A Recurrent Neural Network Based Approach for Coordinating Radio and Computing Resources Allocation in Cloud-RAN

Mahdi Sharara, Sahar Hoteit, Véronique Vèque

A Recurrent Neural Network Based Approach for Coordinating Radio and Computing Resources Allocation in Cloud-RAN

Mahdi Sharara*, Sahar Hoteit*, and Véronique Vèque*
*Université Paris Saclay-CNRS-CentraleSupélec,
Laboratoire des Signaux et Systèmes, 91190, Gif-sur-Yvette, France
Emails: {mahdi.sharara, sahar.hoteit, veronique.veque}@universite-paris-saclay.fr

***Abstract*—Cloud Radio Access Network (Cloud-RAN) is a novel architecture that aims at centralizing the baseband processing of base stations. This architecture opens paths for joint, flexible, and optimal management of radio and computing resources. To increase the benefit from this architecture, efficient resource management algorithms need to be devised. In this paper, we consider a coordinated allocation of radio and computing resources to mobile users. Optimal resource allocation that respects the Hybrid-Automatic-Repeat-Request deadline may require formulating high-complexity and resource-heavy algorithms. We consider two Integer Linear Programming problems (ILP) that implement a coordinated allocation of radio and computing resources with the objectives of maximizing throughput and maximizing users' satisfaction, respectively. Since solving these highly-complex problems requires a high execution time, we investigate low-complexity alternatives based on machine learning models; more precisely on Recurrent Neural Networks (RNN). These RNN models aim to depict the performance of the ILP problems with a much lower execution time. Our simulation results demonstrate the great ability of RNN models to perform very closely to the ILP problems while being able to reduce the execution time by up to 99.65%.**

***Index Terms*—Cloud-RAN, 5G, Integer-Linear-Programming, Recurrent Neural Networks, Resource Allocation**

I. Introduction

Throughout the last decade, mobile data traffic has experienced massive growth. This growth is driven by the increasing demand for different services such as the Internet of Things, vehicular networks, smart cities, and industry automation, among others. The 5^{th} generation of mobile networks presents new enhanced features to respond to the enormous demand for data. One of these key-enabling features is Cloud Radio Access Networks (Cloud-RAN) [1] [2].

In traditional mobile networks, a base station mainly consists of decentralized modules; a Radio Remote Head (RRH) and a Base Band Unit (BBU). While an RRH is responsible only for radio frequency functions, a BBU is responsible for all the other baseband functions of the PHY/MAC/RRC layers. Cloud-RAN, by contrast, allows for centralizing and virtualizing the baseband processing of the distributed RRHs [1]. Hence, some or all baseband functions of these RRHs will be executed on a powerful, centralized, and shared hardware known as a BBU pool. Unlike traditional RAN, the centralization and joint processing provides higher flexibility, which in turn paves the way for improved interference-mitigation, reduced power consumption, and reduced CAPEX and OPEX [2].

978-1-6654-4005-9/21/$31.00 ©2021 IEEE

It is impossible to exploit Cloud-RAN's full advantages without designing efficient algorithms that can optimally and jointly allocate radio and computing resources to the connected RRHs. Different objectives can drive such algorithms to serve different Quality of Service (QoS) requirements. Depending on the operator strategies, these objectives can range from maximizing throughput to minimizing latency or power consumption.

In our previous works, we have focused on coordinating the allocation of radio and computing resources in Cloud-RAN. We have studied how to manage the limited computing resources when they are insufficient for processing all user data [3]. Two Integer-Linear-Programming (ILP) problems have been formed to achieve optimal resources allocation while considering the objectives of maximizing users' throughput and satisfaction rate, respectively. These ILP problems permit coordination between radio and computing resource schedulers; requiring not only the availability of computing resources at the BBU pool, but also the availability of radio resources. This coordination specifically allows for adjusting the Modulation and Coding Scheme (MCS) index, which controls the modulation order and the coding rate of the data to be transmitted. While, the throughput of a user increases with the MCS index, the time required to process this user's data also increases. In the event of an overload in the BBU pool, resulting in a lack of resources to process all users' date, the coordination allows to reduce users' MCS indexes and hence admitting them in the BBU pool. If coordination is not considered, users would transmit, but their data will not be processed in the BBU pool due to computing resources scarcity; hence the data would be lost and would need to be re-transmitted. By adopting the coordination, the reduction of the MCS index will allow for scheduling these users in the BBU pool, despite the throughput being less than what was demanded initially.

The ILP problems are known to be NP-Hard problems having exponential time-complexity. In fact, it could be impractical for mobile operators to implement them, given that

scheduling decisions should be made in a limited amount of time. Machine Learning (ML) has recently shown many potentials for resource allocation in Mobile Networks [4]. Machine learning, precisely Deep Learning (DL), has demonstrated its ability to provide solutions to various problems. Several papers have investigated the ability of DL to provide low-complexity alternatives to high-complexity optimal algorithms as in [5] and [6]. Not only are DL networks able to closely imitate the performance, they can also have much lower execution time, making them suitable for practical implementation.

In this paper, we investigate the ability of Recurrent Neural Networks, a branch of deep learning, to sub-optimally depict the performance of two ILP-based algorithms. This depiction should provide a practical alternative to these algorithms, where the first aims at maximizing the total throughput, and the second maximizes the total users' satisfaction.

The rest of the paper is organized as follows: Section II surveys the related work. Section III presents the ILP models, followed by the Deep Learning model in section IV. Afterwards, the performance evaluation is presented in section V, and finally, the work is concluded in VI.

II. Related Work

Resource Allocation, namely computing resource allocation, in Cloud-RAN has been considered in various works. [7] considers the required processing time of different uplink BBU functions; with the authors demonstrating the decoding function as the highest consumer of computing resources given that its processing time increases with the increase of either the MCS index or the number of Resource Blocks (RB). Additionally, different processing time prediction models, as a function of the MCS index, have been proposed. These models are based on linear interpolation, polynomial interpolation, and deep-learning, respectively. These models take only the MCS index into consideration. A more generalized processing-time model was introduced in [8], it does not only consider the MCS index as in [7], but also the number of Resource Blocks and the CPU frequency. In the context of Cloud-RAN, the effect of paralleling decoding functions has been considered in [9]; the authors demonstrated the ability of parallelism to reduce the run-time.

In [10], two Integer-Linear-Programming (ILP) problems have been formulated to achieve optimal computing resource allocation, maximizing the throughput and the number of admitted users respectively. These two ILP problems were improved in [3]; in the latter, they gained an extra degree of freedom allowing them to modify the MCS index of users to better adapt the transmission parameters (i.e., MCS index) with the availability of computing resources. Unlike [10] where the performance is measured at the PHY layer, [11] takes into consideration the performance at the MAC layer. The different ILP algorithms have been compared with respect to different MAC layer metrics including goodput, bit-error-rate, and average delay.

The role of Machine learning techniques in mobile networks has attracted a lot of attention. The authors in [5] discussed two applications of deep learning in the physical layer. One of these applications is the approximation of highly-complex functions using a deep neural network. Using Graphical Processor Units (GPU), it could be possible to significantly reduce the function run-time, making it possible to respect the different stringent run-time requirements. Deep Neural Networks have been used in Reinforcement Learning (RL) to learn an approximation of the Q-function. In the same context, [6] targets optimally allocating resources to D2D users and formulates a Mixed Integer Linear Programming Problem (MILP). [6] exploits machine learning to solve the Branch and Bound resource algorithm quicker, which MILP-solvers use to find solutions for MILP problems. In [12] and [13], resource block allocation has been considered for mission-critical services and micro-grid communication, where the goal is to minimize delay. The base stations are agents that interact with the environment by selecting an action given their state and receiving a reward plus their new state. Based on the rewards, the agents will learn the quality of the action they take. Instead of storing the value of actions given each state, which may require a gigantic amount of memory, the authors used a Long-Short-Term-Memory (LSTM) network that approximates the Q-value of each state-action pair and also takes into consideration the effect of actions done in previous time steps.

Inspired by these works, we investigate the ability of RNN-based algorithms to optimally allocate computing resources in Cloud-RAN.

III. Problem Formulation

In our study, we consider a Cloud-RAN system that consists of a centralized BBU-Pool connected to a set of RRHs $\mathcal{R}$. We consider the uplink direction since the most resource-heavy operation in the BBU pool is the decoding function; it has an execution time seven times higher than the encoding function's execution time. Hence, the demand for computing resources overwhelmingly comes from uplink traffic [7]. When the computing resources are insufficient to process the data from all users, a group of users will have their data processed, while others will be dismissed. Hence, operators should adopt the selection strategy they want depending on the service agreements. Excluding some users is unavoidable, and it comes with negative consequences for these excluded users. First, these users will have wasted transmission power. Second, the Hybrid Automatic Repeat Request (HARQ) mechanism will be triggered where each transmission should be acknowledged within 8 ms. According to [9], this leaves only 2ms to process uplink data. Otherwise, the BBU pool, by dismissing users' data, will trigger an HARQ re-transmission. If this transmission is successfully processed, the delay in the system will be increased. If it isn't, the transmission power will be again wasted. This has driven the authors in [3] to propose a coordination scheme between the radio and computing schedulers that allow for adjusting the MCS index of users. Initially, the MCS index is selected depending on the channel condition while ensuring that the error probability

TABLE I
SUMMARY OF THE GENERAL NOTATIONS

Parameters	***Definition***
$\mathcal{R}$	Set of RRHs
$\mathcal{U}_r$	Set of users for each RRH $r \in R$
$\mathcal{M}$	Set of MCS indexes that can be used in the system
$\mathcal{C}$	Set of CPU cores in the shared BBU pool (muti-core data center).
$M_{r,u,max}$	Maximum MCS index user $u \in \mathcal{U}_r$ may use
$t_{r,u,m}$	Data processing time of user $u \in \mathcal{U}_r$ having an MCS index $m \in \mathcal{M}$
$b_{r,u,m}$	Data length (in bits) of user $u \in \mathcal{U}_r$ using an MCS index $m \in \mathcal{M}$ during one TTI
$b_{r,u,max}$	Data length (in bits) of user $u \in \mathcal{U}_r$ using its maximum MCS index $M_{r,u,max}$ during one TTI
d	Processing time deadline
$x^c_{r,u,m}$	Binary variable that assigns the data of user $u \in \mathcal{U}_r$ having an MCS index m to the core $c \in \mathcal{C}$
$s_{r,u}$	The satisfaction ratio of user $u \in \mathcal{U}_r$

should be no more than 0.1 [11]. However, adjusting the MCS index to a lower one allows for decreasing the required processing time at the expense of reducing throughput. This would grant an additional degree of freedom for the operator and can help exploit the computing resources better. Regarding users whom the BBU pool will not admit, they will be notified to not send data so that they do not waste their transmission power.

The system under study consists of: a set of $\mathcal{R}$ RRHs, sets of users $\mathcal{U}_r$ per each RRH r, and a set of homogeneous CPU cores $\mathcal{C}$ available in the BBU pool. The set of possible MCS indexes used for the radio transmission is $\mathcal{M}$. For each RRH r, the coordination policy attributes to user $u \in \mathcal{U}_r$ an MCS index $m \in \mathcal{M}$ lower or equal to the maximum allowed one $M_{r,u,max}$; the one initially chosen by the radio scheduler considering only the radio conditions. We note that the coordination policy does not adjust the MCS index to a higher index to avoid a probability of error higher than 0.1. Based on the chosen index m, user u transmits an amount of data equal to $b_{r,u,m}$ which is determined according to [14]. The latter maps the MCS index and the number of RBs to the transport block size (TBS). The TBS is the payload that is carried over the physical layer. When a user uses, $M_{r,u,max}$ index, $b_{r,u,m}$ is denoted as $b_{r,u,max}$. Additionally, the processing time model from [8] is used to approximate $t_{r,u,m}$, which is the time required for processing user's u data on the BBU pool. A summary of general notations is presented in Table I.

Two coordination policies are considered, the first aims to maximize the total system throughput, and the second aims to maximize the users' satisfaction. We define the user satisfaction ratio as the ratio of the throughput achieved when the user operates using the adjusted MCS index to the maximum throughput obtained when operating using the maximum allowed MCS index (i.e., the one that depends solely on the channel conditions). The two ILP Problems are:

1) *Maximize Total Throughput (MTT)*: One possible policy for the operator is to admit users that can maximize the total system throughput. This comes inline with a 5G objective of maximizing the overall throughput. The following ILP problem ensures this objective:

$$\text{maximize} \quad \sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r}\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} x^c_{r,u,m} b_{r,u,m} \tag{1}$$

$$\text{subject to} \quad x^c_{r,u,m} \in \{0,1\},\ \forall r \in \mathcal{R}, u \in \mathcal{U}_r\ , \quad m \in \mathcal{M}, c \in \mathcal{C} \tag{2}$$

$$\sum_{c\in\mathcal{C}}\sum_{m\in\mathcal{M}} x^c_{r,u,m} \leq 1, \forall r \in \mathcal{R}, u \in \mathcal{U}_r \tag{3}$$

$$x^c_{r,u,m} = 0,\ \forall r \in \mathcal{R}, u \in \mathcal{U}_r, c \in \mathcal{C}, \quad m > M_{r,u,max}, \tag{4}$$

$$\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r}\sum_{m\in\mathcal{M}} x^c_{r,u,m} t_{r,u,m} \leq d, \forall c \in \mathcal{C} \tag{5}$$

$x^c_{r,u,m}$ is a binary decision variable; it is equal to 1 if the data of user $u \in \mathcal{U}_r$ is coded using MCS $m \in \mathcal{M}$ and is processed on CPU core $c \in \mathcal{C}$. It is equal to 0 otherwise. The objective function (1) maximizes the overall system's throughput. MTT solution has the following constraints: (2) guarantees that $x^c_{r,u,m}$ is a binary decision variable; (3) ensures the data of user $u \in \mathcal{U}_r$ are encoded using at most one MCS index m and are processed on at most one CPU core c; (4) ensures a user cannot get an MCS index higher than its maximum allowed one, and (5) ensures that the data, processed on core c, should have been processed before the deadline d. To maximize throughput, MTT tends to prioritize users with high MCS indexes.

2) *Maximize total Users' Satisfaction (MUS)*: This ILP problem aims to maximize the total users' satisfaction defined earlier. In comparison with the previous ILP, only the objective function is modified, while the same constraints are used. The objective function of MUS is:

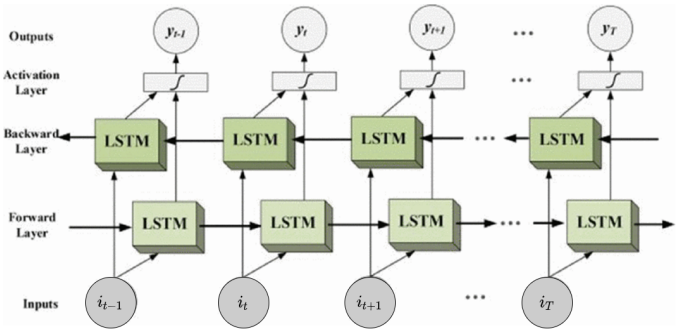
$$\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r}\sum_{m\in\mathcal{M}}\sum_{c\in\mathcal{C}} x^c_{r,u,m} \times \frac{b_{r,u,m}}{b_{r,u,max}} \tag{6}$$

To maximize the total users' satisfaction, MUS tends to assign for each user an MCS index that does not deviate much from the maximum allowed MCS index.

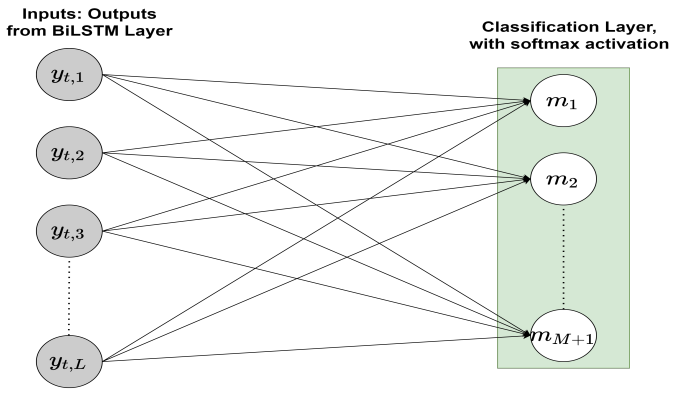
Given that Integer Linear Programming is known to have an NP-Hard complexity, we propose to replace these algorithms with a Recurrent Neural Network (RNN), which should imitate the performance of these algorithms with a much more reduced execution time. In the next section, we discuss the RNN-based algorithms.

IV. RECURRENT NEURAL NETWORK ALGORITHMS

In supervised Machine Learning, a model is given a set of inputs and a set of outputs (i.e., labels) and aims to learn



(a) BiLSTM Architecture [15]



(b) Fully Connected and Classification Layer

Fig. 1. RNN Model Architecture

a mapping between the inputs and these desired outputs. Recurrent Neural Networks are suitable for applications on data with sequential structures. This makes them a great choice for naturally sequential data such as texts, speeches, etc... In such types, the elements in a sequence are more or less dependent on each other. For example, in English language translation use cases, the meaning and the role of each word in the sentence depend on the previous and following elements in the sequence. Hence, such dependency should be taken into consideration by the ML model for it to be able to give accurate results. RNN can process sequential input data of a variable sequence length and can detect dependencies between the elements of the sequence. A very famous layer used in RNN is known as Long-Short-Term-Memory (LSTM). This model is designed to counter the phenomenon of the vanishing gradient problem [4] that slows the learning speed and can hinder the model from improving. While an LSTM can memorize dependencies in one direction, a Bi-Directional LSTM (BiLSTM) uses two LSTM layers instead to traverse the sequence in two directions: a forward direction and a backward direction. Thus, it can capture more dependencies and have a wider scope when making predictions. A BiLSTM-based RNN suits our resource allocation problem presented in the previous section. Given a sequence of users from all RRHs, each user may or may not be admitted. If admitted, the RNN should decide what MCS should be used. Besides, users' selections are dependent because computing resources are limited and shared among them; hence, the decision for a user can affect the decision for another. The RNN model should exploit such dependencies to make decisions that align with the objectives presented in section III. Precisely, this task is handled by the BiLSTM layer.

Fig. 1(a) shows the basic architecture of BiLSTM where $i_{t-1}, i_t, i_{t+1}, ..., i_T$ is the input sequence. Each element in this sequence is a feature vector that carries information about the corresponding item (i.e., a user in our case). This element is input to the two LSTM layers that correspond to the forward and backward directions. The output vectors from each LSTM layer are concatenated and fed into an activation layer; a layer used to increase the ability of the network to learn non-linear dependencies. Finally the output is a vector of length $L = 2H$, where H is a hyperparameter that defines the number of hidden layers in an LSTM. Because of space limitations, the detailed architecture of an LSTM is omitted; it can be found in [15].

In our model, we use a sequence-to-sequence classification. This means that each element (i.e., that represents a user) in the sequence should have an output (i.e., a decision on the MCS index). Each element in the input sequence is in fact a vector of features. The feature vector contains these features: the MCS indexes that can be used, the corresponding throughput and processing time for each MCS index, the number of RBs, and the total demanded throughput, and total demanded processing time. The feature vector is input to the two LSTM layers. The outputs of the two LSTM are input to the activation layer; hyperbolic tangent function, $tanh$, which produces an output vector of L elements. Each element in this vector has a value between -1 and 1. Our proposed RNN model consists of a BiLSTM layer with H=25 hidden layers and L=50; the value of the hyperparameter H is determined by trial and error. These outputs are input to the fully-connected network shown in 1(b). The Fully connected layer has an activation function known as softmax, and it is used for multi-class classification [4]. The classification layer contains $M + 1$ neurons where $M = |\mathcal{M}|$ (i.e., the possible labels of the classifier are the MCS indexes in the system plus another label to indicate a user is dismissed. Among the $M+1$ neurons, the one with the highest activation value gives the label (i.e., the decision).

To train the RRN model, we use a large data set labeled according to the ILP solver results. Our RNN model uses the training dataset to learn how to imitate the performance of the ILP problems as accurately as possible.

V. Performance Evaluation

A. Environment Setup

In our simulation, we consider a BBU pool that has 4 available CPU cores. The number of RRHs connected to the BBU pool varies from 15 to 35. Each RRH operates on 20 MHz bandwidth, thus it has 100 RBs available for allocation. The 100 RBs are allocated to the users of each RRH. Each user is assigned a uniformly random number of RBs between 10 and 30. To assign the maximum allowed MCS

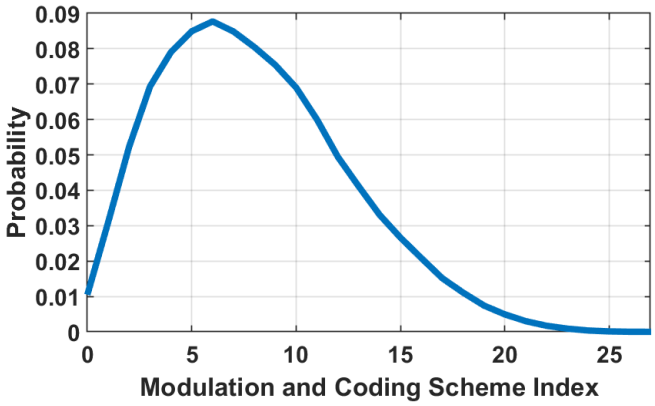


Fig. 2. Probability distribution function of MCS indexes as in [10]

indexes for users, [10] provides an MCS-distribution of data, collected from real users, from which we sample the MCS indexes. The curve of the probability distribution of MCS is provided in Fig.2. The MCS indexes range from 0 to 26. We use the processing time model in [8] to determine the required processing time for users' data. The model provides the processing time as a function of the used MCS, the number of RBs, and the CPU clock speed. We suppose that the 4 CPU cores have a 4GHz clock speed. To determine the throughput, we refer to [14] to find the Transport Block Size (TBS). The TBS is determined as a function of the number of the allocated RBs and the used MCS index, over the Transmission Time interval (TTI) that is equal to 1ms. The simulation is coded using MATLAB, and the ILP problems are solved using CPLEX for MATLAB.

B. Model Training

To train the RNN model to imitate the ILP-problems performance, we generated a dataset by running the simulation 7500 times. For each run, the results of the ILP solvers were added to the data set. Given a group of users from all RRHs, the ILP solver provides the allocation decisions when MTT and MUS problems are solved. To adapt the data to our model, the users in each run were arranged as a sequence, and the allocation results for MTT and MUS were used as labels to train the two RNN models. 67% of the set is used for training, and 33% is used for testing. The accuracy of the training and testing sets are provided in table II.

TABLE II
TRAIN/TEST ACCURACY

Model	MTT	MUS
Training-set Accuracy	*97.27%*	*97.85%*
Testing-set Accuracy	*97.23%*	*97.84%*

C. ILP vs. RNN Comparison

To compare the performance of the ILP problems and RNN models, we are interested in four performance metrics:

1) *Average Throughput*: The average user's throughput.
2) *Admitted Users*: The percentage of admitted users with respect to all users in the system.
3) *Fairness*: Jain's Fairness index [16] is used to compare the fairness among the algorithms. It is defined by:

$$J_I = \frac{\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r} s_{r,u}\right)^2}{\left(N \times \sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{U}_r} s_{r,u}^2\right)} \tag{7}$$

For each user $u \in \mathcal{U}_r$, $s_{r,u}$ is its satisfaction ratio (i.e., the ratio of the attained throughput to the maximum achievable throughput achieved when using the maximum allowed MCS index
4) *Execution Time*: The percentage of reduction of execution time for RNN models with respect to the ILP models.

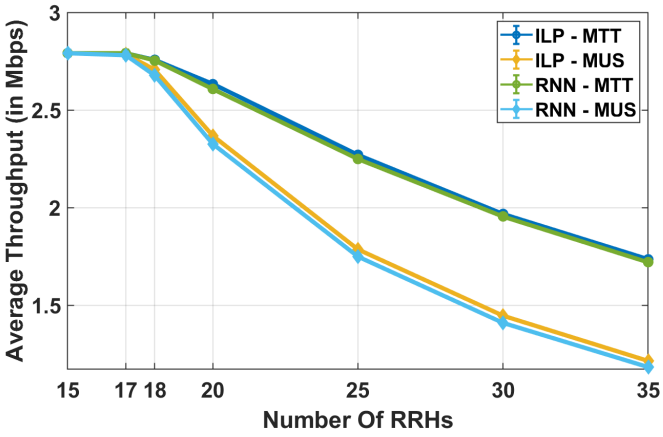
We consider the allocation for only one TTI, and we ran the simulation 1000 times. We provide the 95% confidence intervals.

Fig. 3 shows the performance of the ILP algorithms and their RNN counterparts as a function of the number of RRHs connected to the BBU pool. We note that the BBU becomes fully-loaded when the number of RRHs is equal to 17, after which the algorithms start behaving differently. Before this point, the algorithms perform equally as there would be enough computation resources to admit all users using their maximum MCS index.

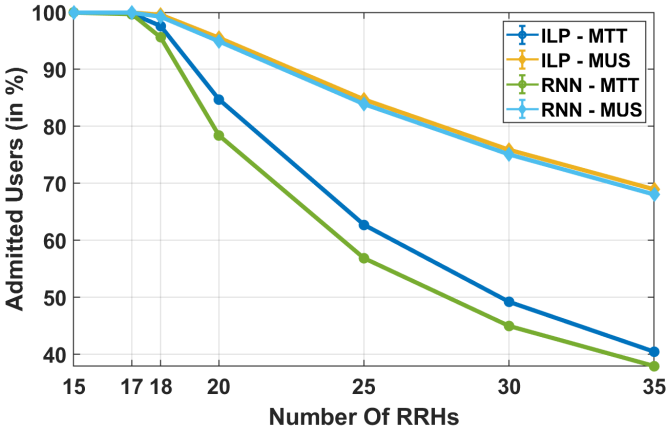
The MTT algorithms achieve higher average throughput per user in comparison to MUS algorithms. However, the latter achieves better performance concerning the metrics of Admitted Users and Fairness. From the real MSC-distribution provided in [10], the bulk of users have MCS indexes between 4 and 10. The users would request less processing time than users with high MCS indexes would. Hence, maximizing the sum of users' satisfaction will lead to favoring the allocation of the bulk of users with low MCS indexes; this justifies the better performance of MUS algorithms with respect to the Admitted Users and Fairness metrics.

More importantly, the figures show that the RNN networks were able to learn MTT and MUS's objectives, respectively. Fig. 3(a) shows that RNN-MTT performs very closely to its ILP counterpart with respect to the average throughput metric. The highest difference between the graphs is minimal and is equal to 0.02 Mbps. Concerning the other metrics in Fig. 3(b) and Fig. 3(c), the margin of difference between ILP-MTT and RNN-MTT increases. RNN-MTT dismisses up to 6.2926% more than the MTT-ILP does and may lose up to 0.062 points in the fairness index, in comparison with MTT-ILP. On the other hand, RNN-MUS captures the performance of ILP-MUS, especially with respect to the fairness metric. The margin of difference between the graphs ILP-MUS and RNN-MUS is not more than 0.0385 Mbps, 0.696%, and 0.0084 concerning the metrics of average throughput, admitted users, and fairness, respectively.

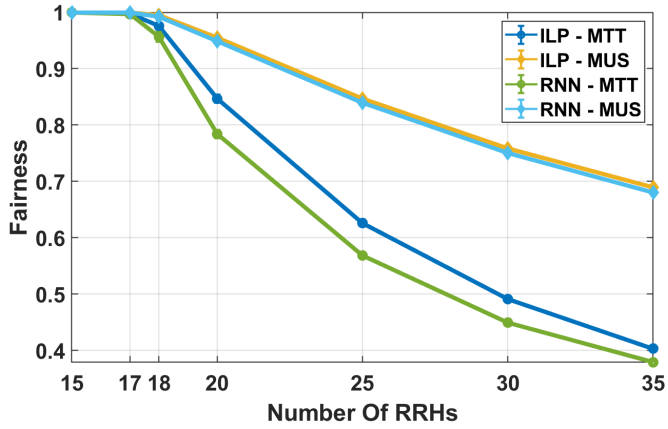
Fig. 3(d) shows the reduction of execution time for the RNN models with respect to the ILP problems. The simulation was done in a MATLAB environment running on an Octa-core



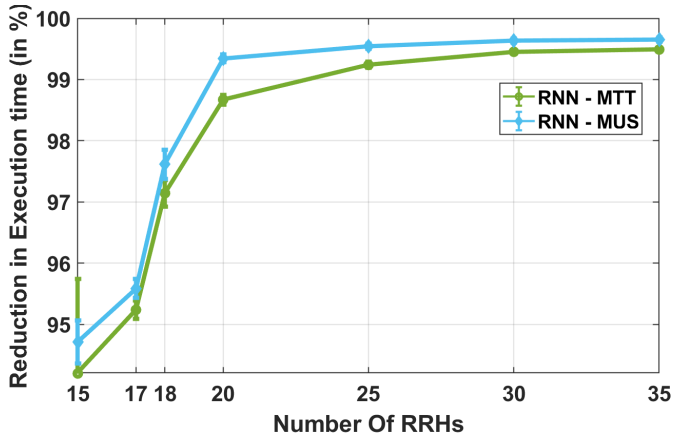
(a) Average user's throughput (in Mbps) as a function of N. of RRHs



(b) Admitted Users (in %) as a function of N. of RRHs



(c) Jain's Fairness Index as a function of N. of RRHs



(d) Reduction in Execution time (in %) with respect to the ILP counterparts as a function of N. of RRHs

Fig. 3. Performance evaluation of the different scheduling solutions

intel CPU core i9-9880H. Compared to ILP-MTT, RNN-MTT reduces the execution time by more than 94% and can reach up to 99.49% of reduction. Similarly, RNN-MUS reduces the execution time by more than 94.7% and can reach up to 99.65% of reduction.

In conclusion, while the performance of RNN models, in comparison to the ILP models, slightly degrades with respect to the throughput, admitted users, and fairness metrics, the significant reduction of the execution time of RNN models makes them more practical for real-time implementation.

VI. Conclusion

In this paper, we have investigated the usage of Recurrent Neural Networks for resource allocation in Cloud-RAN as an alternative to Integer Linear Programming. We considered two ILP problems to allocate computing resources to process users' data and to select their transmission MCS indexes. As solving the NP-Hard ILP problems requires a lot of computing resources, ILP is a bad choice for real-time scheduling. The RNN models have demonstrated their ability to closely depict the performance of the ILP problems with a significant lower execution time. While we have considered in this paper a real-traffic distribution and trained the RNN model based on it, our study will be developed to include different traffic distributions that take into consideration the coexistance of different heterogeneous services in 5G, including enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency (URLLC), and massive-Machine-Type-Communication (mMTC). We will also study the performance of the algorithms at the MAC layer level considering multiple TTIs.

References

[1] C. Mobile, "C-RAN: the road towards green RAN," *White Paper, ver*, vol. 2, pp. 1–10, 2011.

[2] M. A. Habibi, M. Nasimi, B. Han, and H. D. Schotten, "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," *IEEE Access*, vol. 7, pp. 70 371–70 421, 2019.

[3] M. Sharara, S. Hoteit, P. Brown, and V. Veque, "Coordination between radio and computing schedulers in Cloud-RAN," in *17th IFIP/IEEE International Symposium on Integrated Network Management, Bordeaux, France, 2021*.

[4] A. Ly and Y. D. Yao, "A Review of Deep Learning in 5G Research: Channel Coding, Massive MIMO, Multiple Access, Resource Allocation, and Network Security," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 396–408, 2021.

[5] E. Bjornson and P. Giselsson, "Two Applications of Deep Learning in the Physical Layer of Communication Systems [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 134–140, 2020.

[6] M. Lee, G. Yu, and G. Y. Li, "Accelerating Resource Allocation for D2D Communications Using Imitation Learning," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[7] H. Khedher, S. Hoteit, P. Brown, R. Krishnaswamy, W. Diego, and V. Veque, "Processing Time Evaluation and Prediction in Cloud-RAN," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[8] S. Khatibi, K. Shah, and M. Roshdi, "Modelling of Computational Resources for 5G RAN," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018, pp. 1–5.

[9] V. Q. Rodriguez and F. Guillemin, "Towards the deployment of a fully centralized Cloud-RAN architecture," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 1055–1060.

[10] H. Khedher, S. Hoteit, P. Brown, V. Veque, R. Krishnaswamy, W. Diego, and M. Hadji, "Real Traffic-Aware Scheduling of Computing Resources in Cloud-RAN," in *2020 International Conference on Computing, Networking and Communications, ICNC 2020*, 2020, pp. 422–427.

[11] F. Bassi and H. I. Khedher, "HARQ-aware allocation of computing resources in C-RAN," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6.

[12] M. Elsayed and M. Erol-Kantarci, "Deep Reinforcement Learning for Reducing Latency in Mission Critical Services," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[13] M. Elsayed and M. Erol-Kantarci, "Deep Q-Learning for Low-Latency Tactile Applications: Microgrid Communications," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2018, pp. 1–6.

[14] ETSI-LTE, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures 3GPP TS 36.213 v.12.3.0 Rel.12," October 2014.

[15] R. Dhumal Deshmukh and A. Kiwelekar, "Deep Learning Techniques for Part of Speech Tagging by Natural Language Processing," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 76–81.

[16] R. Jain, D. Chiu, and W. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems*. DEC Research Report TR-301, Sep 1984.