AegisRAN: A Fair and Energy-Efficient Computing Resource Allocation Framework for vRANs

Ethan Sanchez Hidalgo, Jose A. Ayala-Romero, *Member, IEEE*, Josep Xavier Salvat Lozano, *Member, IEEE*, Andres Garcia-Saavedra, and Xavier Costa Perez, *Senior Member, IEEE*

***Abstract*—The virtualization of Radio Access Networks (vRAN) is rapidly becoming a reality, driven by the increasing need for flexible, scalable, and cost-effective mobile network solutions. To mitigate energy efficiency concerns in vRAN deployments, two approaches are gaining attention: (*i*) sharing computing infrastructure among multiple virtualized base stations (vBSs); and (*ii*) relying upon general-purpose, low-cost CPUs. However, effectively realizing these approaches poses several challenges. In this paper, we first conduct a comprehensive experimental campaign on a vRAN platform to characterize the impact of computing and radio resource allocation on energy consumption and performance across various network contexts. This analysis reveals several key issues. First, determining the optimal allocation of computing resources is difficult because it depends on the context of each vBS (e.g., traffic load, channel quality) in a non-trivial and non-linear manner. Second, suboptimal resource assignment can lead to increased energy consumption or, even worse, degradation of users' Quality of Service. Third, the high dimensionality of the solution space hinders the effectiveness of traditional optimization or learning methods. To tackle these challenges, we propose AegisRAN, a framework for optimizing computing resource allocation in vRAN. AegisRAN addresses the dual objective of minimizing energy consumption while maintaining high system reliability. Moreover, when computing resources are overbooked, our solution ensures a fair resource partition based on vBS performance. AegisRAN leverages a discrete soft actor-critic algorithm combined with several techniques, including multi-step decision-making, action masking, digital twin-based training, and a tailored reward signal that mitigates feedback sparsity. Our evaluations demonstrate that AegisRAN achieves near-optimal performance and offers high flexibility across diverse network contexts and varying numbers of vBSs, with up to 25% improvement in energy savings compared to baseline solutions in medium-scale scenarios.**

***Index Terms*—5G, virtual RAN, reinforcement learning, digital twin.**

Received 16 January 2025; revised 11 April 2025; accepted 15 April 2025. Date of publication 30 April 2025; date of current version 3 September 2025. This work was supported in part by the European Commission under Grant 101139270 (ORIGAMI) and Grant SNS-JU-101097083 (BeGREEN), in part by NextGeneration EU through UNICO I+D under Grant TSI-063000-2021 (OPEN6G), and in part by the CERCA Programme. Recommended for acceptance by J. Kang. *(Corresponding author: Jose A. Ayala-Romero.)*

Ethan Sanchez Hidalgo is with i2CAT Foundation, 08034 Barcelona, Spain (e-mail: ethan.sanchez@i2cat.net).

Jose A. Ayala-Romero, Josep Xavier Salvat Lozano, and Andres Garcia-Saavedra are with the NEC Laboratories Europe GmbH, 69115 Heidelberg, Germany (e-mail: jose.ayala@neclab.eu; josep.xavier.salvat@neclab.eu; andres.garcia.saavedra@neclab.eu).

Xavier Costa Perez is with the NEC Laboratories Europe GmbH, 69115 Heidelberg, Germany, and also with i2CAT Foundation and ICREA, 08034 Barcelona, Spain (e-mail: xavier.costa@ieee.org).

Digital Object Identifier 10.1109/TMC.2025.3564116

I. Introduction

RADIO Access Network virtualization (vRAN), driven by the O-RAN Alliance [1], enables transitioning RANs from traditional hardwired appliances using dedicated hardware to deploying them into commercial off-the-shelf (COTS) servers [2], [3]. vRANs promise to import the advantages of Network Function Virtualization (NFV), opening the door to deploying communication services in public clouds or edge clusters and providing enhanced management flexibility [4], resource multiplexing [5], and adaptability for system upgrades [6]. Recently, major vendors, cloud providers and operators began collaborating to evaluate the feasibility of offering 5G communication services deployed in public cloud infrastructure [7], [8], [9], [10].

As vRAN workloads must meet very stringent physical layer (L1) processing deadlines with high reliability (99.999% probability) [11], operators often rely on dedicated hardware accelerators (HAs) co-located with shared computing platforms. This approach helps bridge the gap between the demanding processing requirements of 5G and the limitations of COTS hardware. However, relying solely on HAs requires significant capital investments, poses a substantial energy toll, and may compromise the flexibility of virtualizing the RANs.

Recently, industry and academia have turned their attention to using general-purpose CPUs as a complement to HAs [4], [12]. This approach can mitigate the high energy consumption associated with HAs, as CPUs offer good performance for small to medium vRAN workloads [4]. However, CPUs also introduce challenges. First, virtualized workloads compete for shared resources as computing time and memory increasing the unpredictability of processing latencies (the "noisy neighbor problem") [13], [14]. This is particularly detrimental in vRANs due to their inelastic requirements and the potential for connectivity loss if computing deadlines are missed [15]. Second, the computing requirements of vRAN workloads are influenced by factors like load and signal-to-noise ratio (SNR), making short-term resource prediction challenging. This can lead to resource over-provisioning and low utilization. Finally, operators must handle highly variable traffic patterns, characterized by peaks during certain times of day, user mobility, and diverse Quality of Service (QoS) requirements. This variability is confirmed by our experiments conducted in Heidelberg, Germany, in March 2024, which monitor the workloads processed by a cell during 72 hours as depicted in Fig. 1. These insights are also confirmed in the literature [16], [17].

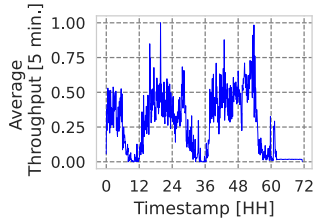


Fig. 1. Relative load of a cell during 3 days.

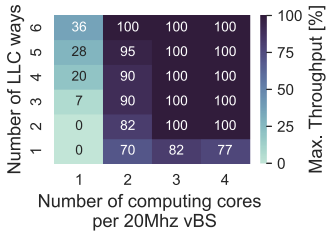


Fig. 2. Capacity of a proof-of-concept vRAN.

The literature offers some solutions to these challenges. On the one hand, approaches like Agora [12] and Nuberu [15] optimize the pipeline of signal processing tasks in virtual BSs, enabling vRANs to run on CPUs. On the other hand, solutions like vrAIn [18], AIRIC [13], and MemorAI [14] focus on optimizing the allocation of computing resources to deployed vRAN instances. However, these existing solutions fail to address the interplay between CPU cores and Last-Level Cache (LLC) memory. To highlight this, we present preliminary results from experiments on a Proof-of-Concept (PoC) vRAN system comprised of instances of a full-fledged 3GPP Rel.15 compliant vBS implemented with srsRAN.[1] Using containers, we deployed a vBS instance with 20 MHz bandwidth, varying the number of computing cores and LLC memory allocation on an Intel E5-2650 v4 @ 2.20 GHz CPU within a shared computing platform. Section III presents the details of our PoC.

Fig. 2 depicts the maximum relative throughput achieved (i.e., *capacity*) for different allocations of CPU resources (cores) and cache memory resources (LLC cache ways) under high SNR conditions (>20 dB) in both downlink and uplink. These preliminary results reveal that simply increasing the allocation of CPU cores does not always improve the maximum achievable network throughput unless LLC cache resources are also optimized. On the contrary, when CPU core allocation is scarce, over-provisioning LLC cache can increase capacity. Fig. 3 shows the energy savings compared to the worst-case scenario for different CPU core and LLC cache way allocations.

We observe that while more CPU cores increase energy consumption, the allocation of more LLC ways increases the energy efficiency, in contrast to the trend shown in Fig. 2. This behavior renders a non-trivial resource allocation, which is exacerbated in real-world scenarios, where many vBS share a common set of computing resources. In those cases, we may find situations where the computing resources are insufficient to ensure the full capacity of the deployed vBS. Thus, we need to devise fair

[1] [Online]. Available: https://www.srsran.com/

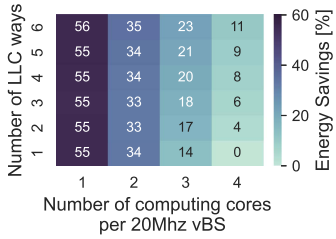


Fig. 3. Energy Savings of a proof-of-concept vRAN.

resource allocation so that the performance losses are distributed fairly.

Moreover, the optimal allocation of both resource types depends on the user demands and user MCS (which depends on the Signal-to-Noise-Ratio) [4], [18], which are very volatile as shown earlier in Fig. 1. Therefore, depending on the network load and the number of vBSs sharing the infrastructure, our goal is to optimize resource allocation in two ways: (i) reduce allocated computing resources to increase energy savings while meeting user demands; and (ii) optimize resource allocation across multiple vBSs sharing the same infrastructure to fairly distribute throughput losses when resources are scarce.

To address these challenges, we present AegisRAN, a vRAN computing resource allocation framework that dynamically allocates computing cores and LLC-cache ways *jointly*. Our framework adapts computing resources to traffic volatility (i.e., channel quality changes, type of traffic change etc.), ensuring reliability by adjusting the allocation of radio resources. AegisRAN identifies opportunities for reducing energy consumption during low-load periods and maintains reliability even under overbooked computing resources by deploying multiple vBS instances on the same hardware and fairly adapting their allocation of radio resources. AegisRAN leverages a discrete soft actor-critic (SAC) reinforcement learning technique enhanced with a multi-step decision-making design, digital twin-enhanced training, action masking, and a crafted reward signal that captures all the challenging aspects of the problem and mitigates the sparsity in the feedback. As a result, AegisRAN provides close-to-optimal performance and high flexibility, adapting seamlessly to scenarios with a wide range of contexts and vRAN instances. Our evaluations show up to 25% improvement on energy saving compared to baseline solutions in medium-scale scenarios.

II. Background

We first briefly provide background information required to understand the AegisRAN framework solution, including a premier on 5G RAN (Section II-A), RAN virtualization and Open RAN (Section II-B), and general-purpose virtualization platforms (Section II-C).

A. A Premier on 5G RAN

The Radio Access Network (RAN) enable cellular communication via a 5G wireless interface called New Radio (NR). Fig. 4 depicts the main components of 5G RAN architecture. The RAN is splitted into three main functions [19]: (i) a Radio

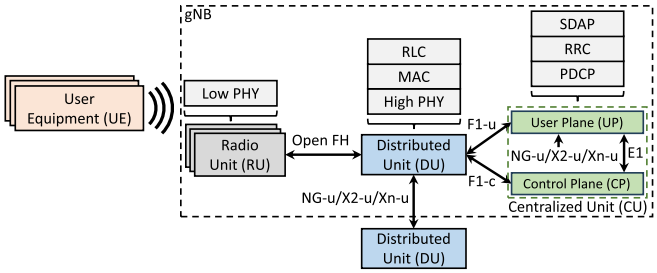


Fig. 4. 5G-NR RAN Architecture.

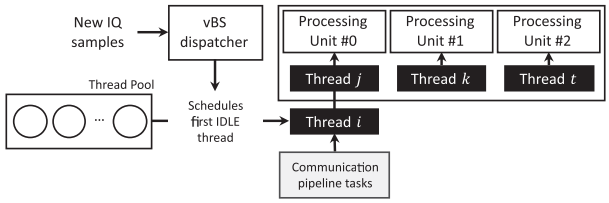


Fig. 5. RAN virtualization architecture.

Unit (RU), (ii) a Distributed Unit (DU), and (iii) a Central Unit (CU). The RU hosts low-level PHY functions (also known as L1), including the iFFT/FFT, the AD/DA converters, the beamforming and other RF functions such as amplification or sampling. The DU hosts the high PHY functions which includes the encoding/decoding, the MAC layer and the RLC layer. Finally, the CU, which is splited into two components for the user plane (UP) and control plane (CP) functions, supports the higher layer protocols such as the PDCP layer, the SDAP layer and the RRC layer. The three functions constitute a gNodeB (gNB) or Base Station (BS).

In 5G NR, radio resources are structured as a two-dimensional plane where the frequency domain is divided into Resource Blocks (RBs) and and the time domain is divided into OFDM symbols. RBs are the smallest resource units allocated to users. The total number of RBs of a BS depends on the available bandwidth [20]. User data is encapsulated in Transport Blocks (TBs), which are then mapped to modulated symbols across one or more RBs. The number of bits carried by one symbol in one RB depends on the Signal-to-Noise Ratio (SNR) and the Modulation and Coding Scheme (MCS), which is often selected dynamically to balance throughput and reliability. Higher-order modulation schemes allow for higher data rates but require better SNR conditions. Conversely, lower SNR conditions necessitate more robust but slower modulation schemes. Every time slot (14 OFDM symbols), the scheduling algorithm at the BS's MAC layer assigns RBs and MCS to users based on their channel conditions (SNR), user requirements, and Quality of Service (QoS) needs.

B. RAN Virtualization and Open RAN (O-RAN)

RAN virtualization (vRAN) enables RAN functions to run on general-purpose shared computing platforms in the cloud or at the network edge. Fig. 5 illustrates the logical thread architecture in a virtualized RAN environment. When the virtalized BS (vBs) receives radio samples for a slot from the RU, a dispatcher

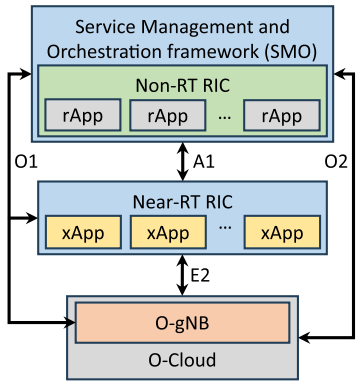


Fig. 6. O-RAN interfaces.

schedules the commuication processing tasks on the first idle thread in the first available or least congested processing unit. This worker thread executes a pipeline of tasks which includes processing data and control channels, and scheduling radio resources.

These workers have strict processing deadlines, yet their processing latency depends on the amount of radio resources carried in the slot and the SNR of the received signals [4]. Dedicated computing platforms provision computing resources to handle the worst-case load, while shared computing platforms multiplex resources across different workers. Consequently, the available computing resources in shared platforms depend on the current compute utilization of other workers. Section III presents a comprehensive experimental analysis of the computing requirements for uplink and downlink processing.

RAN virtualization is a cornerstone of the O-RAN Alliance (O-RAN) [1], which promotes open standards and interoperability, enabling operators to mix and match equipment from different vendors. Fig. 6 depicts the general O-RAN architecture. To support virtualization of CU, DU and even RU[2] components, O-RAN provides the O-Cloud, which is controlled by the Service Management and Orchestration (SMO) function. The O2 interface between the SMO and the O-Cloud enables orchestration and management of the resources required to deploy and operate RAN functions.

O-RAN incorporates two Radio Intelligent Controllers (RICs): the non-Real-time RIC (non-RT RIC) and the near-Real-time RIC (near-RT RIC). The RICs enable machine learning (ML)-driven automation for dynamic and optimized radio resource management. The non-RT RIC operates at a timescale of hundreds of milliseconds (i.e., $>$100 ms), while the near-RT RIC operates at a millisecond timescale (i.e., $<$10 ms). Applications leveraging the non-RT RIC are called rApps, while those in the near-RT RIC are called xApps.

Finally, the A1 and E2 interfaces facilitate communication between RICs and RAN elements for efficient network management and optimization. The A1 interface enables strategic policy enforcement and long-term network optimization by linking the non-RT RIC with the near-RT RIC. Conversely, the E2

[2]RU virtualization is out of the scope of this paper

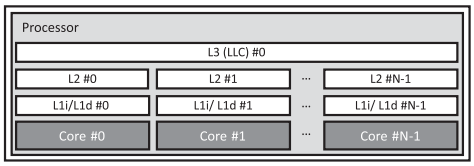


Fig. 7. General-purpose computing platform architecture.

interface facilitates low-latency, real-time control, and network adjustments by connecting the near-RT RIC to the RAN elements. The O1 interface facilitates operations, administration, and management (OAM) functions across the RAN.

C. General-Purpose Computing Platforms

Fig. 7 shows the architecture of a general-purpose computing platform (GPP). Modern server CPUs often feature multiple computing cores, each with different levels of dedicated cache memory. A computing core of a processor executes exactly one thread at a time. Unlike classical thread programming, the tight deadlines of vRAN's signal processing pipeline necessitate the use of real-time threads. When multiple real-time threads contend for the same core, they are enqueued in the core's scheduler queue and executed sequentially. When a thread executes, it loads frequently used memory values into the cache memory for faster access.

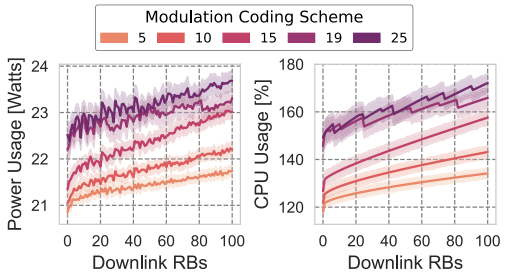
CPU cache memory is typically organized into multiple levels (L1, L2, L3 or LLC) and divided into *cache ways* to improve access efficiency. A cache way is a portion of a set-associative cache, where the total cache is divided into sets, and each set contains multiple ways (blocks) for storing data. Level 1 (L1) cache is the smallest and fastest, split into two types: L1i for instructions and L1d for data. Each physical core has its own dedicated L1 cache. L2 cache is larger and slower than L1, primarily used for data, and is also dedicated to each physical core. The L3 cache, or Last Level Cache (LLC), is the largest and slowest, shared among all cores. While L1 and L2 cache ways are dedicated per core, LLC cache ways can be assigned exclusively to different cores.

III. Experimental Analysis

In the this section, we provide a comprehensive analysis of the impact of radio and computing resource allocation on vRAN performance.

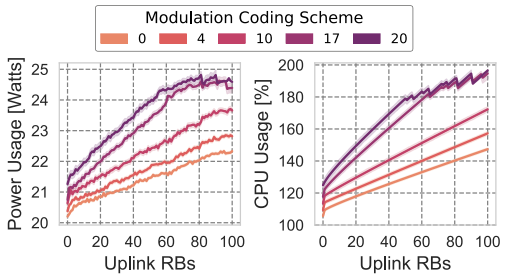
A. vRAN Experimental Platform

We built a proof-of-concept (PoC) vRAN system using a commercial off-the-shelf (COTS) computing node and several software-defined radio (SDR) front-ends following the design guidelines of [21]. The computing node is equipped with an Intel E5-2650 v4 @ 2.20 GHz CPU, containing 12 physical cores, with 384 KiB of L1 cache, 3 MiB of L2 cache, and 30 MiB of L3 cache.The SDR front-ends are Ettus USRP B210 boards, and we use a 3GPP Release 15-compliant vBS and UE stack from srsRAN [22], containerized with Docker. We run the mobile core software on a separate host and pair each vBS with



(a) Power (b) Computing usage

Fig. 8. Power and computing usage as a function of the used bandwidth in the downlink using 2 cores and 6 cache ways.



(a) Power (b) Computing usage

Fig. 9. Power and computing usage as a function of the used bandwidth in the uplink using 2 cores and 6 cache ways.

one UE to generate downlink (DL) and uplink (UL) network load. Our system supports up to 5 vBS deployed in the same computing node.

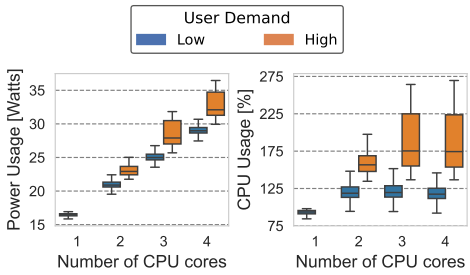
Following the O-RAN specifications, we implemented the E2 interface to enable the configuration of radio scheduling policies in the vBSs, specifically by setting bounds on the maximum number radio resources (i.e., RBs, see Section II-A) that are eligible for uplink and downlink). Furthermore, we incorporated techniques from Nuberu [15] to maintain user synchronization even under computing resource shortages ensuring reliability. We also developed a set of custom tools using the Docker API to dynamically orchestrate the vRAN system and configure various computing settings at runtime. Docker enables control over the allocation of computing cores to each instance. Finally, we leverage Intel CAT[3] to partition and allocate available LLC cache ways to different cores.

B. Radio Resources

We now study the effect of user demands and MCS selection on RAN performance. To this end, we deploy a single vBS in the testbed and measure its power consumption as a function of the allocated radio resources and their corresponding MCSs.

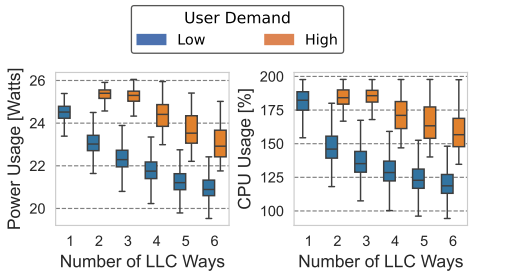
1) User Demand: The radio scheduler of a vBS allocates radio resources (i.e., RBs) to individual users based on their traffic demands and channel conditions. Figs. 8 and 9 depict the impact of increasing RB allocation on computing utilization and power consumption in both uplink and downlink for different MCSs. Observing the trend for a given MCS, we find that computing

[3][Online]. Available: https://github.com/intel/intel-cmt-cat



(a) Power (b) Computing usage

Fig. 10. Power and computing usage as a function of the computing cores allocation using 6 LLC cache ways and a high (>20 dB) SNR.



(a) Power (b) Computing usage

Fig. 11. Power and computing usage as a function of the LLC cache ways allocated using 2 computing cores and a high (>20 dB) SNR.

utilization and power consumption are linearly dependent on the number of allocated RBs. As more RBs are allocated to a user (i.e., the user has higher traffic demands), both computing time and power consumption increase. Moreover, uplink data processing is more computationally expensive than downlink data processing, as forward error correction (FEC) operations are more compute-intensive when decoding data (which may require multiple iterations) than when encoding data [4], [15].

2) Wireless Channel Quality: As mentioned earlier, the computing workload of a vBS depends not only on user demand dynamics but also on the SNR of the wireless channel, which is influenced by user mobility patterns and the dynamics of interfering sources. Higher SNR regimes allow RBs to carry more bits as they can be modulated with higher-order modulation and coding schemes. We confirm this with our own experiments. Figs. 8 and 9 also depict the computing usage and power consumption in the uplink and downlink for different MCSs. We observe that lower SNR regimes (i.e., lower MCSs) require less computing resources and power for the same number of RBs. However, ower MCSs carry less user data per RB which requires less computing capacity.

C. Computing Resources and Power Consumption

Next, we study the effect of allocating varying amounts of CPU cores and cache memory resources to a vBS instance. In this work, we focus solely on dedicated computing and memory resources allocated to vBS instances; we do not consider shared resources, as this results in resource contention effects (i.e., the "noisy neighbor" problem), which have been extensively studied in other works [13], [14], [15]. This experimental characterization aims to assess the impact of computing resource allocation on RAN performance. To this end, we deploy a single vBS in the testbed and evaluate its normalized throughput, computing utilization, and energy consumption as a function of the assigned computing resources.

1) CPU Cores: We begin by analyzing the impact of the number of CPU cores allocated to a vBS. Note that when a CPU core is allocated to a vBS, its dedicated L1 and L2 cache memory are also allocated to that vBS. No other vBS processes can utilize the same L1 and L2 cache unless the same core is allocated to them. Fig. 10(b) shows the distribution of computing utilization for an increasing number of allocated CPU cores under high SNR conditions (>20 dB) for both low traffic load (<25% of total traffic) and high traffic load (>75%), with 6 LLC cache ways allocated. In the case of one allocated CPU core, computing resources are insufficient to transmit and receive more than 75% load, and thus the distribution is not depicted. We observe that computing utilization initially increases and then plateaus as the number of cores increases. Moreover, computing usage increases significantly from 1 to 2 cores for low load and from 2 to 3 cores for high load.

Fig. 10(a) ilustrates the power of a vBS instance as a function of the number of allocated (i.e., pinned) CPU cores. While power increases linearly with the number of active cores, computing utilization tends to plateau once a sufficient number of cores are allocated. This difference arises because CPU core energy consumption has a fixed component when the core is active and a variable component that depends on its utilization [23], [24]. As a result, pinning additional cores to a vBS, introduces a fixed energy consumption offset, even if their utilization remains low. Therefore, while computing utilization may not change significantly, the overall power of the vBS increases. Finally, Fig. 2 shows the vBS performance in terms of the maximum achievable load as a function of the number of CPU cores. With 1 core and 6 LLC cache ways, the maximum achievable load is limited to 36% of the total capacity without losing user synchronization. However, when two or more cores are pinned, the vBS is able to sustain the full load capacity.

2) LLC Cache Memory: Next, we analyze the behavior of a vBS instance under different LLC cache way allocations. As explained in Section II, LLC cache is allocated in *ways* to specific cores. Therefore, only the vBS instances utilizing those cores can access the allocated portions of the LLC. Fig. 11(b) shows the computing utilization for an increasing number of allocated cache ways when two CPU cores are allocated, under high SNR conditions (>20 dB), and for both low traffic load (<25% of total traffic) and high traffic load (>75%). We observe that as more LLC cache ways are allocated to an instance, computing utilization decreases significantly. The more LLC cache ways are allocated to a computational task, the fewer cache misses are triggered. When a process does not find a required data item in the cache memory, it triggers a cache miss and the CPU must interrupt execution to fetch the data from main memory. This incurs additional CPU cycles, leading to increased execution time and, consequently, higher energy consumption. Thus, the

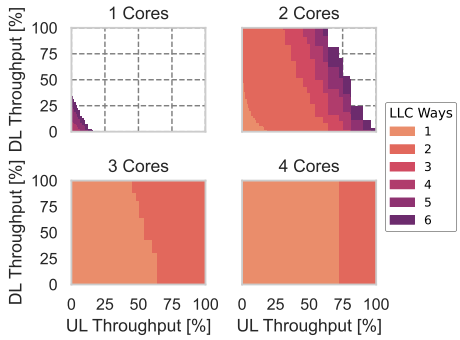


Fig. 12. Pareto front depicting the achievable throughput in uplink and downlink as a function of the computing resources allocated under high traffic load and high SNR regimes.

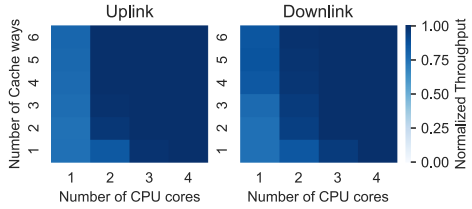


Fig. 13. Average normalized throughput for low user demand and SNR between 5 dB to 25 dB.

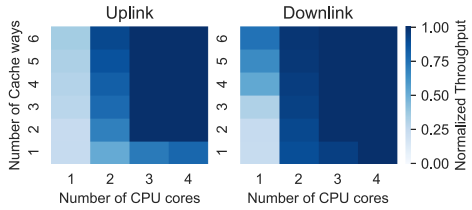


Fig. 14. Average normalized throughput for high user demand and SNR between 5 dB to 25 dB.

more LLC cache ways allocated, active threads experience fewer cache misses, and consequently, fewer computing cycles are wasted loading data into LLC memory [13], [14].

As previously shown in Fig. 2, the maximum achievable load increases as a function of the allocated LLC cache ways when two CPU cores are allocated. Similar to the CPU core allocation scenario, allocating more LLC cache allows for the utilization of more RBs. However, in the case of cache memory, we observe a gradual increase in the maximum achievable load from approximately 70% to 100% as the allocation increases from 1 to 6 cache ways.

D. Computing Resources and User Throughput

Next, we delve deeper into the impact of CPU core and LLC cache way allocation on user throughput. The allocation of these resources determines the maximum available computing capacity, which in turn dictates the amount of uplink and downlink radio resources that can be utilized without missing deadlines and, hence, without losing throughput. Fig. 12 shows a Pareto front of the maximum uplink (x-axis) and downlink (y-axis) user throughput as a function of the number LLC cache ways (filling color) and CPU cores (subplots) allocated, and under high SNR conditions (>20 dB). As observed in the previous sections, using a single CPU core severely limits the achievable uplink and downlink capacity. However, Fig. 12 for two CPU cores demonstrates that allocating more LLC cache ways significantly increases the achievable user throughput in both uplink and downlink. In this case, with two cores, one LLC cache way allows for utilizing 25% of the total uplink capacity, whereas six LLC cache ways enable utilizing up to 90%. Any RAN workload exceeding uplink or downlink capacity will lead to a loss of user synchronization, leading throughput to zero. Therefore, it is crucial to modulate the set of radio resources that a vBS can allocate within the bounds of the available computing capacity of the vBS to ensure reliable operation [15]. Finally, we observe that downlink workloads are less demanding than uplink workloads, as evidenced by the ability to utilize almost all available downlink radio resources even with minimal uplink throughput.

Figs. 13 and 14 depict the average normalized throughput in the uplink and downlink as a function of the number of CPU cores and LLC cache ways for low load (<50% of the maximum capacity of the vBS) and high load (>50%) across a wide SNR range (5 dB to 25 dB). Significant differences are observed between different user loads. The average normalized throughput is high for both uplink and downlink under low user demand, which has less stringent computing requirements compared to high user demand. Consequently, throughput can be maintained even with fewer computing resources under low load. However, under high uplink user demand, throughput cannot be maintained unless 3 CPU cores and 2 LLC cache ways are used. Similar differences are observed for the downlink under low and high user demand. Finally, we remark that under-allocating computing resources is feasible as long as the vRAN workloads do not exceed the maximum computational capacity. In the case of low user load, normalized throughput is higher because it approaches the maximum throughput allowed by the specific allocation of computing resources.

E. The Problem

Computing and memory resources are limited in any shared computing platform, creating a critical trade-off between energy savings and fairness in resource allocation. This trade-off arises due to the dynamic nature of vRAN workloads, which depend on both traffic demands and the maximum computational capacity of the underlying infrastructure.

1) Energy Savings: When user demand is low (i.e., when vRAN workloads have low computing requirements), significant energy savings can be achieved by deactivating unused CPU cores. In such scenarios, with reduced traffic demands and lower computational pressure, tailoring computing resources to user needs increases energy efficiency with minimal performance impact. Figs. 15 and 16 depict the potential average energy savings in our platform as a function of the number of CPU cores and LLC cache ways for low load (<50% of the vBS's maximum capacity) and high load (>50%) across a wide SNR range (5 dB to 25 dB).

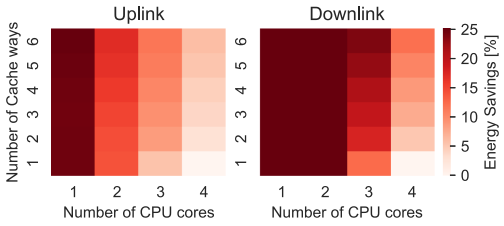


Fig. 15. Average energy savings for a low user demand and SNR between 5 dB to 25 dB.

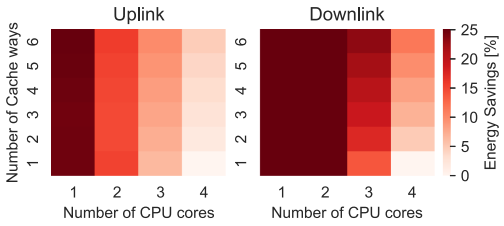


Fig. 16. Average energy savings for a high user demand and SNR between 5 dB to 25 dB.

These plots present the attainable energy savings corresponding to Figs. 13 and 14. We observe that, naturally, the highest energy savings are achieved with smaller allocations of CPU cores and more LLC cache ways. Conversely, the lowest energy savings are obtained with the maximum number of CPU cores and the minimum number of LLC cache ways. Furthermore, there are significant differences between uplink and downlink, with greater energy savings achievable in the downlink. However, as shown in Section III-D, not all configurations can serve all user demands. Therefore, it is crucial to understand the demands to tailor the computing allocation, minimizing resource usage while meeting user needs.

2) Fairness: As the number of vBSs deployed in the same infrastructure increases or their workloads become more computationally demanding, the total computational demand approaches the available capacity, making it impossible to deactivate CPU cores for energy savings. In such high-demand scenarios, computational resources may become constrained, leading to insufficient processing power to handle all vRAN tasks at optimal performance levels. In this case, we advocate for fairly distributing the performance losses (throughput degradation) among the deployed vBSs. We formalize our definition of fairness later.

This trade-off is particularly relevant in scenarios with heterogeneous vBSs. To illustrate this, we consider a scenario with 5 vBSs deployed on a platform with 10 CPU cores and 8 LLC cache lines. Each vBS experiences a different user load: vBS 1 has an average user demand of 40% of the total capacity, vBS 2 has 20%, vBS 3 has 10%, and vBSs 4 and 5 each have 65%.

Fig. 17 presents the performance loss (ratio of bits lost due to insufficient computing capacity) for each vBS. When computing resources are partitioned equally (i.e., each vBS is allocated one cache line, with spare cache lines distributed randomly), the traffic losses are distributed unevenly. vBSs 1 and 2, which have lower user loads (and thus lower computational requirements) than vBSs 4 and 5, experience disproportionately higher losses. Specifically, vBS 2 loses 100% of its traffic, vBS 1 loses slightly less than 25%, and vBS 4 loses 50%.

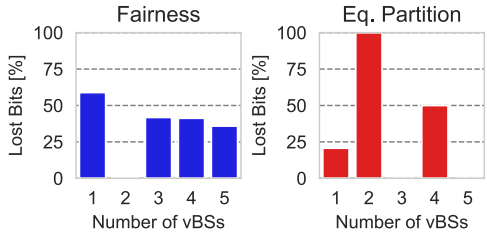


Fig. 17. Fair and computing resource distribution when under different setups.

However, with a proportionally fair share of performance losses (details presented later), these losses are distributed more evenly among the vBSs, minimizing the disparity between them. In this case, vBS 2 experiences no loss, while vBSs 3, 4, and 5 each lose between 35% and 40% of their user demand. vBS 1 loses slightly over 50%. Although more vBSs experience some loss in this scenario, it is preferable because no single vBS suffers a complete loss of its user demand.

These results highlight a key trade-off in vRAN resource management between energy efficiency and fairness. While deactivating unused CPU cores maximizes energy savings during periods of low workload, ensuring fairness in resource allocation becomes paramount during high-demand periods. In the latter case, a fair resource distribution is essential to prevent disproportionate performance degradation across vRAN workloads.

IV. Problem Formulation

We consider a vRAN system comprising a set $\mathcal{N}$ of N virtualized BS (vBS) sharing the same computing node. We divide time into discrete *decision periods* denoted by $t = 0, 1, \ldots,$ T. For each vBS $n \in \mathcal{N}$ the traffic demand in the Uplink (UL) and the Downlink (DL) at time t is denoted by $d_t^{\mathrm{UL},n}$ and $d_t^{\mathrm{DL},n}$, respectively. Moreover, the UL and DL modulation and coding scheme (MCS) is denoted by $m_t^{\mathrm{UL},n}, m_t^{\mathrm{DL},n}$, respectively. We define the *context* of the vBS n as $c_t^n := (d_t^{\mathrm{UL},n}, d_t^{\mathrm{DL},n}, m_t^{\mathrm{UL},n}, m_t^{\mathrm{DL},n}) \in \mathcal{C}$, where $\mathcal{C}$ is the context space. Note that the context c_t^n captures the *average* state of a vBS during the decision period t.

The computing node where the vBSs are deployed provide a set $\mathcal{U}$ of U CPU cores and a set $\mathcal{L}$ of L Last-Level Cache (LLC) ways. In order to avoid interference among the computing processes (e.g., the noisy neighbor problem [13]), we assign dedicated resources to each vBS. The number of CPU cores and LLC ways assigned to vBS $n \in \mathcal{N}$ at time t are denoted by u_t^n and l_t^n, respectively. The computing assignment for vBS n is $x_t^n = (u_t^n, l_t^n) \in \mathcal{X}$, where $\mathcal{X}$ is the set of all possible assignments. We denote the joint assignment vector as $\mathbf{x}_t = \{x_t^n\}_{n \in \mathcal{N}}$. The power consumption of a vBS n depends on its context and the computing resource assignment at time t (as shown in Section III) and is modeled by $p_t^n(c_t^n, x_t^n)$, where $p_t^n : \mathcal{C} \times \mathcal{X} \longrightarrow \mathbb{R}^+$. Finally, the ratio of bits lost by a vBS n due to missing computing deadlines is given by $b_t^n(c_t^n, x_t^n)$, where $b_t^n : \mathcal{C} \times \mathcal{X} \longrightarrow [0, 1]$.

We now formulate a problem with the goal of minimizing overall power consumption while satisfying the reliability threshold in terms of bit loss:

Problem 1:

$$\min_{\{\boldsymbol{x}_t\}_{t=1}^T} \quad \sum_{t=1}^{T}\sum_{n\in\mathcal{N}} p_t^n\left(c_t^n, x_t^n\right)$$
$$\text{s.t.} \quad \frac{1}{T}\sum_{t=1}^{T}\sum_{n\in\mathcal{N}} b_t^n\left(c_t^n, x_t^n\right) \le \epsilon$$
$$\sum_{n\in\mathcal{N}} u_t^n \le U, \ \sum_{n\in\mathcal{N}} l_t^n \le L \quad \forall t = 1, 2\ldots, T$$

where ϵ set the reliability target (e.g., $\epsilon = 10^{-5}$ in [11]).

In Problem 1, we assume that for the set $\mathcal{N}$ of vBSs and their contexts $\{c_t^n\}_{n\in\mathcal{N}}$, there exists a set of configurations $\{x_t^n\}_{n\in\mathcal{N}}$ that satisfies the throughput requirements at each t, i.e., the problem is feasible. However, there are cases in which the vBSs require more computing resources than the ones available on the computing platform. Computing resource scarcity leads to bit losses at the vBSs because they cannot execute network function processes (e.g., PHY processing) in time. In these cases, the objective is to minimize the throughput loss regardless of the consumed energy:

Problem 2:

$$\min_{\{\boldsymbol{x}_t\}_{t=1}^T} \quad \sum_{t=1}^{T}\sum_{n\in\mathcal{N}} f_\alpha\left(b_t^n\left(c_t^n, x_t^n\right)\right)$$
$$\text{s.t.} \quad \sum_{n\in\mathcal{N}} u_t^n \le U \ \sum_{n\in\mathcal{N}} l_t^n \le L \quad \forall t = 1, 2\ldots T$$

where f_α is a generalized α-fairness function defined as follows:

$$f_\alpha(z) = \begin{cases} \frac{z^{1-\alpha}-1}{1-\alpha}, & \text{for } \alpha \in \mathbb{R}_{\ge 0}\backslash\{1\}, \\ \log(z), & \text{for } \alpha = 1. \end{cases} \tag{1}$$

The goal of f_α is to distribute throughput losses fairly across vBSs. This is challenging due to the context-dependent computing requirements of vBSs and the non-linear relationship between reduced computing resources and throughput losses (Section III). The parameter α determines the fairness type: $\alpha = 1$ enforces proportional fairness, while $\alpha \to \infty$ leads to max-min fairness.

Note that solving these problems is a highly complex task for several reasons. First, the power consumption function $p_t^n(\cdot)$ and the bit loss function $b_t^n(\cdot)$ are unknown and exhibit complex behavior, as demonstrated in Section III. Both functions depend on the specific hardware of the computing platform and the implementation of the vBS software functions within $\mathcal{N}$. Moreover, observations of these metrics may be subject to inherent noise from the experimental measurements. Second, it is challenging to identify whether we are facing Problem 1 or 2 in specific operating points. An average bit loss exceeding ϵ may result from either a saturated system (necessitating the solution of Problem 2) or a non-saturated system with a suboptimal selection of x_t^n (requiring the solution of Problem 1). As the traffic load fluctuates over time, the system may saturate only during certain time steps (e.g., peak hours). Finally, the solution space is very large, i.e., $\binom{N+U}{N} \cdot \binom{N+L}{N}$. The number of active vBSs can vary over time, thereby altering the dimension of the solution space. Moreover, the optimal solution changes over time due to the time-varying nature of the vBS contexts. These changes can occur on a timescale of seconds or less.

For all these reasons, the use of traditional strategies from the literature such as optimization or heuristic approaches for resource allocation problems are not applicable to our problem. To overcome these challenges, we propose a data-driven solution, detailed in the next section.

V. Solution Design

In this section, we propose a novel data-driven algorithm that combines several techniques to address the challenges outlined previously. First, we rely on deep reinforcement learning (RL) at the core of our framework, specifically the discrete soft actor-critic (SAC) algorithm, known for outstanding performance without hyperparameter tuning. Second, we break down the dimensionality of the solution space by implementing a multi-step decision-making process that allocates resources to different vBS sequentially. Third, we design a reward signal that addresses the two problems from Section IV and mitigates sparse rewards in long RL episodes. Lastly, we use action masking to prevent infeasible decisions, such as over-allocating resources.

In the rest of this section, we introduce our RL formulation (Section V-A), describe our data-driven framework (Section V-B), outline our solution architecture (Section V-B5), and present a new training method using digital twinning for large-scale scenarios (Section V-C).

A. Discrete-SAC Formulation

We consider a finite-horizon Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. The state space is denoted by $\mathcal{S}$, the action space is $\mathcal{A}$, the state transition probability is $p : \mathcal{S}\times\mathcal{S}\times\mathcal{A} \to [0, 1]$, the reward function is $r : \mathcal{S}\times\mathcal{A} \to [r_{\min}, r_{\max}]$, and $\gamma \in [0, 1]$ is the discounting factor. We let τ_π be the distribution of trajectories induced by a policy π.

While standard RL algorithms consider the maximization of the sum of discounted rewards, i.e., $\sum_{k=1}^{K}\mathbb{E}_{(s_k,a_k)\sim\tau_\pi}[\gamma^k r(s_k, a_k)]$, we consider a more general entropy-regularized objective. Specifically, the optimal policy is defined as follows:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{k=1}^{K}\mathbb{E}_{(s_k,a_k)\sim\tau_\pi}\left[\gamma^k\left(r(s_k, a_k) + \alpha\mathcal{H}(\pi(\cdot|s_k))\right)\right],$$

where α sets the relative importance of the entropy term versus the reward, and $\mathcal{H}(\pi(\cdot|s_k))$ is the entropy of the policy π at state s_k, which by definition $\mathcal{H}(\pi(\cdot|s_k)) = -\log\pi(\cdot|s_k)$.

While most other actor-critic methods enable exploration by adding noise to the result of a deterministic policy, this entropy-regularized RL approach uses entropy as an indicator of the randomness of the policy. This measure is used to improve randomness during training, which encourages exploration and avoids local optima.

1) Policy Definition: The set of possible solutions for our problem is finite and therefore the cardinality of the action space $\mathcal{A}$ is also finite. In the original formulation of SAC, the output of the policy is a Gaussian distribution over a continuous action space [25]. In contrast, our policy needs to generate a discrete probability distribution over the finite action space $\mathcal{A}$. Moreover, we consider that some actions may be invalid or unfeasible at different time steps. Specifically, the set of valid actions at time k depends on the state $s_k \in \mathcal{S}$ and is given by $mask : \mathcal{S} \rightarrow \{0,1\}^{|\mathcal{A}|}$ [26]. Now, we define the policy as a function that outputs the probability of each $a \in \mathcal{A}$ for a given state, i.e., $\pi_\phi(s_k, mask(s_k)) : \mathcal{S} \times \{0,1\}^{|\mathcal{A}|} \rightarrow [0,1]^{|\mathcal{A}|}$, where ϕ are the parameters of the policy. Note that the logits of invalid actions take a value very close to zero, and a softmax function is used to ensure valid probability distributions.

2) Policy Improvement: The objective of the policy update is to maximize the expected return while considering the entropy of the policy, i.e., the maximum entropy principle is used to encourage exploration. The policy update is originally formulated as the minimization of the KL-divergence between the current policy and the exponential of the Q-function [25]. For our specific case of finite actions space, the policy update can be written as:

$$J_\pi(\phi) = \mathbb{E}_{s_k \sim D}\big[\pi_\phi(s_k, mask(s_k))^T \cdot \cdot \left[\alpha \log(\pi_\phi(s_k, mask(s_k)) - Q_\theta(s_k)\right]\big], \quad (2)$$

where Q_θ is the Q-function with parameters θ. Q_θ is responsible for estimating the value of a state, i.e., the expected long-term return of a given state. Note that the Q-function is redefined in (2) so that instead of providing the Q-value for a given state and action, it provides the values of all the actions for a given state. Formally, $Q_\theta : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|}$.

3) Q-Function Learning: To perform the policy improvement according to (2), the Q-function needs to be approximated. For that purpose, the following objective function is formulated [25]:

$$J_Q(\theta) = \mathbb{E}_{(s_k,a_k)\sim D}\Bigg[\frac{1}{2}\left(Q_\theta(s_k,a_k) - \left(r(s_k,a_k) + \gamma \mathbb{E}_{s_{k+1}\sim p}[V(s_{k+1})]\right)\right)^2\Bigg],$$

where the state-value function is defined as $V(s_k) := \mathbb{E}_{a_k \sim \pi}[Q(s_k,a_k) - \alpha \log(\pi(a_k,s_k))]$. In the original SAC formulation, $V(s_k)$ is given by a parametrized function approximator as the expectation is difficult to compute. However, in our case with a discrete policy, we can compute the expectation directly over the policy, avoiding approximation that may lead to higher variance in Q_θ, i.e.,

$$V(s_k) = \pi_\phi(s_k, mask(s_k))^T \cdot \cdot \left[Q_{\hat{\theta}}(s_k) - \alpha \log(\pi_\phi(s_k, mask(s_k)))\right]. \quad (3)$$

To stabilize learning, we define an online and target Q-functions approximators, with parameters θ and $\hat{\theta}$, respectively. The target Q-function is soft updated and used in (3), according to [27]. Note that our algorithm only requires parametrized function approximators for π_ϕ, Q_θ, and $Q_{\hat{\theta}}$. We use discrete SAC at the core of our solution due to its exceptional performance. Particularly, discrete SAC not only outperforms state-of-the-art RL methods in a variety of discrete environments but also is very robust to hyper-parameter configuration across diverse environments [28]. Discrete SAC comes with convergence guarantees given by Theorem 1 in [25].

B. Algorithm Design

As mentioned previously, the large action space poses a significant challenge for standard RL algorithms in learning effectively. This action space encompasses a single action that specifies both the number of CPU cores and LLC ways allocated to each vBS. Its dimensionality increases *combinatorially* with the number of vBSs and the available resources (See Proposition in Appendix A, available online).

To break down the complexity of the problem and allow learning in a reasonable amount of time, we divide the resource assignment decisions into an epoch of N *decision steps* indexed by k. More specifically, at each decision step we assign resources to one vBS, i.e., the number of decision steps in an epoch corresponds with the number of active vBSs. Then, the action space becomes $\mathcal{A} = \{0, \dots U\} \times \{0 \dots L\}$ and therefore its cardinality is $|\mathcal{A}| = (U+1) \cdot (L+1)$. Now, $|\mathcal{A}|$ only depends on the amount of computing resources, while the number vBS modifies the number of decision steps k within the RL epochs. Note that one RL epoch of $k = 1 \dots N$ *decision steps* is executed at every *decision period* denoted by $t = 0, 1, \dots, T$. Now, as N grows, our solution needs to deal with more decision steps (larger horizons). However, it has been consistently shown in the literature that RL performs extremely well with very large decision horizons [29], [30], [31]. Thus, we leverage this property of RL to effectively address the complexity of this problem, avoiding the combinatorial complexity.

Based on this design of the resource assignment decisions, we now define the states, actions, and reward signals that allow our problem to be solved as a finite horizon RL problem.

1) Actions: At each decision step k the algorithm shall select the computational resources to be assigned to the vBS $k \in \mathcal{N}$, i.e., $a_{k,t} := (u_t^k, l_t^k) \in \mathcal{A} = \{0, \dots U\} \times \{0 \dots L\}$.

2) States: The state $s_{k,t}$ shall encode the context of the vBS $k \in \mathcal{N}$ at time t because, as shown in Section III, the context c_t^k determines the amount of computing resources that the vBS requires. However, this information is insufficient to perform a fair splitting of the shared resources (i.e., Problem 2). For example, we may assign a sufficient amount of resources to the vBSs at the beginning of the epoch (e.g., $k = 1$) and run out of resources available for the vBSs at the end of the epoch (e.g., $k = N$). To avoid this issue, we need to capture the context information of the unassigned vBSs at each time step, i.e., at the decision step k, we need to take into account the context information of vBSs $k+1, \dots N$. To implement this idea, the simplest approach is to concatenate the context information $c_t^{k+1}, \dots c_t^N$. However, this approach results in actions with variable dimensionality, which is not convenient for our learning

algorithm. Instead, we propose a very simple strategy:

$$s_{k,t} := \left(N-k, U_t^k, L_t^k, c_t^k, \text{stats}\left(c_t^{k+1}, \ldots, c_t^N\right)\right) \quad (4)$$

where $N-k$ is the number of vBSs left to allocate resources for, and $U_t^k := U - \sum_{j=1}^{k-1} u_t^j$ and $L_t^k := L - \sum_{j=1}^{k-1} l_t^j$ are the resources left to allocate. Here, stats$(\cdot)$ is a function that projects a variable number of vectors of dimensionality D into a fixed set of vectors of the same dimensionality [32]. Specifically, this function performs the following operations across vectors in an element-wise fashion: Sum, Mean, Standard Deviation, Max, Min, Range, and Median; i.e., stats$(\cdot) : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{7\times D}$. We rely on this strategy instead of other learning-based approaches from the literature (e.g., LSTM or transformers) because it is very simple, does not require training, and provides very high performance as shown in Section VI.

3) Reward: Our goal is to design a single reward signal that addresses two distinct but interrelated objectives: (i) minimizing energy consumption while ensuring high reliability, as outlined in Problem 1, and (ii) fairly maximizing throughput in a saturated system, as detailed in Problem 2. To achieve this, we approximate both objectives using a single reward function.

$$r_{k,t} = -p_t^k\left(c_t^k, a_{k,t}\right) - \lambda f_\alpha\left(b_t^k\left(c_t^k, a_{k,t}\right)\right), \quad (5)$$

The reward function consists of two terms: the first term quantifies the energy consumption of the vBS assigned at decision step k with resource allocation $a_{k,t}$; the second term accounts for the bit loss incurred by the vBS, transformed by the α-fairness function. Here, λ serves as a weighting parameter, and the negative sign ensures the minimization of the energy consumption. When there is no throughput loss, the second term becomes negligible, effectively reducing the problem to the minimization of energy consumption (Problem 1). On the other hand, in the presence of throughput loss, a sufficiently large λ is chosen to prioritize the second term, thereby addressing the fairness aspect of throughput maximization (Problem 2). Note that the value of α should be fixed before training according to the fairness objectives of the system.

A natural approach for designing the reward signal assigns $r_{k,t} = 0$ for all $k < N$ and computes a final reward at $k = N$. However, this sparse structure delays feedback, making it hard for the agent to link actions to outcomes, a common challenge in reinforcement learning with sparse rewards [33], [34]. Sparse rewards also slow learning, reduce the number of policy updates, and increase the risk of getting stuck in local optima due to high variance and infrequent feedback. These issues are amplified in long episodes with many vBSs [35], [36], [37]. In contrast, our proposed reward signal (5) avoids these problems, enabling faster convergence.

We would like to highlight that we do not assume any model for the behavior of the vBS (i.e., power consumption and lost bits). The reason is that these metrics depend on many deployment-specific aspects (e.g., hardware specifications of the computing platform, and software implementation of the vBS stack). Instead, we follow a data-driven approach, i.e., we observe (and learn) the behavior of the vBS in terms of power consumption and bit losses as a function of the context.

4) Invalid Action Masking: When resources are allocated at each decision step, certain actions become invalid. For instance, at decision step $k = 1$, u_t^1 CPU cores are assigned to the first vBS. Consequently, at the second decision step $k = 2$, only $U_t^2 = U - u_t^1$ CPU cores remain available, rendering actions that assign $[U_t^2 + 1, \ldots U]$ CPU cores invalid. This principle extends to all dimensions of the action space. To address this, it is more effective to mask invalid actions (i.e., make them unavailable to the algorithm) than to allow the algorithm to select them and subsequently penalize such choices in the reward [26]. Therefore, we re-normalize the probability distribution over actions, $\pi(\cdot|s_{k,t})$, to exclude invalid actions, ensuring the agent samples only from valid ones. Furthermore, we apply gradient masking by setting the gradients of the logits corresponding to invalid actions to zero during backpropagation, thus preventing updates based on invalid choices [26]. This approach improves efficiency and accelerates learning.

5) Architecture Integration: AegisRAN is deployed as an rApp in the non-RT RIC of the O-RAN architecture (see Fig. 6). Our solution uses the O2 interface to allocate computing resources to the O-RAN vBS instances deployed in an O-Cloud platform. As vBSs are commonly deployed on an isolated environment using a virtualization manager (e.g., Kubernetes), our rApp can reconfigure the computing resources allocated to each vBS independently during run time. Typically, resource reconfiguration takes several tenths of milliseconds on containerized environments such as Docker. Since our rApp is designed to operate at non-RT RIC time scales (i.e., $1 - 60$s) we consider the reconfiguration overhead negligible. The vBS context observation is received via E2 interface and reward aggregates metrics from the E2 interface (i.e., telemetry on the data rate and throughput) and O2 interface (i.e., infrastructure data on energy consumption).

C. Large Scale Training Via Digital Twining

Data-driven learning algorithms require a training phase, usually done in a controlled environment to avoid network performance degradation during learning. However, even in controlled environments, training can be resource-intensive and time-consuming, especially with a large number of vBSs, which leads to a high-dimensional complex global context (joint combination of traffic demands and channel quality). Additionally, offline datasets are not suitable for RL algorithms, which need data generated by their own policy [38]. For these reasons, we propose an alternative training methodology based on Digital Twins (DT). We rely on the fact that vBSs processes use dedicated resources and therefore there is no interference among them. We define the DT of vBS $n \in \mathcal{N}$ as $\eta_n : \mathcal{C} \times \mathcal{A} \longrightarrow \mathbb{R}^2$, i.e., for a given context and resource allocation, the DT computes an approximation of the power consumption $\tilde{p}$ and an approximation of the lost bits $\tilde{b}$. We define he global DT as $\eta = \{\eta_n\}_{n\in\mathcal{N}}$. The global DT computes the global performance in terms of power and lost bits of the system as a function of the global context $\mathbf{c}_t : -\{c_t^n\}_{n\in\mathcal{N}}$ and the resource allocation $\mathbf{a}_t : -\{a_{n,t}\}_{n\in\mathcal{N}}$. For that purpose, $|\mathcal{N}_{\text{type}}|$ DTs need to be trained, where $\mathcal{N}_{\text{type}} \subseteq \mathcal{N}$ is the set of unique vBS

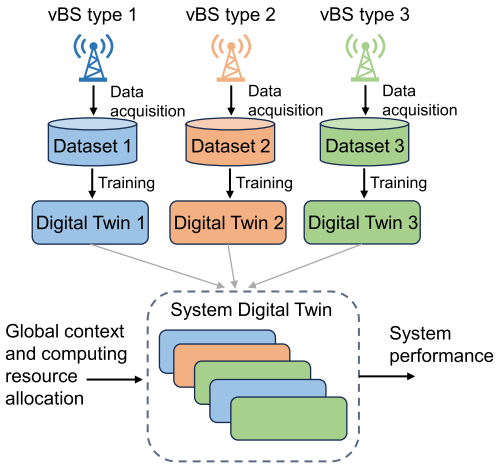


Fig. 18. Illustration of DT generation for 3 vBS types, each with its own dataset for individual training. The DTs are then combined based on the deployed vBSs to emulate the system's global performance.

types. Fig. 18 illustrates the DT generation, also detailed as follows:

- *Data acquisition:* We generate a dataset for each vBS type in $\mathcal{N}_{\text{type}}$. The type of vBS refers to different implementations of the vBS virtual functions (e.g., FlexRAN [39], srsRAN [22]) of different versions of the same implementation. Specifically, the dataset characterizes the power consumption and the bit loss ratios for a given vBS context and computing resource allocation. Note that the dataset is generated for a specific computing platform as different hardware (e.g., CPU chipset) can consume different power to achieve the same performance.
- *DT Training:* We train one DT per vBS type, i.e., $\eta_n \ \forall n \in \mathcal{N}_{\text{type}}$. The DT approximates the power consumption and the bit loss as a function of the vBS context and the computing resource allocation.
- *System DT operation:* We stack together the DTs as a function of the type and number of active vBSs in the system, i.e., $\eta = \{\eta_n\}_{n\in\mathcal{N}}$. Now, for a given global context $\mathbf{c}_t$ and joint action $\mathbf{a}_t$, the system DT computes an approximation of the global power consumption, denoted by $\{\tilde{p}_t^n\}_{n\in\mathcal{N}}$ and the total bit loss, denoted by $\{\tilde{b}_t^n\}_{n\in\mathcal{N}}$.

VI. Experimental Evaluation

Using the experimental platform in Section III-A, we evaluate our solution's performance. First, we evaluate a medium-sized deployment (up to 5 vBSs), comparing AegisRAN with two heuristic approaches and three oracle schemes. Next, we consider a large-scale scenario (up to 60 vBSs), where exhaustive oracles are computationally infeasible. In both cases, we assess settings with over-dimensioned resources (focused on energy minimization) and under-dimensioned ones (focused on fair throughput maximization).

It is important to note that the size of the scenario, defined by the number of aggregated vBSs, is influenced by multiple factors, including the geographical location of the radio units (RUs) and the degree of network densification. Specifically, the length of the fronthaul –the link connecting the RU to the vBS– cannot exceed certain thresholds without risking synchronization issues among the vBS components. This constraint means that urban areas with higher levels of network densification are more conducive to aggregating a larger number of vBSs.

The largest known real-world base station (BS) centralization was conducted by NTT Docomo, which successfully aggregated 48 BSs within a single computing server [40]. Building on this precedent, our evaluation explores scenarios with up to 60 aggregated vBSs within a single server.

Algorithm 1: AegisRAN Training

Inputs: Parameters ϕ and θ; Replay Buffer D filled with random transitions; Trained DT η; Learning rates λ_θ, λ_ϕ; Polyak averaging hyperparameter τ.
$\hat{\theta} \leftarrow \theta$
for each training episode **do**
  for $t = 1, \ldots, T$ **do**
    Observe global context $\mathbf{c}_t : -\{c_t^1, \ldots, c_t^{N_t}\}$
    Observe the number of active vBSs N_t
    for $k = 1, \ldots, N_t$ **do**
      Observe system state $\mathbf{s}_{k,t} = (N_t - k, U_t^k, L_t^k, c_t^k, stats(\mathbf{c}_t))$
      Compute action masking $m_{k,t} = mask(s_{k,t})$;
      Compute action $a_{k,t} \sim \pi_\phi(s_{k,t}, m_{k,t})$
      Compute lost bits and power consumption $(\tilde{b}_t^k, \tilde{p}_t^k) \sim \eta_k(c_t^k, a_{k,t})$
      Compute $r_{k,t}$ by (5)
      Get new context $c_t^{k+1} \in \mathbf{c}_t$
      $D \leftarrow D \cup \{(c_t^k, a_{k,t}, r_{k,t}, m_{k,t}, c_t^{k+1})\}$
    end for
  end for
  for each gradient step **do**
    $\theta \leftarrow \theta - \lambda_\theta \hat{\nabla} J(\theta)$
    $\phi \leftarrow \phi - \lambda_\phi \hat{\nabla} J_\pi(\phi)$
    $\hat{\theta} \leftarrow \tau\hat{\theta} + (1-\tau)\hat{\theta}$
  end for
end for
Output: Trained policy π_ϕ

A. Implementation Details

1) Digital Twin (DT): The DT receives as input a tuple with the context and resource allocation of vBS n, (c^n, x^n) and approximates its power consumption and bit loss ratio for DL and UL $(\tilde{p}^n, \tilde{b}^{DL,n}, \tilde{b}^{UL,n})$ via supervised learning. We implement the DT using a Kolmogorov–Arnold Network (KAN) [41] instead of traditional neural networks (NN) because they showed in our experiments superior performance with lower complexity (e.g., fewer layers and units). Specifically, the network has two hidden layers of sizes $\{32, 16\}$, respectively, a grid size of 50, and degree 3 for the splines. We use MAE as a loss function as it is more robust to the outliers that are present in our datasets due to experimental noise. We use AdamW optimizer and decoupled weight decay regularization, following the guidelines from [41].

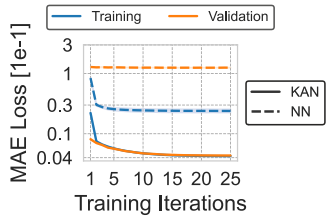


Fig. 19. DT training curves comparing KANs vs. NNs.

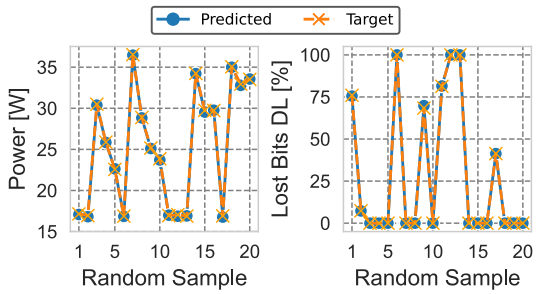


Fig. 20. DT evaluation in 20 random samples.

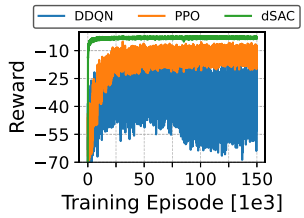


Fig. 21. Comparison of the training curve of AegisRAN using different RL algorithms and with up to 5 vBS.

We also use early stopping [42] with patience equals 5 to avoid over-fitting during the DT training as suggested in [41]. We finally train the DT with an experimental dataset of $1.5 \cdot 10^6$ samples, from which 33% are used only for validation. The DT is trained during a total of 25 epochs with a batch size of 256 samples.

2) RL Framework: We implement both actor and critic using NNs with two hidden layers of 256 units each and ReLU activation. We use orthogonal initialization for the NN weights to stabilize the training phase and prevent vanishing or exploding gradients. For the RL learning phase, the temperature parameter is set to 0.01, the learning rate is 10^{-3}, and a soft update hyperparameter is 0.005 [43], [44]. We use Adam optimizer [45], a batch size of 256, and a replay buffer of 10^5 samples. We set $\lambda = 1$ in the reward for the setting in Section VI-C and $\lambda = 10$ in Section VI-D (see Appendix B, available online). We also use two separately trained soft Q-networks to avoid state-value overestimation [46]. We use PyTorch[4] as a deep learning framework for all the models.

B. Convergence

We begin our evaluation by assessing the training and inference performance of the DT. Without loss of generality, we consider one vBS type, as described in Section III-A. Fig. 19 presents the training and validation curves of the DT's training phase considering a KAN model described in the previous version and a traditional fully connected NN architecture with 5 layers and 256 units per layer. According to the analysis in [41], the NN architecture presents 3x more computational complexity compared to the KAN model. Fig. 19 shows the superior performance of the KAN architecture in comparison to a more complex neural network (NN) architecture. Specifically, the NN exhibits higher loss during training and poorer generalization at validation, confirming the KAN's enhanced capacity to generalize. These results are consistent with findings reported in the literature [41], which emphasize the KAN's advantages in terms of both convergence and generalization. In particular, after training the KAN-based DT incurs an error of $8.4 \cdot 10^{-2}$ W, $4.66 \cdot 10^{-8}, 1.35 \cdot 10^{-7}$, for Power, Lost bits Downlink, Lost bits Uplink, respectively. This very small error is illustrated visually in Fig. 20, which depicts the values of 20 randomly selected ground truth values (orange crosses connected by dashed lines) alongside the corresponding predictions made by the DT model (blue points connected by a solid blue line). The near-perfect overlap between the predicted and the ground truth points indicates minimal variance in the prediction error.

We now evaluate the training performance of AegisRAN. To this end, for each episode during the training phase, we randomly select the number of active vBSs, $N \in [2, 5]$, and their contexts from the global context space $\mathcal{C}^N$. We also compare against two state-of-the-art (SoTA) RL algorithms Double DQN (DDQN) and proximal policy optimization (PPO) with the same configurations (i.e., same reward function, implementation of action masking, etc.) Fig. 21 shows the average reward as a function of the training epochs. Our proposal (discrete SAC) converges around episode $2 \cdot 10^3$, with minimal reward fluctuations thereafter, indicating robust learning. We also observe that discrete SAC not only exhibits a faster convergence but also shows a higher performance (the reward values are higher after convergence and with less variance). The superior performance of discrete-SAC is due to its entropy-based exploration. A better exploration strategy can potentially improve not only the convergence of the algorithm but also its performance at inference time due to a more effective learning phase. These results are in line with the literature that shows that entropy-based exploration strategies work better in complex environments [47], [48].

Finally, we evaluate the convergence of our solution in large-scale scenarios (up to 60 vBSs). Fig. 22 shows the episode reward per vBS for different levels of vBS aggregation. We consistently observe a quick convergence regardless of the number of aggregated vBSs. This finding is notable given that the solution space expands in a combinatorial manner with the number of vBSs (see Appendix A, available online). Despite the growing complexity of the problem with the increase in vBSs, our approach exhibits strong scalability with respect to the horizon length (N), effectively managing the increasing problem size.

[4][Online]. Available: http://www.pytorch.org

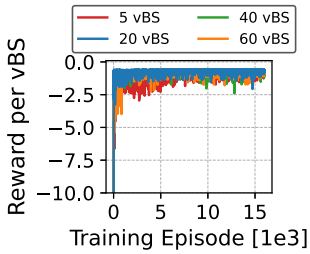


Fig. 22. Training curve of AegisRAN for different deployment scales over 10 independent runs.

C. Small/medium-Scale Scenarios

Next, we compare the performance of AegisRAN and the following benchmarks:

- *Oracle:* It uses exhaustive search to find the solution that maximizes the aggregated reward in (5) for a given number of vBS and set of contexts, i.e., $\min_{\{a_{k,t}\}_{k=1}^{K}} \sum_{k=1}^{K} r_{k,t}$. Note that this solution is only feasible to compute with relatively small solution spaces.
- *"CPU Oracle":* It finds the optimal CPU allocation by exhaustive search given that the LLC are equally shared among all the vBSs, i.e., $l^k = \lfloor \frac{L}{N} \rfloor \ \forall k$. This benchmark provides an upper bound of the performance of the approach in [13].
- *"LLC Oracle":* It finds the optimal LLC allocation by exhaustive search given that the CPU cores are equally shared among the vBSs, i.e., $u^k = \lfloor \frac{U}{N} \rfloor \ \forall k$. This benchmark provides an upper bound of the performance of the approach in [14].
- *Eq. Partition:* Both CPU cores and LLC ways are equally shared among the vBSs, i.e., $u^k = \lfloor \frac{U}{N} \rfloor \ \forall k$ and $l^k = \lfloor \frac{L}{N} \rfloor \ \forall k$.
- *Proportional Partition:* Both the CPU cores and the LLC ways are split among the vBS proportionally according to their traffic loads, i.e., $u^k = \lfloor \omega_k \cdot U \rfloor$ and $l^k = \lfloor \omega_k \cdot L \rfloor$, where

$$\omega_k = \frac{d^{\mathrm{UL},k} + d^{\mathrm{DL},k}}{\sum_j d^{\mathrm{UL},j} + d^{\mathrm{DL},j}}. \tag{6}$$

We would like to highlight that both the "CPU Oracle" and "LLC Oracle" represent adaptations of state-of-the-art solutions from existing literature [13], [14]. Additionally, the "Eq. partition" and "Proportional partition" approaches are introduced as heuristics, which serve to demonstrate that simplistic non-AI-based methods are insufficient for solving our problem. Specifically, the "Proportional partition" heuristic is based on the premise that vBSs with higher computational loads are likely to consume greater computing resources. This assumption is empirically validated in Section III (Figs. 8 and 9).

For this evaluation, we generate $5 \cdot 10^4$ random global contexts for a random number of vBSs $N \in [2, 5]$. We then apply the different solutions and compute the empirical CDF (ECDF) of the obtained results. Fig. 23 shows the ECDF of the power consumption attained by each solution, considering 2 (left) and 3 (right) vBSs, with 12 CPU cores and 12 LLC ways. In this setting, we observe no bit loss for any of the benchmarks, indicating that we are addressing Problem 1 (i.e., minimizing energy consumption). Our solution closely approaches the Oracle (optimal solution). In particular, AegisRAN achieves an average of 87% of the energy gains of the Oracle, compared to the baseline Eq. Partition. Additionally, the CPU Oracle is also very close to the optimal solution, suggesting that CPU cores are the key resource in this scenario. This is because CPU core allocation has a greater impact on energy consumption compared to LLC allocation, as shown in Figs. 15 and 16. We also observe that both Eq. Partition and LLC Oracle consume significantly more energy due to suboptimal CPU allocation. While Proportional partition shows improvements only with 3 vBSs, it remains distant from our approach, demonstrating the ineffectiveness of simple heuristic approaches for this problem.

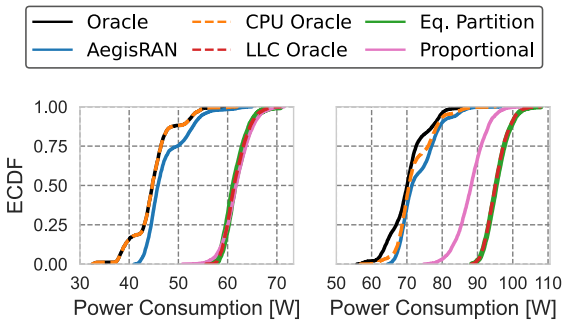


Fig. 23. Power consumption for 2 (left) and 3 (right) vBSs with random contexts and a computing platform with over-dimensioning of resources: 12 CPU cores and 12 LLC ways.

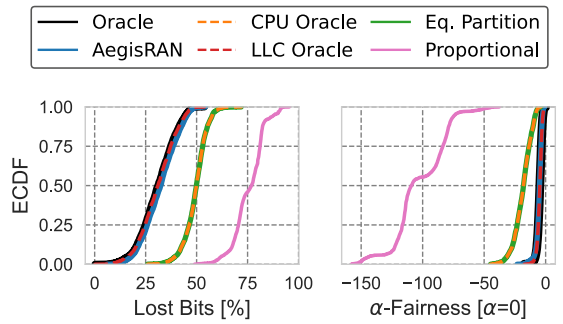


Fig. 24. Lost bits (left) and throughput fairness (right) in a scenario with 5 vBSs with random contexts and a computing platform with under-dimensioning of resources: 10 CPU cores and 8 LLC ways for 5 vBS.

We now evaluate a setting where the computing resources are insufficient to satisfy the demand (Problem 2). In this case, the objective is twofold: (i) minimize throughput loss; (ii) maximize throughput fairness across vBSs. To this end, we consider 5 vBSs and a computing platform with 10 CPU cores and 8 LLC ways.

On the one hand, Fig. 24 (left) shows the ECDF of lost bits for the different benchmarks. We observe that our solution achieves the same bit loss rate than the Oracle. Furthermore, the LLC Oracle also yields the same result, indicating that LLC ways are the key resource in this scenario. When resources are saturated, CPU cores are equally split among the vBSs, a choice common to all solutions. In contrast, LLC allocation significantly impacts bit loss and is highly dependent on the vBS context. Consequently, we observe significant differences between approaches with trivial LLC allocation (Eq. Partition and CPU Oracle) and those optimizing it (AegisRAN and LLC Oracle).

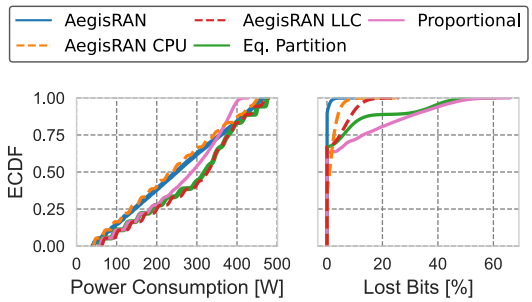


Fig. 25. Power consumption (left) and lost bits (right) for a variable number of vBSs $N \in [2, 20]$ with random contexts and 40 CPU cores and 80 LLC ways.

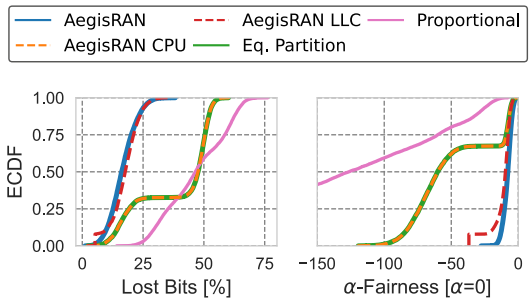


Fig. 26. Lost bits (left) and throughput fairness (right) for a variable number of vBSs $N \in [16, 20]$ with random contexts and 40 CPU cores and 35 LLC ways.

In particular, AegisRAN achieves a mean improvement of 17% with respect to CPU Oracle and Eq. Partition; and a mean improvement of 43% compared to Proportional. Moreover, we note that LLC allocation does not follow a linear pattern, as the Proportional approach performs worse than Eq. Partition. On the other hand, Fig. 24 (right) shows the ECDF of the aggregated α-fairness function (1) of the lost bits. We observe that AegisRAN, along with Oracle and LLC Oracle, achieves the highest fairness. As noted previously, solutions optimizing LLC allocation provide higher fairness in lost bits. Specifically, our solution provides a mean improvement of 70% in the α-fairness compared to CPU Oracle and Eq. Partition; and a mean improvement of 95% with respect to Proportional.

In conclusion, this section demonstrates that the objective of our problem can vary depending on the specific setting (Problems 1 or 2). Importantly, our algorithm seamlessly handles all these settings without requiring any modification or retraining. This adaptability is attributed to the design of the reward function in (5), which is also utilized by all the Oracles in this section.

D. Large-Scale Scenarios

Finally, we evaluate AegisRAN in large-scale deployments with up to 20 vBSs on the same computing node. The algorithm is trained as before, but Oracles are infeasible due to the vast solution space ($\sim 10^{36}$ possibilities). Instead, we introduce two new benchmarks:

- *AegisRAN CPU:* This is a variation of our proposed framework that only configures the CPU allocation, while the LLC ways are equally partitioned, similar to the CPU Oracle.
- *AegisRAN LLC:* This is another variation of our framework that only configures the LLC allocation, while the CPU cores are equally partitioned, similar to the LLC Oracle.

We first evaluate a scenario with a random number of vBSs, $N \in [2, 20]$, with random contexts in a computing platform equipped with 40 CPU cores and 80 LLC ways. Fig. 25 shows the ECDF of the power consumption (left) and the lost bits (right). On the right-hand side, we observe that AegisRAN is the only solution that achieves near-zero lost bits. This highlights the importance of jointly assigning CPU and LLC resources. The benchmarks based on our learning framework (AegisRAN CPU and AegisRAN LLC) cannot find a solution without bit loss due to the trivial assignment of one of the resources. In particular, AegisRAN provides a 1.4% of mean improvement compared to AegisRAN CPU, 2.4% of mean improvement compared to AegisRAN LLC. For simpler solutions (Eq. Partition and Proportional), the bit loss rate is even higher (6% mean decrease of bit loss rate for Eq. Partition and 8.4% for Proportional.). Moreover, AegisRAN not only provides the lowest bit loss ratio but also a very low power consumption (Fig. 25, left). In detail, our solution shows a mean improvement of 16% in the power consumption compared to Eq. Partition and only 5% higher power consumption than AegisRAN CPU. Note that achieving this balance is challenging, as higher bit loss is typically associated with lower power consumption.

Finally, we evaluate a scenario with a clear under-provisioning of computing resources, specifically 40 CPU cores and 35 LLC ways, for up to 20 vBSs. Fig. 26 shows the ECDF of the lost bits (left) and its α-fairness (right). We observe that our solution minimizes the bit loss rate compared to other benchmarks. Specifically, AegisRAN provides a mean improvement of 0.7% compared to AegisRAN LLC, 22% compared to AegisRAN CPU and Eq. Partition, and 30% compared to Proportional.

The remaining conclusions are similar to the smaller scenario (Fig. 24, left): approaches optimizing LLC efficiently (AegisRAN and AegisRAN LLC) achieve better results than those with trivial LLC allocation (AegisRAN CPU, Eq. Partition, and Proportional). Moreover, we observe in Fig. 26 (right) that AegisRAN achieves the highest fairness. That is, the mean improvement of our solution in terms of fairness is 18% compared to AegisRAN LLC, 64% compared to AegisRAN CPU and Eq. Partition, and 90% compared to Proportional. The gap between AegisRAN and AegisRAN LLC indicates that as the problem size grows, the joint allocation of CPU and LLC resources becomes increasingly important for performance.

VII. Related Work

There are many works in the literature resource allocation in RANs [49], [50], [51]. One particular approach aiming at optimizing the rewource waste due to over-provisioning is BBU pooling [52], [53]. BBU pooling aims to run RAN workloads in a cloud or data center environment. This approach enables more efficient resource management as the combined workload of multiple base stations is processed collectively, leading to improved resource utilization and operational efficiency. However, BBU pooling is technically challenging due to the need for

low-latency communication links and the constraint of meeting hard processing deadlines.

To overcome the technical challenges associated with BBU pooling, several works, such as [4], [13], [14], [18], [54], [55], [56], have proposed different resource allocation strategies across multiple vRAN instances. To begin with, vrAIn [18] jointly optimizes computing resources and radio policies (e.g., Modulation and Coding Scheme (MCS)) to minimize resource utilization. AIRIC [13] optimizes the number of CPU cores allocated to vRAN instances, considering resource contention to reduce energy consumption. MemorAI [14] optimizes the Last-Level Cache allocation to vRAN instances for energy efficiency. Recently, Aquifer [54] developed a solution to schedule transparently vRAN workloads at a microsecond-scale scheduler sharing the same computing resources. CloudRIC [4] develop a solution to allocate heterogeneous resources (i.e., GPU, CPU, or ASIC processors) to vRAN workloads to minimize energy consumption. The authors in [55] also take a similar direction than CloudRIC but they consider a multi-agent scenario where they offload the vRAN workloads opportunistically to HAs. Finally, the authors in [56] develop a compute policy for assigning the base station data loads to O-Cloud servers balancing energy savings with performance.

On the other hand, orthogonal to our solution, we highlight other works optimizing the vRAN processing task pipeline. Agora [12] exploits parallelism in vRAN tasks to facilitate massive MIMO baseband process ng. At the same time, Nuberu [15] redesigns the classical pipeline to make vRANs resilient under performance degradation via predictive tools and control lo ps. Finally, we present solutions that develop a framework so that vRAN can share the underlying resources with other processes (i.e., ML workloads). Concordia [11] develops a framework so that vRAN workloads can share the computing resources with different workloads to increase the total infrastructure utilization. YinYangRAN [5] develops a framework so that vRANs can share GPU resources to run ML workloads co-located with vRAN workloads.

VIII. Conclusions and Future Work

In this work, we have addressed the problem of computing resource assignment in vRANs with the goal of reducing energy consumption while maximizing reliability. We first conducted an exhaustive experimental analysis to characterize the impact of computing resource allocation on energy consumption and performance as a function of the network context. As a result, we identify several challenges: First, the optimal computing resource allocation is difficult to obtain as it depends on the context of each vBS (e.g., traffic load, channel quality) in a non-trivial and non-linear way. Second, a suboptimal resource assignment can cause higher overall energy consumption or, even worse, the reduction of users' QoS. Third, the solution space of our problem is huge in real-world deployment, making it difficult for traditional approaches to be effective. Based on the obtained insights, we propose AegisRAN, a data-driven framework for fair and energy-efficient computing resource allocation. At its core, our solution leverages a discrete-SAC algorithm combined with several techniques such as multi-step decision-making, action masking, digital twin-based training, and a tailored reward signal that addresses the dual-objective of the problem and mitigates feedback sparsity. We evaluate AegisRAN in different scenarios with varying size and complexity. The results show that AegisRAN achieves near-optimal performance and offers high flexibility across diverse network contexts and varying numbers of vBS instances.

Future work can focus on improving the adaptability of AegisRAN in real-world deployments, where performance degradation may occur due to discrepancies between the training environment and the actual operating conditions. To address these challenges, adaptive learning mechanisms can be implemented, enabling the model to continuously learn from data collected in real-time from the deployed network. One approach to enhance adaptability involves generating a dynamic dataset from the operational network, which could then be used to re-train the model and adapt it to the current conditions. Meta-learning is a promising strategy for this purpose, as it allows the model to quickly adjust to new traffic patterns with minimal retraining [57]. Additionally, offline-to-online reinforcement learning offers a viable solution, where the model is initially pre-trained on an offline dataset and later fine-tuned with data gathered in the field [58]. These methods would significantly improve the robustness of AegisRAN, ensuring its performance remains optimal even as network conditions evolve.

References

[1] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the virtualized RAN ecosystem," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 96–103, Dec. 2021.

[2] S. Stanley, "5G Transport: A 2021 heavy reading survey," White Paper, Feb. 2022. [Online]. Available: https://eu-assets.contentstack.com/v3/assets/blt23eb5bbc4124baa6/bltabcf066af86e987d/64d5e898777bc203c036b136/Heavy-Readings-Accelerating-Open-RAN-Platforms-Operator-Survey.pdf

[3] RCR Wireless News, "From greenfield to brownfield: Open RAN in 2022 (with large scale carrier commitments in place, what's next for the open RAN ecosystem?)," *Editorial Report*, Oct. 2021.

[4] L. L. Schiavo et al., "Cloudric: Open radio access network (O-RAN) virtualization with shared heterogeneous computing," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, 2024, pp. 558–572.

[5] L. L. Schiavo, J. A. Ayala-Romero, A. Garcia-Saavedra, M. Fiore, and X. Costa-Perez, "YinyangRAN: Resource multiplexing in GPU-accelerated virtualized RANs," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 1–10.

[6] J. Xing, J. Gong, X. Foukas, A. Kalia, D. Kim, and M. Kotaru, "Enabling resilience in virtualized RANs with atlas," in *Proc. 29th Annu. Int. Conf. Mobile Comput. Netw.*, 2023, pp. 1–15.

[7] A. Rao, "Samsung and AWS Power Innovation in Virtualized RAN Networks on Public Cloud," 2024. Accessed: Sep. 2024. [Online]. Available: https://aws.amazon.com/cn/blogs/industries/samsung-and-aws-power-innovation-in-virtualized-ran-networks-on-public-cloud/

[8] V. Bahl, "Scalable management of virtualized RAN with Kubernetes," 2024. Accessed: Sep. 10, 2024. [Online]. Available: https://azure.microsoft.com/en-us/blog/scalable-management-of-virtualized-ran-with-kubernetes/

[9] S. Dasgupta and D. Ayyadevara, "Reimagining Radio Access Networks with Google Cloud," 2024. Accessed: Sep. 10, 2024. [Online]. Available: https://cloud.google.com/blog/topics/telecommunications/google-cloud-reimagines-radio-access-networks

[10] P. Lédl et al., "Reimagining radio access networks with Google cloud," 2024. Accessed: Sep. 10, 2024. [Online]. Available: https://cloud.google.com/blog/topics/telecommunications/google-cloud-reimagines-radio-access-networks

[11] X. Foukas and B. Radunovic, "Concordia: Teaching the 5G vRAN to share compute," in *Proc. ACM SIGCOMM 2021 Conf.*, 2021, pp. 580–596.

[12] J. Ding et al., "Agora: Real-time massive MIMO baseband processing in software," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, 2020, pp. 232–244.
[13] J. X. S. Lozano, A. Garcia-Saavedra, X. Li, and X. C. Perez, "AIRIC: Orchestration of virtualized radio access networks with noisy neighbours," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 2, pp. 432–445, Feb. 2024.
[14] E. S. Hidalgo et al., "MemorAI: Energy-efficient last-level cache memory optimization for virtualized RANs," *IEEE Int. Conf. Mach. Learn. Commun. Netw.*, 2024, pp. 25–30.
[15] G. Garcia-Aviles et al., "Nuberu: Reliable RAN virtualization in shared platforms," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 749–761.
[16] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. IEEE 29th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2018, pp. 1827–1832.
[17] C. Zhang, M. Fiore, and P. Patras, "Multi-service mobile traffic forecasting via convolutional long short-term memories," in *Proc. IEEE Int. Symp. Meas. Netw.*, 2019, pp. 1–6.
[18] J. A. Ayala-Romero et al., "vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, pp. 1–16.
[19] 3GPP, "5G; NG-RAN; architecture description," Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.401, May, 2024, version 18.1.0.
[20] 3GPP, "5G; NR; User equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone," Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.101, Aug., 2024, version 18.6.0.
[21] J. X. Salvat, J. A. Ayala-Romero, L. Zanzi, A. Garcia-Saavedra, and X. Costa-Perez, "Open radio access networks (O-RAN) experimentation platform: Design and datasets," *IEEE Commun. Mag.*, vol. 61, no. 9, pp. 138–144, Sep. 2023.
[22] I. Gomez-Miguelez et al., "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. 10th ACM WiNTECH*, 2016, pp. 25–32.
[23] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 13–23, 2007.
[24] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *Proc. 4th IEEE Int. Conf. Autonomic Comput.*, 2007, pp. 4–4.
[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[26] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," 2020, *arXiv: 2006.14171*.
[27] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[28] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv: 1910.07207*.
[29] D. Silver et al., "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv: 1712.01815*.
[30] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[31] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv: 1912.06680*.
[32] Y. Xia et al., "Recurrent neural network based scenario recognition with multi-constellation GNSS measurements on a smartphone," *Measurement*, vol. 153, 2020, Art. no. 107420.
[33] D. Hein et al., "A benchmark environment motivated by industrial control problems," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2017, pp. 1–8.
[34] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *J. Cheminformatics*, vol. 9, pp. 1–14, 2017.
[35] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.
[36] H. Rahmandad, N. Repenning, and J. Sterman, "Effects of feedback delay on learning," *Syst. Dyn. Rev.*, vol. 25, no. 4, pp. 309–338, 2009.
[37] B. Han, Z. Ren, Z. Wu, Y. Zhou, and J. Peng, "Off-policy reinforcement learning with delayed rewards," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8280–8303.
[38] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv: 2005.01643*.
[39] Intel, "FlexRAN LTE and 5G NR FEC software development kit modules," May 2019. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/flexran-lte-and-5g-nr-fec-software-development-kit-modules.html
[40] M. Fujii et al., "Base-station equipment with the aim of introducing 3.5-Ghz band TD-LTE," *NTT Docomo Tech. J.*, vol. 18, no. 2, pp. 8–13, 2016.
[41] Z. Liu et al., "KAN: Kolmogorov-Arnold networks," 2024, *arXiv:2404.19756*.
[42] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. Berlin, Germany: Springer, 1998, pp. 55–69.
[43] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[44] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 349–359, Mar. 1997.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[46] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[47] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," 2018, *arXiv: 1812.11103*.
[48] J. W. Mock and S. S. Muknahallipatna, "A comparison of PPO, TD3 and SAC reinforcement algorithms for quadruped walking gait generation," *J. Intell. Learn. Syst. Appl.*, vol. 15, no. 1, pp. 36–56, 2023.
[49] A. Mohajer, J. Hajipour, and V. C. Leung, "Dynamic offloading in mobile edge computing with traffic-aware network slicing and adaptive TD3 strategy," *IEEE Commun. Lett.*, vol. 29, no. 1, pp. 95–99, Jan. 2025.
[50] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "EdgeBOL: A Bayesian learning approach for the joint orchestration of vRANs and mobile edge AI," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2978–2993, Dec. 2023.
[51] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 361–376, Feb. 2020.
[52] A. Arfaoui, S. Hamouda, L. Nuaymi, and P. Godlewski, "Minimization of delays in multi-service cloud-RAN BBU pools," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf.*, 2017, pp. 1846–1850.
[53] W. Al-Zubaedi and H. S. Al-Raweshidy, "A parameterized and optimized BBU pool virtualization power model for c-RAN architecture," in *Proc. IEEE 17th Int. Conf. Smart Technol.*, 2017, pp. 38–43.
[54] Y. Jia et al., "Aquifer: Transparent microsecond-scale scheduling for vRAN workloads," *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3171–3184, Nov./Dec. 2024.
[55] J. A. Ayala-Romero et al., "Mean-field multi-agent contextual bandit for energy-efficient resource allocation in vRANs," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 911–920.
[56] F. Aslan et al., "Fair resource allocation in virtualized o-RAN platforms," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 1, pp. 1–34, 2024.
[57] B. Huang, F. Feng, C. Lu, S. Magliacane, and K. Zhang, "AdaRL: What, where, and how to adapt in transfer reinforcement learning," 2021, *arXiv:2107.02729*.
[58] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

Ethan Sanchez Hidalgo received the BSc degree in mathematics from the Autonomous University of Barcelona, in 2023 and currently works as a research scientist with the i2CAT Foundation, Barcelona. He previously interned at NEC Laboratories Europe, where he contributed to research on multi-agent reinforcement learning for energy-efficient 5G networks. He has actively participated in several EU-funded projects, including SNS-JU BeGREEN and SNS-JU ORIGAMI. He is pursuing a master's in Quantum Science and Technology, and his research interests include machine learning for telecommunications, vRAN optimization, and quantum communications.

Jose A. Ayala-Romero (Member, IEEE) received the PhD degree from the Technical University of Cartagena, Spain, in 2019. Then, he joined Trinity College Dublin, Ireland, as a post-doctoral researcher. From 2021 to 2022, he was a senior data scientist with Huawei Ireland Research Center, Ireland. Currently, he is a senior researcher with NEC Laboratories Europe, Germany. His research interests include the application of machine learning to mobile network problems.

Josep Xavier Salvat Lozano (Member, IEEE) received the PhD degree from the Technical University of Kaiserlautern, in 2022 and he currently works as senior research scientist in the 6G Network group with NEC Laboratories Europe, Heidelberg. He worked as reviewer of several international scientific conferences and journals, including *IEEE Transactions on Mobile computing*, IEEE ICC, and *Computer Communications Journal* and has actively participated in several EU-founded projects, including H2020 5Growth, H2020 Daemon and SNS-JU BeGREEN. His research interests lie in the application of machine learning to real-life computer communications systems, including resource allocation and energy efficiency problems.

Andres Garcia-Saavedra received the PhD degree from the University Carlos III of Madrid (UC3M), in 2013. Then, he joined Trinity College Dublin (TCD), Ireland, as a research fellow. Since July 2015, he has been with NEC Laboratories Europe, where he is currently a Principal Research Scientist. He has published with top research venues, such as IEEE INFOCOM, ACM SIGMETRICS, ACM MobiCom or ACM MobiSys, and holds several patents. His research interests include the application of fundamental mathematics to real-life wireless communication systems. He is a SNS JU Technology Board Member. He has served on the Program Committee and Editorial Team of several conferences and journals, such as IEEE INFOCOM, ACM MobiSys, *IEEE Transactions on Wireless Communications* or *IEEE Transactions on Mobile Computing*.

Xavier Costa Perez (Senior Member, IEEE) received the MSc and PhD degrees in telecommunications from the Polytechnic University of Catalonia (UPC), in Barcelona and was the recipient of a national award for the PhD thesis. He is ICREA research professor, scientific director with the i2cat Research Center and head of 6G R&D with NEC Laboratories Europe. His team generates research results which are regularly published at top scientific venues, produces innovations which have received several awards for successful technology transfers, participates in major European Commission R&D collaborative projects. He has held multipe leadership positions both in industry and research organizations such as deputy general manager, chief researcher, technology board member and scientific advisory board member. As a standards delegate, he contributed to multiple standardization bodies (e.g., IEEE 802.11, 802.16, WiFi Alliance, 3GPP,...) and was recognized in several standards as a top contributor. He has served on the Organizing Committees of several conferences (including ACM MOBICOM, IEEE INFOCOM, WCNC, and Greencom), published papers of high impact and holds about 100 granted patents. He has served as editor with *IEEE Transactions on Mobile Computing (TMC)*, *IEEE Transactions on Communications (TCOM)* and *Elsevier Computer Communications journals (COMCOM)*.