vèque, "Processing time evaluation and prediction in cloud-ran," in *Proceedings of International Conference on Communications ICC*, 2019.

[17] OpenAirInterface. (2021) https://openairinterface.org/.