Reinforcement Learning based model for Maximizing Operator's Profit in Open-RAN

Mahdi Sharara, Sahar Hoteit, Véronique Vèque

Reinforcement Learning based model for Maximizing Operator's Profit in Open-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*—Open Radio Access Network (O-RAN) is a novel architecture that enables the disaggregation and the virtualization of network components. This would provide new ways to mix and match network components by "opening up" the interfaces between them. O-RAN enables driving down the costs of network deployments and allows the entry of new players into the RAN market. It enables network operators to maximize resource utilization and deliver new network edge services at a lower cost, resulting in higher profits for operators. In this context, we consider a computing resource allocation problem for maximizing the operator's profit. Given that an operator receives subscribers' payments and pays the infrastructure provider's costs, we model the problem using Mixed Integer Linear Programming (MILP). Then, we propose to solve the problem using Reinforcement Learning (RL). Our simulation results demonstrate the ability of the RL agent to increase the operator's profit while reducing the algorithmic complexity of the MILP solver.**

***Index Terms*—Open-RAN, Reinforcement Learning, Resource Allocation, Profit Maximization.**

I. Introduction

With emerging applications such as the metaverse, autonomous vehicles, and factories, among others, the demand for mobile data is massively increasing. To satisfy the growing demands, mobile operators are under colossal pressure; an increased number of base stations is required, which needs huge capital for their installation and management. To cater such needs, Cloud Radio Access Network (C-RAN) has started the movement toward RAN disaggregation by enabling a distributed deployment of RAN functions. It allows for centralizing and virtualizing the baseband processing of multiple base stations. This technology is cost-effective and has enhanced energy efficiency and centralized network architecture [1]. To further break down the disaggregation granularity, Open-RAN (O-RAN) was proposed as a step forward in virtualization and openness. O-RAN permits disaggregating and softwarizing network components. Furthermore, it improves competition allowing more vendors to enter the market while ensuring interoperability [2]. Additionally, O-RAN paves the way for the extensive use of Artificial Intelligence (AI) and Machine Learning (ML) through the non-Real-Time and near-Real-Time Radio Intelligent Controllers (RIC) [3].

In O-RAN, the BS consists of three disaggregated parts; Open Radio Unit (O-RU), Open Distributed Unit (O-DU), and Open Central Unit (O-CU). The O-RU is responsible for the Radio Frequency (RF) and low-PHY functions. The O-DU manages high-PHY, MAC, and RLC layers functions, while the O-CU is responsible for the PDCP and RRC layers. The O-DU and O-CU components can run as virtual machines in an Open-Cloud (O-Cloud), which is a cloud computing platform that is made up of physical infrastructure nodes [4]. The O-Cloud would be owned by an independent infrastructure provider (InP), and the mobile operator would be just one of the customers of this InP [5]. As the computing infrastructure in O-RAN is shared, mobile operators must request computing resources to meet the demands of their users while maximizing their own profits.

In this work, we consider a scenario in which the mobile operator should request computing resources to process its users' frames while considering the various service types with different quality of service (QoS) and deadline requirements. We consider the two types of services, the enhanced Mobile BroadBand (eMBB) and the Ultra Reliable Low Latency (URLLC) service. The main driving motive for operators is maximizing revenues and minimizing expenses. Operators spend resources on capital and operational expenditures, including telecom equipment and resources they own or rent, energy, maintenance, etc. On the other hand, they profit from selling their services to customers. To maximize the operator's profit, we formulate a Mixed Integer Linear Programming (MILP) problem that efficiently allocates computing resources to users' frames while considering different services' priorities. However, solving the MILP problem is highly complex. Hence, we resort to Reinforcement Learning (RL) techniques, particularly the Policy-Gradient-based model, to allocate computing resources to users.

RL is widely used because it can autonomously solve problems and adapt to changing environments. In RL, an agent, in a given state or observation, interacts with its environment and receives rewards or penalties. Intending to maximize rewards, the agent selects actions that maximize its reward. RL was used in [6] to satisfy the different communication and computing users' demands. Q-Learning is used in [7] to allocate radio resource blocks (RB) and power. Moreover, in [8], Deep Deterministic Policy Gradient (DDPG) is used to allocate RBs and power, while a Double Deep Q-Network-learns the optimal RAN slicing strategies. Additionally, RL was used in [9] to select the functional split that minimizes the routing and computational cost. In [10], computing resources are allocated using Integer Linear Programming (ILP) to maximize throughput and fairness, while [11] proposes a low-complexity ML-based alternative for these two objectives. In [12], the radio and computing resources allocation aims to minimize power consumption, reducing operators' OPEX.

In contrast to all these works, we consider in this paper

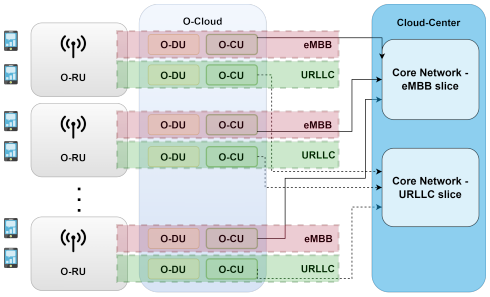


Fig. 1: System Architecture

the computing resource allocation problem that maximizes the mobile operator's profits. Additionally, we propose a low-complexity policy-gradient-based RL model to solve the problem, and we compare its performance with the optimal MILP solution.

The rest of the paper is organized as follows: The MILP problem is formulated in Section II, and the RL model is presented in Section III. The simulation results are discussed in Section IV, and our work is concluded in Section V.

II. Context and Problem Formulation

We consider a set $\mathcal{R}$ of Open Radio Units (O-RU). The set of users connected to an O-RU $r \in \mathcal{R}$ is denoted by $\mathcal{U}_r$; it combines the sets of both eMBB users $\mathcal{U}_r^E$ and URLLC users $\mathcal{U}_r^U$. As each service type has different delay tolerance, the processing deadlines for eMBB users and URLLC users are expressed by d^E and d^U, respectively, where $d^E > d^U$. For each O-RU, there exists two O-DUs (Open Distributed Unit) serving each service type (i.e., eMBB and URLLC), and each O-DU is connected to an O-CU (Open Central Unit). The O-DUs and O-CUs run as virtual machines in the Open Cloud (O-Cloud) and have to share the computing resources available at this O-Cloud, which is not owned by the operator. [13], [14] show that achieving high reliability and low latency requires deploying redundant hardware. However, we are only targeting the case where computing resources are not enough either due to bad provisioning or because of some financial restrictions for the operator. Fig. 1 shows the system architecture. We consider a set of CPUs $\mathcal{C}$. These CPUs are owned by an infrastructure provider (InP), and the operator must share the available computing resources with different users (i.e., other operators). For each CPU, $k \in \{1, 2\}$ defines the priority level of a frame. Frames processed with priority k=1 will force the CPU to preempt what it currently processes and process these frames immediately. When k=2, the CPU could process the frames after fully processing the ongoing tasks. For each CPU core $c \in \mathcal{C}$, $F^{c,2}$ is the amount of the available computing resources over the next 2ms which is the eMBB processing deadline ($F^{c,2} <= d^E$). $F^{c,1}$ is the portion of $F^{c,2}$ which could be immediately used for processing user frames without waiting for the current tasks to finish (i.e., priority k=1). Each user pays the operator $p_{user}^{r,u}$ unit of money per bit. The operator pays for the InP when it processes the users' frames an amount equal to $p_{op}^{c,k}$ unit of money per second, depending on which CPU c and priority level k are used. We define $b^{r,u}$ as the number of bits carried by the frame of user $u \in \mathcal{U}_r$, while $e^{r,u}$ is the amount of time required to process the frame. We note that the number of bits and the processing times depend on the used Modulation and Coding Scheme (MCS) and the number of RBs [10]. To maximize the operator's profits, we formulate the following MILP problem. The problem uses the binary variable $x_{c,k}^{r,u}$ which is equal to 1 if user $u \in \mathcal{U}_r$ is processed on CPU $c \in \mathcal{C}$ with priority k:

$$\max \quad \sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} \sum_{c\in\mathcal{C}} \sum_{k\in\{1,2\}} x_{c,k}^{r,u}(b^{r,u}p_{user}^{r,u} - e^{r,u}p_{op}^{c,k}) \tag{1}$$

$$\text{sub. to} \quad x_{c,k}^{r,u} \in \{0,1\}, \forall r \in \mathcal{R}, u \in \mathcal{U}_r, c \in \mathcal{C}, k \in \{1,2\} \tag{2}$$

$$\sum_{c\in\mathcal{C}} \sum_{k\in\{1,2\}} x_{c,k}^{r,u} \leq 1, \ \forall r \in \mathcal{R}, u \in \mathcal{U}_r, \tag{3}$$

$$\sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} \sum_{k\in\{1,2\}} x_{c,k}^{r,u} e^{r,u} \leq F^{c,2}, \forall c \in \mathcal{C} \tag{4}$$

$$\sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} x_{c,1}^{r,u} e^{r,u} \leq F^{c,1}, \forall c \in \mathcal{C} \tag{5}$$

$$\sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r^U} x_{c,1}^{r,u} e^{r,u} \leq min(F^{c,1}, d^U), \forall c \in \mathcal{C} \tag{6}$$

$$\sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r^U} x_{c,2}^{r,u} e^{r,u} \leq v_c, c \in \mathcal{C} \tag{7}$$

$$v_c \in \mathbb{R}_{\geq 0}, \ \forall c \in \mathcal{C} \tag{8}$$

$$v_c \geq F^{c,2} - (d^E - d^U) - \sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} x_{c,1}^{r,u} e^{r,u}, \forall c \in \mathcal{C} \tag{9}$$

$$v_c <= M z_c, \forall c \in \mathcal{C} \tag{10}$$

$$v_c \leq F^{c,2} - (d^E - d^U) - \sum_{r\in\mathcal{R}} \sum_{u\in\mathcal{U}_r} x_{c,1}^{r,u} e^{r,u} + M(1 - z_c), \forall c \in \mathcal{C} \tag{11}$$

$$z_c \in \{0,1\}, \ \forall c \in \mathcal{C} \tag{12}$$

The objective (1) maximizes the total operator profit. (2) ensures the binary nature of the integer variable. (3) ensures that a frame is processed once. (4) ensures that a CPU is able to process all the assigned frames, while (5) sets the limit on the total frames processed with priority k=1. The total time of processed URLLC frames with priority k=1 can not exceed the minimum of the available CPU time and the URLLC deadline as (6) shows. On the other hand, URLLC frames can only be processed with priority k=2 if there are enough resources before the deadline, as (7) mandates. The difference between the eMBB and URLLC deadline indicates the amount of time that can only be used to process eMBB users. Hence, the only way to process URLLC users is if the remaining available resources are bigger than the difference in deadlines. In this case, the amount of available resources is tracked by the auxiliary continuous variable v_c defined in (8). However, this difference could be negative; hence v_c is equal to the value explained above if it's non-negative, and it is equal to zero otherwise. (8),(9),(10),(11) together enforce this condition. They use the auxiliary binary variable z_c defined in (12). M is the big-M notation. This MILP problem is NP-Hard [15]; thus, we propose to solve it using Reinforcement Learning techniques.

III. Reinforcement Learning Model

We consider a policy gradient-based RL model where each Transmission Time Interval (TTI) is an episode composed of steps. Resources are scheduled to users for a duration of one

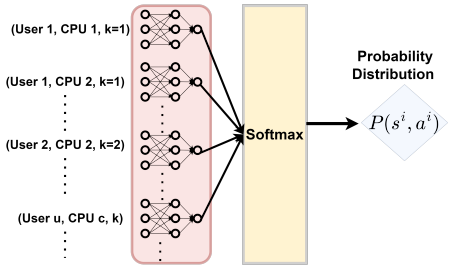


Fig. 2: Neural Network Architecture

TTI. Each (user, CPU, priority) tuple is represented by $\delta_{r,u}^{i,c,k}$. In policy gradient algorithms, the agent learns the optimal probability distribution of actions that maximizes its rewards. We use a flexible Neural Network that scales with the number of users and CPUs. Each $\delta_{r,u}^{i,c,k}$ is fed into the same neural network and outputs a value. Hyperbolic tangent tanh is used as an activation function in the neural network, given it scales its values between -1 and 1, which could aid convergence. The outputted values from all $\delta_{r,u}^{i,c,k}$ are fed into a softmax function that outputs a probability distribution. The agent selects a tuple according to this probability distribution. At each step, a tuple is selected, assigning a user to a CPU and a priority level. It would have been possible to use a static NN architecture with a defined max number of users. However, the non-occupied tuples should be zero-padded. This will make the size of the NN huge and penalize the convergence time. As in [16] and [17], we use the dynamic architecture shown in Fig.2. The Markov Decision Process (MDP) model is defined as follows:

A. State

In each episode (i.e., TTI), one user will be selected in step i. The state includes the execution time $e^{r,u}$, number of bits $b^{r,u}$, $service^{r,u}$ that indicates if the service is eMBB or URLLC, the payment of the user $b^{r,u}p_{user}^{r,u}$, the cost of processing $e^{r,u}p_{op}^{c,k}$, and the profit of the operator (i.e., payment minus cost). Then:

$$\delta_{r,u}^{i,c,k} = \left\{ e^{r,u}, b^{r,u}, service^{r,u}, b^{r,u}p_{user}^{r,u}, e^{r,u}p_{op}^{c,k}, (b^{r,u}p_{user}^{r,u} - e^{r,u}p_{op}^{c,k}) \right\}$$

Given that $F^{c,k}$ is the available computing resources of CPU c at step i and that a^j is the action at step j, the state at step i of a TTI is defined as

$$s^i = \left\{ \delta_{r,u}^{i,c,k} : \forall r \in \mathcal{R}, u \in \mathcal{U}_r, c \in \mathcal{C}, k \in \{1,2\}, u \notin a^j, \forall j < i, e^{r,u} \leq F^{c,k} \right\}$$

In each step, users have two priority choices and a set of CPUs to be assigned to. Once selected, the user can not be reselected and is omitted from the state and action space. Hence, the user and all the corresponding CPUs c and priorities k tuples are removed from the state representation. Moreover, the state only includes feasible choices; if there are no more resources to assign a user to a specific CPU and a priority, this tuple will be excluded from the state.

B. Action

At each step i of one TTI, the action a^i assigns a user $u \in U_r, r \in \mathcal{R}$ to CPU c and priority k, hence the action is the tuple:

$$a^i = (u, c, k), u \in \mathcal{U}_r, r \in \mathcal{R}, c \in \mathcal{C}, k \in \{1,2\}$$

C. Reward

The goal is to optimize the profit. The reward of action $a^i = (u, c, k)$ at step i being in state s^i is:

$$r^i(s^i, a^i) = \frac{2}{\pi} arctan(b^{r,u}p_{user}^{r,u} - e^{r,u}p_{op}^{c,k})$$

Using $arctan$ in the reward function allows us to finally scale the reward to be between -1 and 1, which aids convergence.

D. RL algorithm

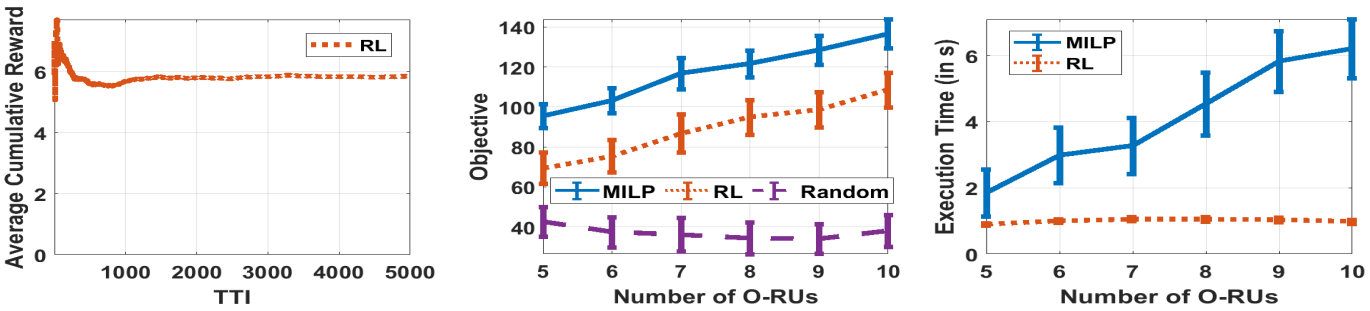
The RL algorithm is based on REINFORCE algorithm with a baseline [18]. The weights θ of the Neural Network are initialized. For each TTI, the $\delta_{r,u}^{i,c,k}$ and state s^i should be initialized. Then the agent executes action a^i, gets r^i, and moves to s^{i+1}. This will be repeated until the terminal state is reached (i.e., no more resources or no users to be allocated). The weights θ should be updated using the following formula:

$$\theta \leftarrow \theta + \alpha v^i \nabla log(P(s^i, a^i))$$

where α is the learning rate, $P(s^i, a^i)$ is the probability value yielded by the NN, and v^i is the discounted reward, normalized by subtracting the mean of the rewards in an episode and divided by the standard deviation of the rewards.

IV. Performance Evaluation

To test our simulation, we consider a scenario where the number of O-RUs competing for computing resources in the O-Cloud varies from 5 to 10. In each O-RU, 90 RBs are reserved for eMBB users, while 10 RBs for URLLC users. The number of RBs of an eMBB user follows a uniform distribution between 20 and 40, while it is between 1 to 5 for a URLLC user. The eMBB and URLLC deadlines d^E and d^U are 2ms and 0.25 ms [17], respectively. We use the real traffic distribution in [10] to sample the MCS index for each user. Users are resampled at each TTI. To calculate the number of transmitted bits, the 3GPP specifications [19] provide the Transport Block Size (TBS) as a function of the number of RBs and MCS. We use a processing time model as a function of the MCS, the number of RBs, and the CPU frequency [20]. We set the CPU frequency to 2.6 GHz. The total available CPU time $F^{c,2}$ is a uniform random variable between 0.001ms and 2ms. The percentage of the amount of total computing resources $F^{c,2}$ available for prioritized processing is a uniform random variable between 0 and 100% (i.e., this percentage indicates how much $F^{c,1}$ is out of $F^{c,2}$, where $F^{c,1}$ is a partial amount of $F^{c,2}$). The cost of the non-prioritized processing per CPU per μs is a uniform random variable between 0.01 and 0.05 monetary units. The cost of prioritized processing, in which ongoing tasks are preempted to immediately schedule frames with priority k=1, is uniformly random, between 1 to 3 times more than the cost of the non-prioritized one. A user pays a uniformly distributed value from 0.2 to 3.6 monetary units per Kilobit. The reason for this extreme randomness is that we wanted to ensure the RL agent could adapt to diverse scenarios. For the RL agent, 8 neurons represent the input to the NN, which has 1 hidden layer with 10 neurons, and an output layer with 1 neuron. The learning rate for the agent



(a) Average Cumulative Reward (b) Objective value in monetary units (c) Execution time of algorithms

Fig. 3: Performance Figures

is 0.01, while the rewards discount factor is 0.999. We use MATLAB for the simulation, the Deep Learning toolbox for training the RL agent, and CPLEX to solve the MILP problem. In the graphs, we plot the 95% confidence intervals.

A. Simulation Results

Fig.3a shows the average cumulative reward. The RL agent converges after 1250 TTI. The agent is initially trained by varying the number of O-RUs, which varies the number of users and the total demand, from 5 to 10. This justifies why we initially have some fluctuations in the curve because the profit is affected by the number of existing users, strongly varying the total reward from one TTI to another. Fig. 3b compares the value of the objective function when the MILP solver is used versus when the RL model is used. Additionally, an algorithm named "Random" is used for benchmarking; it randomly allocates users to a CPU with random priority. Results show that the MILP can perfectly pick the user-CPU-priority tuples that maximize the operator's profit. The RL model achieves at least 73% of the optimal MILP solution and reaches 80% when the number of O-RUs is 10. In comparison, the random allocation algorithm does not exceed 44% of the MILP solution and drops to 26% of the optimal solution. Fig. 3c shows the execution time of the MILP solver versus the RL algorithm. The RL can reduce the execution time by up to 85% compared with the MILP. We note that the study was done on a core i9-9880H, and the GPU used for RL is Nvidia Quadro P620. Given that scheduling decisions are made every TTI equal to 1ms, running the MILP problem in a real scenario is impossible. The RL model has been tested on the mentioned GPU, which is not optimized for Machine Learning computation. However, the RL model execution speed could be accelerated by using powerful GPUs that can parallelize the solution and output results in the required amount of time. Additionally, using MATLAB will add overhead, increasing the execution time.

B. Improvement Perspectives

The MILP algorithm can perfectly pick the optimal solution at the cost of high complexity. In contrast, random allocation is a simple algorithm that curbs potential profits. The RL model presents a trade-off between lower complexity and optimality. However, there are a lot of potential paths for improvement. In addition to improving the hardware, as stated above, tweaking the Neural Network architecture and the activation function could have potential benefits in improving the performance of the RL model. Moreover, the MILP problem and the RL models assumed perfect processing time and cost knowledge. Even the infrastructure provider may refuse to provide detailed information regarding the availability of resources for operational and security reasons. If the information is missing, it is impossible to run the MILP problem. However, it would be possible to tune the state representation of the RL model to exclude the missing information. The RL would still learn the policy by receiving rewards from the environment. This is an important advantage for the RL versus the MILP problem. Moreover, we have heavily randomized the cost models to account for diverse scenarios. The payment and cost model would be more deterministic in the real world. This would allow the RL model to converge faster and probably improves its performance, given that it will have less to learn.

V. Conclusion

In this paper, we have considered computing resource allocation for maximizing the operator's profit in O-RAN. O-RAN enables the virtualization of RAN components. Instead of owning infrastructure that is not always needed and would increase the CAPEX and OPEX, the operator can request computing resources from InPs when required. However, this mechanism must be profitable for the operator. We have formulated a Mixed Integer Linear Programming Problem that allocates computing resources to process users' frames to maximize the operator's profits. We have also proposed an RL model with the same objective but lower complexity. The simulation results highlight the ability of the RL model to score high profits while having lower complexity. In the future, we plan to test this model in a real O-RAN-compliant system to validate its performance and adapt it to the different service types in 6G. Also, we intend to investigate various Neural Network architectures in order to bring the RL's performance closer to that of the MILP, and explore different parameters in the state representation to help the RL agent better learn and converge. We also intend to adapt the model to multi-operator scenarios in which operators share infrastructure, compete, and collaborate with one another.

Acknowledgment

This work was funded by the ANR HEIDIS (https://heidis.roc.cnam.fr/; ANR-21-CE25-0019) project.

REFERENCES

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

[2] A. K. U and G. Gundu Hallur, "Economic and Technical Implications of Implementation of OpenRAN by "RAKUTEN MOBILE"," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, 2022, pp. 959–964.

[3] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[4] (2019) Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN.

[5] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *CoRR*, vol. abs/2202.01032, 2022. [Online]. Available: https://arxiv.org/abs/2202.01032

[6] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement Learning for Dynamic Resource Optimization in 5G Radio Access Network Slicing," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.

[7] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Radio Resource Allocation in 5G for URLLC and eMBB Users," in *2019 IEEE 2nd 5G World Forum (5GWF)*, 2019, pp. 590–595.

[8] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent Radio Access Network Slicing for Service Provisioning in 6G: A Hierarchical Deep Reinforcement Learning Approach," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 6063–6078, 2021.

[9] F. W. Murti, S. Ali, and M. Latva-aho, "Deep Reinforcement Based Optimization of Function Splitting in Virtualized Radio Access Networks," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.

[10] M. Sharara, S. Hoteit, P. Brown, and V. Vèque, "Coordination between Radio and Computing Schedulers in Cloud-RAN," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021.

[11] M. Sharara, S. Hoteit, and V. Vèque, "A Recurrent Neural Network Based Approach for Coordinating Radio and Computing Resources Allocation in Cloud-RAN," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021.

[12] M. Sharara, S. Hoteit, V. Vèque, and F. Bassi, "Minimizing Power Consumption by Joint Radio and Computing Resource Allocation in Cloud-RAN," in *IEEE Symposium on Computers and Communications (ISCC 2022)*, Rhodes, Greece, Jun. 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03737135

[13] J. Varga, A. Hilt, C. Rotter, and G. Járó, "Providing ultra-reliable low latency services for 5g with unattended datacenters," in *2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)*, 2018, pp. 1–4.

[14] J. Varga, A. Hilt, J. Bíró, C. Rotter, and G. Járó, "Reducing operational costs of ultra-reliable low latency services in 5G," *Infocommunications Journal*, vol. X, pp. 37–45, Dec. 2018.

[15] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th ed. Springer Publishing Company, Incorporated, 2012.

[16] J. S. Shekhawat, R. Agrawal, K. G. Shenoy, and R. Shashidhara, "A Reinforcement Learning Framework for QoS-Driven Radio Resource Scheduler," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.

[17] M. Sharara, T. Pamuklu, S. Hoteit, V. Vèque, and M. Erol-Kantarci, "Policy-Gradient-Based Reinforcement Learning for Computing Resources Allocation in O-RAN," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, Paris, France, Nov. 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03791024

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[19] (2018, October) 5G; NR; Physical layer procedures for data, ETSI TS 138 214 V15.3.0.

[20] 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.