Abstract—Flying and ground-based cars require various services such as autonomous driving, remote piloting, infotainment, and remote diagnosis. Each service requires specific Quality of Service (QoS) and network features. Therefore, network slicing can be a solution to fulfill the requirements of various services. Some services, such as infotainment, may have similar requirements to serve flying and ground-based cars. Therefore, some slices can serve both kinds of cars. However, when network slice resource sharing is too aggressive, slices can not meet QoS requirements, where resource under-provisioning causes the violation of QoS, and resource over-provisioning causes resources under-utilization. We propose two closed loops for managing RAN slice resources for cars to address these challenges. First, we present an auction mechanism for allocating Resource Block (RB) to the tenants who provide services to the cars using slices. Second, we design one closed loop that maps slices and services of tenants to Open Distributed Units (vO-DUs) and assigns RB to vO-DUs for management purposes. Third, we design another closed loop for intra-slices RB scheduling to serve cars. Fourth, we present a reward function that interconnects these two closed loops to satisfy the time-varying demands of cars at each slice while meeting QoS requirements in terms of delay. Finally, we design distributed deep reinforcement learning approach to maximize the formulated reward function. The simulation results show that our approach satisfies more than 90% vODUs resource constraints and network slice requirements.

Index Terms—Open radio access network, network slicing, urban aerial mobility, connected car systems.

I. INTRODUCTION

A. Background and Motivations

FLYING cars were recently introduced in Urban Air Mobility (UAM) as an innovative concept for the transportation of people and goods [1]. Furthermore, the authors in [2] discussed the feasibility of passenger drones and flying car technologies in tomorrow’s transportation. The difference between a flying car and a passenger drone is that the flying car can carry passengers on both the ground and in the air, while a drone can only be used in the air. In other words, the flying car would be a vehicle where the driver/pilot/AI system can drive the vehicle in its car configuration mode to a Station Hub (SH), reconfigure the vehicle to airplane mode, and then fly to a destination SH. Flying cars are expected to become a reality in smart cities. Some essential projects for flying cars have recently been introduced, such as electric Vertical Take-Off and Landing (eVTOL) and Personal Aerial Vehicles (PAVs). The cruising altitude of the flying cars can reach around 300 meters. The flying cars can fly at very high speeds, up to 300 km/h. A heterogeneous network of base stations, high-altitude platform stations (HAPS), and satellites can be used for connecting flying cars. In this case, multi-connectivity or handover from one network to another should be considered. However, in this work, we focus on Near-Ground Spaces (NGS) [3], where flying cars do not occupy or compete for high-altitude spaces currently occupied by air traffic for long-distance transfer. NGS is suitable for urban mobility scenarios. In other words, we consider Vertical Take-Off and Landing (VTOL) model, where flying cars can take off and land vertically. In this mode, flying cars do not need runways for either take-off or landing [3]. However, connecting flying cars using existing base stations in the cellular network without antennas adjustment is almost infeasible because antenna propagate toward the ground [4]. As discussed in [5], base stations can have additional antennas pointing toward the sky with omnidirectional coverage to address this challenge. Therefore, flying cars can operate within the coverage domains of ground base stations [6]. In other words, the ground base stations can serve both ground-based cars and flying cars. In urban mobility, multiple flying cars will transport passengers from different sources to the destination SHs. Therefore, more SHs for flying cars will be required. Flying cars need to be connected to SHs and ground-based vehicles. Ground-based vehicles can serve various transit functions, from flying cars’ SHs to urban services such as banks, restaurants, and clinics. However, each car may need different services of different QoS and connectivity requirements such as high-definition maps, remote piloting, autonomous driving, and remote diagnosis. Therefore, network slicing that enables virtualized networks on the same physical network can be an appropriate solution to fulfill the diverse requirements for services of the flying and ground-based cars. However, such heterogeneity of services per car cannot be effectively managed and efficiently mapped onto
one slice. We need a slice per service. Also, some slices such as infotainment slices may serve flying and ground-based cars.

Several prototypes have been designed for network slicing at the core network [7]. However, Radio Access Network (RAN) slicing is still in the early stages. RAN has experienced some transformations to increase deployment flexibility and network dynamics. One of the proposed RAN modifications is Cloud RAN (C-RAN), which modified the traditional base station structure by separating the base station into the Remote Radio Unit (RRU) and Baseband Processing Unit (BBU). This transformation creates a fronthaul interface between the RRUs and BBU. However, C-RAN considers proprietary centralized BBU and fronthaul. Virtual RAN (vRAN) has therefore, been to overcome this issue by considering BBU as software running on a generic hardware platform, where software and hardware components can be supplied by different vendors [8]. Similarly to C-RAN, vRAN considers proprietary RRU and fronthaul interface. The recent RAN transformation is the Open Radio Access Network (O-RAN) that replaces the legacy, proprietary fronthaul interface and RRU with an open fronthaul interface and Open Radio Unit (O-RU). This enables radio units from one vendor to interoperate with units from other vendors [9]. With O-RAN and machine learning, it is possible to perform fast real-time monitoring of RAN traffic, and radio conditions on the car level [10]. Then, O-RAN applications can use historical traffic data to predict the handover anomalies and apply a proper measurement to mitigate identified anomalies. Therefore, this work focuses on RAN slicing and considers the Open Radio Access Network (O-RAN) as a use case. However, O-RAN is not restrictive. O-RAN uses distributed intelligent controllers, where Near-Real-Time RAN Intelligent Controller (Near-RT RIC) enables training, testing, utilization, and updating machine learning. In contrast, Non-Real-Time RAN Intelligent Controller (Non-RT RIC) enables machine learning functionalities for policy-based guidance of applications and features. In O-RAN, there are three types of control loops. Loop 1 operates at a time scale of less than 10 msec. Loop 1 can be employed for Resource Block (RB) scheduling in Transmission Time Interval (TTI). Loop 2 operates at Near-RT RIC within the range of 10-500 msec. Loop 2 can be appropriate for resource optimization. In Non-RT RIC, Loop 3 operates at a time scale greater than 500 msec. Loop 3 can be employed for policies-based resource orchestration. Also, O-RAN supports O-RAN Central Unit Control Plane (O-CU-CP) and O-RAN Central Unit User Plane (O-CU-UP), O-RAN Central Units (O-CU-CP and O-CU-UP) interfaces with O-RAN Distributed Unit (O-DU) to provide services to edge devices via O-RAN Radio Units (O-RUs).

B. RAN Slicing Challenges in Dealing With Car Services

Considering slicing in RAN and O-RAN, the following are key challenging issues for serving the cars:

- Heterogeneity of services per each car such as ultra-low latency connectivity for autonomous driving/pilot, a high data rate for infotainment, and an extremely high connection density for remote diagnosis. One slice can not meet all required network features of the services needed by the car.
- High mobility of cars requires fast decisions in radio resource allocation. Therefore, a closed loop with real-time analytics is needed for taking appropriate and quick radio resource allocation decisions.
- Satisfy slice requirements with high efficiency in finite radio resources. If radio resource sharing is too aggressive, the slices can not meet the required QoS for car services, and this can cause services to degrade.
- There is a lack of literature discussing network slicing for a heterogeneous scenario of flying and ground-based cars.

C. Contributions

To address the aforementioned challenges, this work proposes two-level closed loops for managing RAN slice resources serving flying and ground-based cars. Our key contributions are summarized as follows:

- We propose an auction mechanism for allocating RBs to the tenants who provide services to flying and ground-based cars using slices. We assume the RBs are limited, and tenants should compete to get them.
- We propose one closed loop to create the slices associated with the services of tenants to vO-DUs for RBs scheduling purposes. We consider virtualized O-DU, where vO-DU is virtualized instance of O-DU.
- We propose another closed loop for intra-slices RB scheduling to serve flying and ground-based cars. Also, we design a communication planning approach that supports the proposed closed loop in RB scheduling.
- We formulate an optimization problem that joins two closed loops and considers QoS fulfillment in delay. However, an optimization problem leads to a stationary proximal solution, which is inappropriate for continuing tasks of slice resource auto-scaling. Therefore, we change the optimization problem to the reward functions. Then, we design a distributed Deep Reinforcement Learning (DRL) approach that enables two closed loops to exchange data to maximize the formulated reward functions.

The rest of this paper is organized as follows. Section II discusses the related work, while Section III presents the system model. In Section IV, we present initial resource allocation, while Section V demonstrates the problem formulation. We discuss the proposed solution in Section VI. Section VII presents a performance evaluation. We conclude the paper in Section VIII.
authors in [11] proposed RAN inter-slice resource partitioning and allocation as a two-dimensional knapsack problem that facilitates inter-slice radio resource sharing. Then, they developed a heuristic algorithm to handle the formulated problem sequentially. Since network slicing is a continuing and not stationary task, a machine learning-based solution is preferable rather than an optimization approach. The authors in [12] discussed multiple resources sharing fairness between slices. They formulated a versatile optimization framework and applied the ordered weighted average operator to handle the formulated problem. However, an optimization problem leading to progressive convergence toward an optimum slice resource allocation is insufficient for a dynamic network environment. The authors in [13] proposed a QoS framework that uses Defined Networking (SDN) and Network Function Virtualization (NFV) in the network slicing to satisfy the QoS requirement of different 5G application scenarios. However, the processing time for the proposed RAN slicing is still large to meet the 5G lower latency requirement. In our previous work [14], we proposed radio resource allocation using matching theory and auctions in a visualized wireless environment. However, [14] needs to be improved to accommodate a dynamic network environment. In [15], the authors used deep learning and Lyapunov stability theories to enable network slicing in dynamic and heterogeneous networks. The authors in [16] proposed a decentralized DRL approach for edge computing networks that learns demands for network slices and optimally orchestrates end-to-end resources. In [17], the authors proposed a slice resource orchestration framework that uses regression trees to classify and predict network traffic to satisfy the slices’ QoS requirements. The authors in [18] used DRL to perform RB allocation to the RAN slice, where each DRL agent manages one network slice. In [19], the authors presented an energy-efficient DRL-based solution for power and radio resource allocation in RAN slices. However, the proposed approaches in [11], [14], [15], [16], [17], [19] are not applicable to the O-RAN environment due to the distributed O-RAN elements. In [20], the authors presented a closed loop deployment for automatic slicing assurance in 5G RAN to meet the SLA of each deployed slice. Furthermore, the authors in [21] discussed three closed loops to coordinate services for network slices. However, these related works in [20], [21] do not provide detailed mathematical models. Also, their papers contain no performance evaluation to show the effectiveness of their proposed closed loops. Therefore, modeling and designing interconnected closed loops in O-RAN for network slicing are still in their infancy, which motivates us to carry out this research.

**Network slicing and closed loop for serving vehicles:** In [22], the authors discussed vertical industries with multiple use cases, where each use case is associated with diverging services and connectivity requirements. They proposed Vehicle to Everything (V2X) communication slices as a specific use case for ground-based cars. However, a performance evaluation is needed for the proposed use case. The authors in [23] discuss AerialSlice as a network slicing framework to handle Unmanned Aerial Vehicle (UAV) applications. Then, they presented a proof-of-concept deployment of AerialSlice.

In [24], relying on the testbed, the authors proposed a new 5G network slicing approach that provides connectivity to cars and trains using UAVs. Their proposal uses UAVs as base stations. The author in [25] discussed off-line RL for allocating resources to RAN slices that serve enhanced mobile broadband (eMBB) and V2X services. The authors in [26] discussed a two-layer constrained RL algorithm to satisfy the different QoS requirements for the Internet of Vehicles services, where multiple slices are implemented at roadside units. Still, used RAN slicing in [25], [26] can not be applied to new RAN architecture, such as O-RAN, where RAN functions are distributed. In [27], the authors proposed RAN slicing for vehicle-to-infrastructure communication as an optimization problem. Then, they used a heuristic algorithm to handle the formulated problem. However, a dynamic driving environment requires low complexity solutions than the traditional heuristic optimization approach. The authors in [28] proposed a cooperative crowdsensing system for controlling respiratory viral disease using flying and ground vehicles. The authors in [23], [24], [28] used UAVs as flying vehicles, which can only be used in the air and carry no passengers. In contrast, our approach considers flying cars that can be used in the air for carrying passengers in NGS that is not conflicting with high-altitude air spaces. Also, the services and bandwidth requirements of flying cars and UAVs are different due to the possibility of carrying passengers in the flying cars. Regarding the aforementioned works in [22], [23], [24], [25], [26], [27], [28], the problem of network slicing that considers a heterogeneous scenario of flying and ground-based cars, where closed loops are used to manage RAN slices serving cars, has not yet been tackled in the literature.

**III. System Model**

Our system model is depicted in Fig. 1, and the summary of our key notations is available in Table I. We consider
\( \mathcal{V} = \{1, \ldots, V\} \) as a set of cars. In the cars, it includes both flying cars \( \mathcal{V}_a \) and ground-based cars \( \mathcal{V}_g \), such that \( \mathcal{V} = \mathcal{V}_a \cup \mathcal{V}_g \). Each car \( v \in \mathcal{V} \) can require one or more services such as infotainment content, remote diagnosis, and computation in Multi-Access Edge Computing (MEC) server. We use \( \mathcal{K} = \{1, \ldots, K\} \) as a set of services. Each service \( k \in \mathcal{K} \) needed by car \( v \in \mathcal{V} \) is associated with delay budget \( \tau_{vk} \), where the delay budget is based on 5G QoS Identifier (5QI) defined in [29]. Each car requires a network connection to get service. We assume each car can be connected to O-RU via a wireless network. We consider the Orthogonal Frequency Division Multiple Access (OFDMA) downlink scenario, where O-RU provides wireless connection to certain number of cars. We denote \( \mathcal{M} = \{1, \ldots, M\} \) as a set of O-RUs. In O-RUs, it includes O-RUs of type RSUs (Road-Side Units), which support both O-RU and V2X functionalities.

The O-RUs and vO-DUs belong to Infrastructure Provider (InP), where InP has RBs \( B \) at the cost of \( \Gamma(B) \). We assume that the RBs are divisible for being allocated to the tenants who provide services to cars using the slices. We consider cars are subscribed to the slices of tenants. We denote \( \mathcal{L} = \{1, \ldots, L\} \) as a set of tenants. Each service of tenant can be mapped to specific slice types such as enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). We use \( \mathcal{C} = \{1, \ldots, C\} \) as a set of slices, where each slice manages one service. We use the auction to allocate RBs to the slices associated with the services of tenants. Near-RT RIC gets slice requirements from tenants via RAN Network Slice Subnet Management Function (NSSMF) and performs RBs allocation. In the near real-time loop (loop 2 works in 10 ms to 1 s), Near-RT RIC assigns RBs and slices to vO-DUs for management purposes. In the real-time loop (less than or equal to 10 ms), each slice at vO-DU allocates RBs to cars. Here, we consider slicing at the core network and Data Network (DN) to be outside the scope of this paper. Also, we consider slice-aware Access & Mobility Management Function (AMF) and O-CU-UP selection as future work.

\[ \text{IV. INITIAL SLICE AND RESOURCE BLOCK ALLOCATION} \]

\textbf{A. Resource Block Allocation to the Tenants} We consider RBs are limited. The tenants, who provide service to vehicles using slices, should compete to get RBs from InP. Therefore, InP makes RBs \( B \) available to \( L \) tenants of \( K \) services for buying via auction. In the auction, we consider InP as a seller of RBs and multiple tenants \( L \) as buyers.

The workflow of Auction for RB (ARB) is presented in Fig. 2 and summarized as follows:

- **Step 1:** The InP announces available RBs for auction to tenants \( L \) and reserve price \( b_p \) per unit of RB \( b \). A reserve price \( b_p \) represents the minimum price that InP would accept from tenants per unit of RB \( b \).
- **Step 2:** In receiving available RBs for auction and reserve price \( b_p \), each tenant \( l \in \mathcal{L} \) of service \( k \) prepares and submits a bid \((J^{l,k}_b, n^{l,k}_b)\) to InP as demand for RBs. \( J^{l,k}_b \) represents bid per unit of RB \( b \) for service \( k \) and \( n^{l,k}_b \) represents the initial number of RB \( b \) needed for service \( k \).
- **Step 3:** InP collects all of the bids from the tenants and evaluates them. For \( J^{l,k}_b \geq b_p \), the InP sorts the bids in descending order. Then, InP allocates the RBs to tenants starting with the tenant with the highest bidding values. The InP calculates the payment \( J^{l,k}_b, n^{l,k}_b \) that each winning tenant \( l \) of service \( k \) has to pay for RBs. Then, the InP declares the winning tenants and the winning price \( J^{l,k}_b, n^{l,k}_b \).
ARB helps the InP to choose winning tenants that submitted bidding values that maximize its revenue and the social welfare. In ARB, we consider that each tenant \( l \in L \) submits its bid for RB \( b \in B \) without knowing the bidding values of other tenants. Also, each tenant \( l \in L \) can submit one bid per service. We consider that each tenant \( l \) has its own valuation for RB \( b \) denoted \( \Upsilon_{l,k}(n_{b}^{l,k}) \). Here, \( \Upsilon_{l,k}(n_{b}^{l,k}) \) is given by:

\[
\Upsilon_{l,k}(n_{b}^{l,k}) = \begin{cases} 
    t_{l,k}^{l,k} n_{b}^{l,k}, & \text{if the tenant } l \text{ participates in ARB,} \\
    0, & \text{otherwise,}
\end{cases}
\]

where \( t_{l,k}^{l,k} \) is the true valuation of tenant \( l \) for service \( k \) that requires RB \( b \). However, when tenant \( l \) does not participate in the ARB, its true valuation is 0. On the other hand, the valuation \( \Gamma(B) \) of the InP is defined using reserved price \( b_{p} \) such that \( \Gamma(B) = B b_{p} \). InP sets \( b_{p} \) that ensures its revenue does not become negative. In other words, its revenue covers its CAPEX and OPEX associated with RBs.

In our action, we choose Vickrey Clarke Groves (VCG) mechanism [30] over other auction mechanisms because VCG mechanism enables welfare maximization of all tenants and guarantees a truthful outcome. VCG enables to achieve better efficiency in RBs allocation and competition between tenants. It allows optimal price \( J_{l,k}^{b} \) that maximizes its revenue and the social welfare. In ARB, we consider that each tenant \( l \) will participate in ARB if and only if

\[
\Upsilon_{l,k}(n_{b}^{l,k}) \text{ is given by:}
\]

\[
\Upsilon_{l,k}(n_{b}^{l,k}) = \arg \max_{J_{l,k}^{b}} \sum_{k \in K} J_{l,k}^{b} n_{b}^{l,k}.
\]

In the VCG, each tenant \( l \) should pay for the damage it may cause on other tenants by participating in the ARB. Therefore, we compute the total valuation \( \Upsilon_{l,k}(n_{b}^{l,k}) \) without each tenant \( l \), where \( \Upsilon_{l,k}(n_{b}^{l,k}) \) is given by:

\[
\Upsilon_{l,k}(n_{b}^{l,k}) = \arg \max_{J_{l,k}^{b}} \sum_{k \in K} J_{l,k}^{b} n_{b}^{l,k}.
\]

From (2) and (3), we can compute the price \( J_{l,k}^{b}(n_{b}^{l,k}) \) for each tenant \( l \) of service \( k \) to pay to InP as follows:

\[
J_{l,k}^{b}(n_{b}^{l,k}) = \Upsilon_{l,k}(n_{b}^{l,k}) - \sum_{j \neq l} J_{j,k}^{b} n_{b}^{l,k}.
\]

Definition 1 (Tenant Utility): In ARB, in which tenant submit a bid \( J_{l,k}^{b}(n_{b}^{l,k}) \), if the tenant \( l \) wins the ARB, it pays \( J_{l,k}^{b}(n_{b}^{l,k}) \) to InP. Otherwise, if tenant \( l \) loses the ARB, it pays nothing. Therefore, the utility \( U_{l,k} \) of any tenant \( l \) of service \( k \) is given by:

\[
U_{l,k} = \begin{cases} 
    J_{l,k}^{b}(n_{b}^{l,k}) - \Upsilon_{l,k}(n_{b}^{l,k}), & \text{if tenant } l \text{ wins ARB,} \\
    0, & \text{otherwise,}
\end{cases}
\]

where \( W \) is the set of the winners. We consider each tenant will participate in ARB if and only if

\[
J_{l,k}^{b}(n_{b}^{l,k}) \geq \Upsilon_{l,k}(n_{b}^{l,k}) \text{ in other words, a tenant will participate in ARB when its utility is not negative.}
\]

Definition 2 (Individual Rationality): ARB is individually rational if and only if no tenant \( l \in L \) receives negative utility, i.e., \( U_{l,k} \) is not negative (\( U_{l,k} \geq 0 \)).

Definition 3 (Truthfulness): ARB is truthful if and only if, for each tenant \( l \in L \), bidding the truth value \( b_{l}^{l,k} = J_{l,k}^{b} \) is the dominant strategy. In other words, bidding \( b_{l}^{l,k} \) that maximizes the utility of each tenant \( l \in L \) given for all possible bidding values is the dominant strategy.

Theorem 1: The ARB is truthful.
Proof: We consider that each tenant \( l \in L \) wins the ARB by submitting its true valuation, i.e., \( b_{l}^{l,k} = J_{l,k}^{b} \). Also, ARB satisfies monotonicity and critical payment conditions of truthful bidding defined in [31].

- Monotonicity: Let us consider a scenario of two tenants \( l \) and \( l' \) submitted bidding values \( J_{l,k}^{b} = J_{l,k}^{b'} = b_{b} \) for service \( k \in K \), where \( J_{l,k}^{b} > J_{l',k}^{b} \). ARB chooses bidding value that maximizes total valuation in descending order of the bidding values. Therefore, \( J_{l,k}^{b} \) will give more chance tenant \( l \) to win ARB over \( J_{l',k}^{b} \) because \( J_{l,k}^{b} > J_{l',k}^{b} \).

- Critical payment: In ARB, the payment of the winner is based on its bidding value and the bidding values of other tenants, where VCG tries to maximize social welfare. The ARB makes tenants \( l \in L \) with maximum bidding value \( J_{l,k}^{b} \) as the winner whatever other bidding values such as \( J_{l',k}^{b} \), and winner \( l \in L \) pays \( J_{l,k}^{b}(n_{b}^{l,k}) \leq J_{l',k}^{b} n_{b}^{l,k} \).

Theorem 2: The ARB is individually rational.
Proof: Considering Definition 2 and individually rational condition defined in [31], ARB becomes individually rational when no tenant receives negative utility. Based on the above Theorem 1 and (5), ARB makes tenants \( l \in L \) with maximum bidding value \( J_{l,k}^{b} \) as the winner whatever other bidding values and pays \( J_{l,k}^{b}(n_{b}^{l,k}) \leq J_{l',k}^{b} n_{b}^{l,k} \). Otherwise, based (5), tenant who does not win ARB receives zero utility (\( U_{l,k} = 0 \)). Therefore, \( U_{l,k} \geq 0 \).

The above ARB can be designed as a Total Revenue Maximization (TRM) problem, where the TRM is expressed as follows:

\[
\max_{x} \sum_{k \in K} \sum_{l \in L} x_{l,k}^{b} n_{b}^{l,k} J_{l,k}^{b}.
\]

subject to:

\[
\sum_{k \in K} \sum_{l \in L} x_{l,k}^{b} n_{b}^{l,k} \leq B, \forall b \in B,
\]

\[
x_{b}^{l,k} J_{l,k}^{b} \geq b_{p},
\]

\[
x_{b}^{l,k} \in \{0, 1\}.
\]

In TRM problem (6), the RBs needed to be allocated to tenants must be less than the total RBs. In (6b), the bidding value of the tenant should be greater or equal to the reserve price of InP. In (6c), we use \( x_{b}^{l,k} \) as a binary decision variable, where \( x_{b}^{l,k} = 1 \) if tenant \( l \) submit bid \( J_{l,k}^{b} \) and wins the auction, and \( x_{b}^{l,k} = 0 \) otherwise.

TRM problem is an Integer Linear Programming (ILP) problem. To handle (6), we propose an algorithm (Algorithm 1)
for Winner and Price Determination. Algorithm 1 is based on the VCG mechanism. The inputs of Algorithm 1 includes a set of tenants \( L \), a set of services \( K \), available RBs \( B \) for auction, vector of bids \( J \), vector of the number of RBs needed \( n \). At line 3, the algorithm initializes the parameters of the auctions including the set of winners \( W \) and set of tenants \( W' \) who do not win the auction. Then, the algorithm performs iterations for winner and price determination until all RBs \( B \) are allocated to the tenants or no more tenants need RBs. The outputs of the Algorithm 1 are set of winning tenants \( W \), vector \( x \) of winning decision variables, and vector of \( J^* \) payments. We assume that \( J^* (n^*_b) \) is the flat price that the tenant \( l \) and InP agreed for RBs of slice associated with service \( k \) during the auction. Once the tenant RB usage passes the initial number of RB \( n^*_b \) requested in the auction, i.e., cap, InP does not stop the tenant service, but InP introduces a flat rate increase described in [32]. However, we consider a flat rate increase to be outside the scope of this paper. Also, the auction is performed outside the closed loops. In other words, the auction helps to get RBs that will be managed using closed loops.

**Theorem 3:** Computational complexity of ARB is \( O(n^2) \)

**Proof:** In Algorithm 1, we have a while loop at lines 4-17 that performs \( n \) iterations for checking submitted bids \( J^* (n^*_b) \geq b_p > 0 \), where \( n \) is the size of the vector \( J \). Inside the while loop, we have another loop at lines 7-16 for allocating RBs to the tenants starting from the tenant with maximum bidding value and this loop takes \( n \) iterations. We have a third loop at lines (19-27) for finding the winners if each tenant with maximum bidding value does not participate in ARB, which takes \( n - 1 \) iterations. The last loop is at lines (28-31) for calculating total evaluation and it takes \( n \) iterations. As result, Algorithm 1 takes \( n^2 + n - 1 \) iterations. In conclusion, the computational complexity of RA is \( O(n^2) \), which is linear time.

### B. RBs Distribution to vO-DUs for Scheduling Purpose

In closed loop two, initially, InP assigns RBs \( B \) to vO-DUs equally such that \( B = \sum_{d=1}^{D} b_d \), where \( b_d = \left[ \frac{B}{D} \right] \) is the RB assigned to each vO-DU \( d \). After the auction, InP creates slices \( C \) associated with \( K \) services at vO-DUs and assigns RBs to slices. However, the auction is not restrictive; other techniques such as proportional allocation [33] can be applied. Then, InP uses round-robin policy [34] to create each slice \( c \in C \) associated with service \( k \in K \) of each winning tenant \( l \) at vO-DU. The round-robin policy cyclically creates slices associated with services to vO-DUs starting from vO-DU 1 such that \( \sum_{k=1}^{K_d} b_c^d \leq b_d \), where \( b_c^d \) is RBs of each slice \( c \) at each vO-DU \( d \) for service \( k \). \( K_d \) represents the number of services at vO-DU \( d \) and \( b_c^d = y_{b,c,k} n^*_b \). Furthermore, we define \( y_{b,c,k} \) as decision variable indicating whether slice \( c \) of service \( k \) has assigned radio resources at vO-DU \( d \), where \( y_{b,c,k} \) is given by:

\[
y_{b,c,k} = \begin{cases} 1, & \text{if slice } c \text{ of service } k \text{ has assigned RBs at vO-DU } d, \\ 0, & \text{otherwise.} \end{cases}
\]  

To ensure that each slice \( c \) of service \( k \) is created at one vO-DU, InP imposes the following constraint:

\[
\sum_{c \in C} y_{b,c,k} \leq 1, \forall d, b, k.
\]  

### C. Intra-Slices RBs Scheduling for Cars

In closed loop one, we consider vO-DUs are connected to O-RUs via wired fronthaul network, where O-RUs serve \( V \) cars available in their coverage areas. Based on chosen numerology \( i \), each RB \( b_c^d \) is partitioned into \( f_{i,d} \), number of sub-bands, indexed by \( F_{i,d} = \{1, 2, \ldots, f_{i,d}\} \) in
calculate the remaining distance $\varsigma^v_m$ of each car $v$ to reach area $\Lambda_m$ covered by each nearby O-RU $m$, where $\varsigma^v_m$ is given by:

$$\varsigma^v_m = \chi^m_v \cos \gamma^m_v.$$  

We use $g^m_v$ as an estimated angle between the trajectory of movement of car $v$ and the line from O-RU $m$. By using $c^v_m$, the RT-SC can compute the probability $p^m_v$ that O-RU $m$ can serve car $v$ using wireless communication such that:

$$p^m_v = \begin{cases} 1, & \text{if } c^v_m = 0 \text{ and } \tau^v_v \leq \tau^v_k, \\ 0, & \text{otherwise.} \end{cases}$$  

When $c^v_m = 0$, the car $v \in V$ reaches the area $\Lambda_m$ covered by O-RU $m$. We define $\tau^v_m$ as the time required by car $v$ to leave the coverage area of O-RU $m$, where $\tau^v_m$ is given by:

$$\tau^v_m = \frac{\Lambda_m}{I_v}.$$  

Here, $I_v$ is the estimated speed of car $v$. When $\tau^v_m \leq \tau^v_k$, the car can easily use O-RU $m$ for wireless communication and meet delay budget $\tau^v_k$. Otherwise, when $\tau^v_m > \tau^v_k$, our approach can select the next O-RU to use that can satisfy the delay budget. In other words, RT-SC collaborates with AMF, where AMF can initiate Xn handover between O-RUs. However, we consider O-RU handover and reliability for flying and ground-based cars as future work. Furthermore, in our approach, there is a relationship between flying and ground-based cars because both types of cars use RBs and are served by the same O-RUs. Therefore, flying and ground-based cars can not always be treated differently. Also, we remind that in flying cars, we can have helicopter-car and airplane-car modes, where the flying cars can navigate in a road environment like traditional ground-based cars [3].

According to Shannon’s theory, the achievable data rate for the car $v$ on the RB $(t_{i,d}^c,k,t_{i,d}^c,k)$ can be written as:

$$R_{t_{i,d}^c,k}^v = \omega_{t_{i,d}^c,k}^m p_{t_{i,d}^c,k}^m \log_2 \left(1 + \frac{\delta_{t_{i,d}^c,k}^v}{\sigma^2_v} \right), \forall v \in V, \tag{15}$$

where $\omega_{t_{i,d}^c,k}^m$ is the bandwidth of the RB with numerology $i$. Then, the data rate of each car $v$ can be computed as:

$$R_v = \sum_{i=1}^4 \sum_{k=1}^{T_{i,d}^c} \sum_{t_{i,d}^c,k} z_{t_{i,d}^c,k}^v R_{t_{i,d}^c,k}^v, \tag{16}$$

where $z_{t_{i,d}^c,k}^v$ is a binary decision variable that indicates whether car $v$ uses RB $(t_{i,d}^c,k,t_{i,d}^c,k)$ of numerology $i$ at O-RU $m$, where $z_{t_{i,d}^c,k}^v$ is given by:

$$z_{t_{i,d}^c,k}^v = \begin{cases} 1, & \text{if } p_{t_{i,d}^c,k}^m = 1 \text{ and RB } (t_{i,d}^c,k,t_{i,d}^c,k) \text{ is allocated to car } v, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

To comply with the requirement of OFDMA system, where each RB $(t_{i,d}^c,k,t_{i,d}^c,k)$ can only be allocated to a single car, we impose the following orthogonality constraint:

$$\sum_{u \in V} z_{t_{i,d}^c,k}^v \leq 1, \forall v, t_{i,d}^c,k \in C_{i,d}.$$  

the frequency-domain and $T_{i,d}^c,k$ number of TTIs, indexed by $\mathcal{T}_{i,d}^c,k = \{1, 2, \ldots, T_{i,d}^c,k\}$ in the time-domain. Therefore, a total $F_{i,d}^c,k \times T_{i,d}^c,k$ number of RBs are available for the service $k$ using numerology $i$. RBs scheduling can be modeled using perfect Channel State Information (CSI). However, in practice, it is challenging to obtain perfect CSI due to some limitations such as delayed feedback. As described in [35], the channel perfect Channel State Information (CSI). However, in practice, using numerology

$$\tilde{h}_{t_{i,d}^c,k}^v = \hat{h}_{t_{i,d}^c,k}^v + \epsilon_{t_{i,d}^c,k}^v,$$  

where $\hat{h}_{t_{i,d}^c,k}^v$ and $\epsilon_{t_{i,d}^c,k}^v$ represent the estimated CSI and estimated error, respectively. Using $\tilde{h}_{t_{i,d}^c,k}^v$, the achievable SNR at the cars $v$ on the RB $(t_{i,d}^c,k,t_{i,d}^c,k)$ becomes:

$$\delta_{t_{i,d}^c,k}^v = \frac{c_{d_{t_{i,d}^c,k}^v}^m}{h_{t_{i,d}^c,k}^v |h_{t_{i,d}^c,k}^v|^2 \tilde{p}_{t_{i,d}^c,k}^m \chi_{d_{t_{i,d}^c,k}^v}^m \sigma^2_v}, \tag{10}$$

where $\tilde{p}_{t_{i,d}^c,k}^m$ is the allocated power to the each RB $(t_{i,d}^c,k,t_{i,d}^c,k)$. $\chi_{d_{t_{i,d}^c,k}^v}^m$ is the distance between the car $v$ and O-RU $m$ and $\sigma^2_v$ is the noise power.

As shown in Fig. 3, due to car mobility, the distance $\chi_{d_{t_{i,d}^c,k}^v}^m$ keeps changing. Therefore, the combination of global navigation satellite systems (GNSS) such as GPS and GLONASS can be applied to find $\chi_{d_{t_{i,d}^c,k}^v}^m$. The same approach was applied in [36], [37], [38]. Furthermore, we consider the distance of a flying car from the earth and the height of the O-RU, where O-RU has antennas pointing toward the sky for aerial coverage to serve flying cars. As described in [5], $\chi_{d_{t_{i,d}^c,k}^v}^m$ for the flying cars can be calculated as follows:

$$\chi_{d_{t_{i,d}^c,k}^v}^m = \sqrt{\eta^m_m^2 + (\eta_v - \eta_m)^2}, \forall v \in V_c, \tag{11}$$

where $\eta_m$ is the height of O-RU $m$, $\eta_v$ is the estimated flying car to O-RU $m$ projection distance on the ground, and $\eta_v$ is the estimated height of the flying car.

We consider the list of O-RUs is a priori known at the edge cloud, i.e., at Real-time Slice Controller (RT-SC). RT-SC can
V. PROBLEM FORMULATION FOR TWO-LEVEL CLOSED LOOPS

The previous section discussed the two closed loops in initial RBs distribution and scheduling. This section discusses RBs distribution and scheduling feedback.

Feedback for closed loop 1: After RBs scheduling for cars, we monitor RBs utilization. We consider \( \lambda^v_k \) as the arrival rate of the packets for each service \( k \) needed by car \( v \). RT-SC maps incoming packets with vO-DU that manages slice \( c \) of service \( k \). Each service has its queue, where queuing delay can be modeled with M/M/1 queuing system, where queuing delay \( q_{v,c}^{v,m} \) can be expressed as follows:

\[
q_{v,c}^{v,m} = \frac{\bar{\gamma}^{v,m} \tau_{v,c}^m}{\lambda^v_k - \mu^v_k},
\]

where \( \mu^v_k \) represents the service rate. \( \bar{\gamma}^{v,m} \) is a binary decision variable indicating whether or not the packet is assigned to slice \( c \) associated with service \( k \) at vO-DU \( d \), where \( \bar{\gamma}^{v,m} \) is given by:

\[
\bar{\gamma}^{v,m} = \begin{cases} 
1, & \text{if the packet is assigned to slice } c \text{ associated with service } k \text{ at vO-DU } d, \\
0, & \text{otherwise}.
\end{cases}
\]

Furthermore, we consider buffer \( \beta_{c,k}^{v,m} \) associated with service \( k \) that uses slice \( c \) at vO-DU \( d \). Then, we introduced a queuing status parameter \( \Psi^{d,c}_{v,k} \) associated with each service \( k \) and buffer threshold \( \beta_{c,k}^{d} \), where \( \Psi^{d,c}_{v,k} \) can dynamically be computed as follows:

\[
\Psi^{d,c}_{v,k} = \max \left\{ \left( \beta_{c,k}^{d} - \bar{\gamma}^{v,m} \right), 0 \right\},
\]

where \( \bar{\gamma}^{v,m} \) is the expected number of packets in the queue or queue occupancy for service \( k \).

Besides queuing delay and status, we consider transmission and propagation delays. We assume that each packet of the car \( v \) passes through fronthaul and wireless network. Let us consider \( \varepsilon_{v,c}^{v,k} \) as the size of the packet. The transmission delay for the wireless network between the car and O-RU becomes:

\[
\tau^{v,c} = \frac{\varepsilon_{v,c}^{v,k}}{R_m}. 
\]

Furthermore, the transmission delay \( \tau^{m,c} \) for fronthaul between O-RU \( m \) and vO-DU \( d \) can be expressed as follows:

\[
\tau^{m,c} = \frac{\bar{\gamma}^{v,m} \tau_{v,c}^m}{\bar{\gamma}^{v,m} \tau_{v,c}^m + \tau^{m,c} + \tau^{m,d}}.
\]

We consider \( \tau^{v,c} \) as feedback for the loop 1, where \( \tau^{v,c} \) should satisfy delay budget constraint \( \tau^{v,c} \leq \tau^{v,k} \).

To evaluate intra-slices RB allocation using closed loop 1, we defined network slice requirement satisfaction \( \varphi^{v,c}_{k} \) as follows:

\[
\varphi^{v,c}_{k} = \sum_{i=1}^{\mid V_k \mid} \bar{\gamma}^{v,m} \frac{\xi^{v,c}_{k}}{V_k},
\]

where \( V_k \) is a set of cars that use service \( k \) and \( \xi^{v,c}_{k} \) is the delay budget fulfillment parameter. \( \xi^{v,c}_{k} \) is given by:

\[
\xi^{v,c}_{k} = \begin{cases} 
1, & \text{if } \tau^{v,c}_{k} \leq \tau^{v,k}, \\
0, & \text{otherwise}.
\end{cases}
\]

To update initial RBs allocation for cars, we define intra-slice orchestration parameter \( \Omega^{d,c}_{v,k} \) for close loop 1, where \( \Omega^{d,c}_{v,k} \) is given by:

\[
\Omega^{d,c}_{v,k} = \begin{cases} 
\frac{\beta_{c,k}^{d}}{\beta_{c,k}^{d}}, & \text{if } \Psi^{d,c}_{v,k} = \beta_{c,k}^{d}, \\
0, & \text{if } \Psi^{d,c}_{v,k} > \beta_{c,k}^{d}, \\
1, & \text{otherwise}.
\end{cases}
\]

For close loop 1, when \( \Psi^{d,c}_{v,k} = \beta_{c,k}^{d} \), we consider that there are many incoming packets for slice \( c \) associated with service \( k \). In this scenario, vO-DU \( d \) needs to perform slice resource scale-up with \( \Omega^{d,c}_{v,k} = \frac{\beta_{c,k}^{d}}{\beta_{c,k}^{d}} \). Also, if \( \Psi^{d,c}_{v,k} > \beta_{c,k}^{d} \), the vO-DU \( d \) needs to perform slice resource scale-down with \( \Omega^{d,c}_{v,k} = \frac{\beta_{c,k}^{d}}{\beta_{c,k}^{d}} \) rate because the RB are underutilized (\( E[\lambda^v_k] \) is small). When \( \Psi^{d,c}_{v,k} = \beta_{c,k}^{d} \), there is no demands for slice \( c \) associated with service \( k \), vO-DU \( d \) can terminate RB allocation to that slice using \( \Omega^{d,c}_{v,k} = 0 \) because \( E[\lambda^v_k] = 0 \). Otherwise, we consider the initial RB allocation is well performed and there is no need to update the initial RB allocation and we set \( \Omega^{d,c}_{v,k} = 1 \).

Feedback for loop 2: We define RB usage to evaluate the usage of RB \( b_d \) allocated to vO-DU \( d \), where RB usage \( \varphi^{d,c}_{v,k} \) is given by:

\[
\varphi^{d,c}_{v,k} = \sum_{i=1}^{\mid V_k \mid} \bar{\gamma}^{v,m} b_d.
\]

Based on RB usage and slice requirement satisfaction, we formulate the following optimization problem that maximizes resource utilization, while meeting resource constraints and QoS requirements in terms of latency:

\[
\max_{(y,z,w)} \sum_{d \in D} y^{c,d} \varphi^{d,c}_{v,k} + \sum_{v \in V_k} \sum_{c \in C} w^{v,m} \varphi^{v,c}_{k} 
\]

subject to

\[
\sum_{v \in V_k} \sum_{c \in C} y^{c,d} \leq 1, \forall m \in M, \quad (30a)
\]

\[
\sum_{v \in V_k} \sum_{c \in C} y^{c,d} \leq 1, \quad (30b)
\]
In the formulated optimization problem in (30), the constraint in (30a) ensures that the RBs allocated to cars \((t_{i,c}^{k,d} f_{i,d}^{k})\) can only be allocated to a single car. The constraint in (30b) guarantees that each slice \(c\) associated with service \(k\) is created at one vO-DU. We use the penalty \(\nu_c\) to ensure the RBs allocated to cars \((R_i^v)\) represent \((t_{i,d}^{k,c} f_{i,d}^{k,c})\) do not exceed the available vO-DU resources. The constraint in (30d) relates to intra-fronthaul network and it ensures that each node does not send more traffic than the fronthaul capacity. Here, \(\tau\) allows to convert network traffic (MB) to traffic per second (Mbps) so that it can be comparable with \(\varpi_m.d\).

The problem in (30) is a combinatorial optimization problem, which is NP-hard and does not have an efficient polynomial-time solution. Also, an optimization problem that can lead to a stationary solution is not appropriate for resource auto-scaling because the slice resource auto-scaling process is a continuing, not stationary task [39]. Demands for network slices should be learned continuously to adapt to the change in workload and network environment. Therefore, we change (30) to reward functions that can reflect different QoS fulfillment, workload, and network condition changes.

We formulate a reward function \(r_{t,c}(z, w)\) for closed loop 1 so that it can reflect intra-slice QoS fulfillment in terms of delay and workload changes at time \(t\):

\[
\begin{align*}
    r_{t,c}(z, w) = & \ w_{k,c}^{v_c,d} \nu_c^c + \Delta_m \left( \varpi_m.d - \sum_{v \in V_k} \nu_v \sum_{i \in V} \sum_{c \in C} \nu_{c}^{v_c,d} \Delta_v^m \right) \\
    & \ + \Delta_v \left( 1 - \sum_{v \in V} \nu_v \sum_{i \in V} \sum_{c \in C} \nu_{c}^{v_c,d} \Delta_v^m \right) + \Delta_v \nu_c^d, \quad (31)
\end{align*}
\]

where \(\nu_c^d = \left| b_c^{v_c,d} - \sum_{v \in V} \nu_{c}^{v_c,d} \right|\). We use \(\Delta_m\) to denote the penalty of violating fronthaul resource constraint. \(\Delta_v\) is the penalty parameter for violating RB allocation constraint. \(\Delta_v^m\) is the penalty parameter to ensure that intra-slice scaling does not violate the vO-DU RBs capacity constraint. In other words, the penalties \(\Delta_m, \Delta_v, \text{ and } \Delta_v^m \in [0, 1]\) balance trade-off of violating fronthaul resource constraint, violating RBs allocation constraint for the cars, and violating the vO-DU RBs capacity constraint.

We formulate a reward function \(r_{t,d}(y)\) for closed loop 2 to evaluate the RB \(b_d\) utilization at vO-DU \(d\) at time \(t\):

\[
\begin{align*}
    r_{t,d}(y) = & \ y_{b,d}^{c,d} \Phi_c^{d} + \Delta_d \left( 1 - \sum_{c \in C} y_{b,d}^{c,d} \right) \\
    & \ + \Delta_b \left( B - \sum_{c \in C} y_{b,d}^{c,d} + \nu_c^d \right) \quad (32)
\end{align*}
\]

where \(\Delta_d\) is the penalty parameter to ensure each slice is managed by one vO-DU. We use \(\Delta_b\) to denote the penalty that guarantees RB updates do not violate RB constraint. In other words, \(\Delta_b\) and \(\Delta_d \in [0, 1]\) balance trade-off between violating constraint of RB distribution to vO-DU for scheduling and RB capacity constraint.

Connecting two loops: Closed loop 1 maximizes reward function \(r_{t,c}(z, w)\) by satisfying intra-slice QoS in terms of delay and workload changes at time \(t\). On the other hand, closed loop 2 needs to maximize reward \(r_{t,d}(y)\) and avoid violation of RB capacity constraints at vO-DU \(d\). Therefore, RB usage at vO-DU \(d\) depends on intra-slice RB allocation. Therefore, we formulate a main reward function \(r_t(y, z, w)\) that interconnects the two proposed closed loops at time \(t\), where \(r_t(y, z, w)\) is given by:

\[
r_t(y, z, w) = r_{t,d}(y) + \phi_{dis} r_{t,c}(z, w) \quad (33)
\]

Since the closed loop two has to maximize reward in (33) that combines (31) and (32), where (31) is already maximized with closed loop one, we introduce \(\phi_{dis}\) as discount parameter for \(r_{t,c}(z, w)\) to allow the closed loop 2 to put more emphasis on (32). Maximizing reward functions helps to avoid penalties, i.e., system failure and performance degradation when constraints are violated. In other words, rewards get reduced every time the constraint is violated.

VI. PROPOSED SOLUTION

In (32), closed loop 2 at Near-RT RIC needs to deal with actions \(A(y)\) consist of assigning initial RBs, keep initial RBs allocation \((\nu_c^d = 0)\), RBs scale-up \((\nu_c^d > 0)\), RBs scale-down \((\nu_c^d < 0)\), and terminate RBs allocation for vO-DUs \((\nu_c^d = -b_c^{v_c,d})\), i.e., \(\sum_{v \in V} \nu_v \sum_{i \in V} \sum_{c \in C} \nu_{c}^{v_c,d} R_i^v = 0\). The states \(S = \{ (B, D, C) \}\) at Near-RT RIC consist of the states of RBs \(B\), vO-DUs \(D\), and slices \(C\) managed by vO-DUs. On the other hand, closed loop 1 needs to deal with actions \(A'(z, w)\) consist of assigning initial RBs, keep initial RBs allocation \((\nu_c^d = 1)\), RBs scale-up \((\psi_c^{d} = \beta_c^{d})\), RBs scale-down \((\psi_c^{d} > \beta_c^{d})\), and terminate RBs allocation \((\psi_c^{d} < \beta_c^{d})\) for cars. The states \(S' = \{ (V, \Omega, \Psi) \}\) at RT-SC consist of the states of \(V\) cars managed by slices, intra-slice orchestration \(\Omega\), and queue \(\Psi\). The closed loop 1 has direct access to the environment, observes cars’ demands, and assigns RBs to cars. Based on queue status and intra-slice satisfaction, the closed loop 1 can keep or update the RBs allocation for cars. Then, it gives feedback to closed loop 2 so that closed loop 2 can have an overview of \(A(y, z, w)\), maximize (32), and update RBs for vO-DUs \(D\). Since the initial RBs allocation to the services of tenants is based on ARB, in RB auto-scaling using \(\nu_c^d\) and \(\Omega_c^{d} \), we assume the InP and tenants can negotiate flat rate increase or decrease on \(J^{k,c}_{b}(R_i^{k})\).

RL or DRL [41] can be applied to handle the formulated rewards. However, finding one RL or DRL model that uses two closed loops is a challenging issue. To overcome this issue, we choose Ape-X [40] shown in Fig. 4 as distributed RL over other RL or DRL approaches. Ape-X decomposes DRL into two components: learner and actor. The actor and learner use the same deep neural network model. In other words, the deep
neural network model is distributed to actors, where actors implement and evaluate the deep neural network model by interacting with the environment. Then, actors store the observation data in a replay memory, where the learner samples batches of data and updates the model parameters. In recent implementation, Ape-X works with two popular reinforcement learning approaches: Deep Q-Networks (DQN) and Deep Deterministic Policy Gradient (DDPG) [40]. In this work, we use Ape-X with DQN, where DQN [42] integrates deep learning into Q-Learning. The simplest form of Q-Learning, which is called one-step Q-Learning, is given by:

\[
Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_t) - Q(s_t, a_t)],
\]

(34)

where \(\alpha\) is the learning rate and \(\alpha \in \mathcal{A}\) is an action that was taken in the state \(s_t\) by an agent. \(\gamma_t\) (\(0 < \gamma_t \leq 1\)) is discount factor. On the other hand, DQN uses standard feed-forward neural networks to calculate Q-Value. The DQN uses two networks, Q-Network to calculate Q-Value in the state \(s_t\) and target network to calculate Q-Value in the state \(s_{t+1}\) such that:

\[
Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left( r_{t+1} + \gamma_{t} \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right).
\]

(35)

The loss function \(\Phi_t(\theta)\) to be minimized in Ape-X [40] can be expressed as follows:

\[
\Phi_t(\theta) = \frac{1}{2} \left( \tilde{G}_t - Q(s_t, a_t, \theta) \right)^2,
\]

(36)

where \(\theta\) represents the parameters of the neural networks and \(\tilde{G}_t\) is the return function. As described in [40], Ape-X computes \(\tilde{G}_t\), where \(\tilde{G}_t\) is expressed as follows:

\[
\tilde{G}_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, \argmax_a Q(s_{t+n}, a, \theta), \theta^-).
\]

(37)

In (37), \(n\) is the number of steps. We use \(t\) to represent a time index of sampling experience in replay memory. The experience sampling starts with state \(s_t\), action \(a_t\), and parameters of the target network \(\theta^-\). We use \(T\) to denote the total number of time steps until the end of the training process.

Fig. 5 shows the application of Ape-X as a solution to our problem. In our approach, Near-RT RIC acts as learner and actor for closed loop 2 and vO-DUs act as actors for closed loop 1. In Algorithm 2, Near-RT RIC initializes \(\theta_0\) and \(b_{d,c,k}^0\). Then, Near-RT RIC sends \(h_{d,c}^0\) and \(\theta_0\) to vO-DUs via RT-SC and save them to replay memory. Also, Algorithm 2 keeps checking the replay memory to get updates from closed loop 1 and computes the loss function \(\Phi_t(\theta)\) and updates \(\theta_t\) to \(\theta_{t+1}\). Then, Near-RT RIC computes Temporal Difference (TD) error \((\gamma_t \max_a Q(s_{t+1}, a) - Q(s_t, a_t))\) using DQN and updates replay memory and sends \(\theta_{t+1}\) and updated RBs \(b_{d,c,k}^t\) to RT-SC for vO-DUs.

In Algorithm 3, vO-DU gets initial parameters from the learner and via RT-SC such as \(\theta_0\) and RBs \(b_{d,c,k}^t\) and slices assigned to vO-DU. Then, vO-DU performs intra-slices actions. We use \(T'\) to denote the total number of time steps for vO-DU. Each vO-DU stores states, \(v_{d,c}^t\), actions, rewards, and discount factors in local memory. In each period \(T\), states, orchestration parameters, actions, rewards, discount factors, and TD, are sent to replay memory via RT-SC.

**Algorithm 2** RBs Allocation to vO-DUs (Near-RT RIC as Learner and Actor)

1. **Input:** \(T\);
2. Initialize \(t = 0\);
3. \(\theta_0 \leftarrow \text{InitializeLearningParameter}()\);
4. \(b_{d,c,k}^0 \leftarrow \text{AssignRBtovODU()}\);
5. for all \(t = 1\) to \((t = T)\) do
6. \(a_{t-1} \leftarrow \text{KeepUpdateSliceResourcevODU()}\);
7. \(\gamma_{c,k}^t \leftarrow \text{CalculateODUUtilization()}\);
8. \(r_{t,d}(y) \leftarrow \text{CalculateReward()}\);
9. InLocalMemory.add(\(s_{t-1}, a_{t-1}, r_{t,d}, \gamma_t\));
10. \(i, \tau \leftarrow \text{GetSampleFromReplayMemory}()\);
11. \(\Phi_t(\theta) \leftarrow \text{CalculateLoss}(\tau; \theta_t)\);
12. \(\theta_{t+1} \leftarrow \text{UpdateLearningParameters}(\Phi_t(\theta), \theta_t)\);
13. \(b_{d,c,k}^t \leftarrow \text{UpdateRBAllocation()}\);
14. \(r_{t}(y, z, w) \leftarrow \text{CalculateReward()}\);
15. \(p \leftarrow \text{CalculateTD()}\);
16. InReplayMemory.SetTD(\(i, p, r_t\));
17. Periodically(UpdateReplayMemory())
18. end for

**Theorem 4:** Computational complexity of Algorithms 2 and 3 is \(O(n)\).

**Proof:** In the Algorithm 2, we have one loop at lines 5-18, which depends on the number of vO-DUs and slices. On the other hand, the Algorithm 3 contains one loop at lines \((6 \sim 19)\) and it depends on the number of vehicles. In extreme scenario, we may have \(n\) number of vehicles, slices, and vO-DUs. As result, Algorithms 2 and 3 have computational complexity \(O(n)\).

VII. PERFORMANCE EVALUATION

In this section, we present the performance evaluation of the proposed closed loops for RAN slice resources management serving flying and ground-based cars. We use Python [43] for numerical analysis and OpenAI Gym [44] for making DRL environment.
A. Simulation Setup

We start with a realistic scenario of 3 flying cars and ground-based cars uniformly distributed over the range from 10 to 35 cars. We use 6 O-RUs and one edge cloud to provide a network connection to cars. For the locations of O-RUs, travel distances, time, and routes of flying and ground-based cars, we use VeRoViz as a suite of tools designed for car routing from the Optimator Lab at the University at Buffalo [45]. In Table II, d_flying is the flying distance in meters and t_flying words, flying and ground-based cars fly/navigate in the area of 6 O-RUs. In other words, flying and ground-based cars fly/navigate in the area of 6 O-RUs, where the flying/travel starts from O-RU 1 and ends at O-RU 6.

We use 100 MHz channel bandwidth with 30 kHz subcarrier spacing and 0.5 millisecond TTI. The number of RBs is 273 managed by 3 vO-DUs, where each vO-DU initially has \( b_d = 91 \) RBs. In ARB, we use 10 tenants, where the demand \( n_b \) of each tenant is in the range of 6 to 40, and \( J_{b,l}^{k} \) is in the range from 10 to 20. We set \( p = 15 \) and consider that the number of slices associated with services varies based on the output of the auction. We consider seven services from 5QI [29], such as advanced driving and remote driving, whose delay budget \( \gamma_k \) ranges from 5 to 300 milliseconds. In other words, 273 RBs managed by 3 vO-DUs should be allocated to 7 services. Each car chooses one or more services randomly from the list of 7 services. The packet size \( a_{b,l}^{k} \) is generated randomly in the range from 1 kilobyte to 10 megabytes. However, considering passengers with varying services and associated delay budgets and devices’ capabilities in the cars, other tools will be required in addition to VeRoViz.

As described in [46], to implement Ape-X, we use Ray [47] and Keras with TensorFlow [48]. In Ape-X, for the deep neural network model, we use the input layer of 3 neurons, two hidden layers of 128 neurons per hidden layer, and an output layer of 4 neurons. The input of 3 neurons corresponds to states. We assume initial RBs allocation can be performed using auction and based on initial demands. The four neurons in the output layer consist of 4 actions: keep initial RBs allocation, RBs scale-up, RBs scale-down, and termination of RBs allocation. Time step is set to 100000, maximum sample size is set to 50000 records, \( \alpha = 0.0001 \), and \( \gamma_t = 0.99 \).
The penalties vary in the range between 0.01 and 0.002. Our implementation codes are available on GitHub [49].

B. Simulation Results

The simulation results in Fig. 6 show RBs allocation to the tenants who provide services to the cars. Based on available RBs and bidding values \( (j^b, k \geq b) \), 7 tenants win the auction using the VCG and get 72% of the total RBs. Furthermore, we solve the optimization problem in (6) using MOSEK [50] as a mixed-integer optimization solver and compare MOSEK solution with VCG solution. In MOSEK, only a small number of tenants of \( j^b, k \geq b \) win the auction and get 26% of the total RBs. We consider unallocated RBs as the residual resources for RBs allocation scale-up. However, using MOSEK, InP remains with more unallocated RBs. Therefore, VCG has better performance than MOSEK. The common behavior of VCG and MOSEK, they do not allow InP to allocate more than available RBs. Also, as shown in Fig. 7, with VCG and MOSEK, all winning tenants pay prices that are less or equal to their bidding values. In other words, our ARB satisfies individual rational and truthful bidding, where the winner pays a price that is less or equal to its bidding value, while the tenant who does not win ARB pays nothing.

After the auction, hereafter, we use the results from VCG. Fig. 8 shows RBs allocated to the services of the tenants who won the ARB, where each service corresponds to one slice. RBs of services are distributed to vO-DUs for scheduling purposes in the closed loop 2. Fig. 9 shows the RBs distributed to vO-DUs using the round-robin policy starting from vO-DU 1, where vO-DU 1 and vO-DU 2 manages 3 slices, while vO-DU 3 has one slice. Here, we remind that each vO-DU has \( b_d = 91 \) RBs as the maximum limit, and RBs allocated to the slices at vO-DU has to respect RBs constraint \( \sum_{k=1}^{K_d} b^{c,k}_d \leq b_d \). In other words, the observation space of Ape-X for each vO-DU is in the range from \( b_d = 0 \) to \( b_d = 91 \). RB allocation, scale-up, and scale-down should vary in this range. This figure shows the number of cars getting service(s) from each vO-DU. Fig. 10 shows the number of cars (minimum, first quartile, median, third quartile, and maximum) that use specific slices, where slice 3 is more utilized than other slices.

Fig. 11 presents the RBs usage ratio defined in (29) for vO-DUs. Since each vO-DU manages limited RBs, we consider \( \phi^d_{c,k} = 1 \) as the maximum RB usage ratio. In general, this figure shows that our approach satisfies vO-DUs resource constraints with a minor resource constraint violation at
vO-DU 2 (at $\phi_{d,c} > 1.0$, i.e., at more than 100% utilization, the incoming request for RBs needs to be rejected).

Furthermore, Fig. 12 shows network slice requirement satisfaction in terms of delay as described in (26), wherein in most cases, our approach reaches 100% slice requirement satisfaction except slices 0 and 1 managed by vO-DU 1. Furthermore, we consider a larger scenario, where flying cars are uniformly distributed in the range from 1 to 20 cars, and ground-based cars remain uniformly distributed over the range of 10 to 35 cars. We consider this setting of 55 cars as large-scale compared to the first scenario of 38 cars. Fig. 13 shows the number of flying cars per vO-DU. Furthermore, Fig. 14 presents slice requirement satisfaction for flying cars. Considering flying and ground-based cars and comparing Figures 12 and 15, Fig. 15 clearly shows that increasing the number of cars without increasing total RBs, will decrease slice requirement satisfaction. Therefore, increasing the number of cars should be compensated by adding more RBs.

Fig. 16 presents the reward per actor using Ape-X. In other words, the rewards of vODU 1 (actor 1), vODU 2 (actor 2), vODU 3 (actor 3), and Near-RT RIC (actor 4). Here, we remind that vODUs focus on maximizing $r_{t,c}(z,w)$ in closed loop 1, while Near-RT RIC focuses on maximizing $r_{t,d}(y)$ in closed loop 2. Rewards are not the same for actors because different vODUs manage different slices. Also, the slices do not have the same numbers of RBs and serve varying numbers of vehicles. To compare Ape-X-based solution with other DRL approaches, we use the reward function in (33), where the discount parameter is set to $\phi_{dis} = 0.0018$. Fig. 17 shows the mean of total reward using Ape-X and Actor-Critic DRL. Actor-Critic is popular in DRL-based network slicing literature such as [51]. The results in this figure show that Ape-X has better performance than Actor-Critic DRL. Figs. 16 and 17 show clearly that our Ape-X based solution converges from 10
episodes. Furthermore, the proposed approach has linear computation complexity $O(n)$, where its execution time depends on the number of vehicles. As shown in [52], such linear computation complexity is $O(n)$, where its execution time depends on the number of vehicles.

Fig. 17. Mean total reward maximization.

VIII. CONCLUSION

This paper presents two-level closed loops for managing RAN slice resources serving flying and ground-based cars. We have used an auction mechanism for allocating RBs to the tenants who provide services to cars using slices. Then, we proposed two closed loops that complement each other, where closed loop 2 distributes RBs to vO-DUs and closed loop 1 at vO-DUs performs intra-slices RB scheduling for cars. Closed loop 1 sends resources utilization updates to closed loop 2 so that the closed loop 2 can update RB distribution to vO-DUs. Using Apé-X as distributed reinforcement learning, the simulation results demonstrate that our approach satisfies more than 90% vO-DUs resource constraints and network slice requirements. One of our future works is extending our framework for more performance evaluation in different simulation environments.

ACKNOWLEDGMENT

The authors thank Mitacs, Ciena, and ENCQOR for funding this research under the IT13947 grant.

REFERENCES

[1] S. Ansari, A. Taha, K. Dashpipour, Y. Sambo, Q. H. Abbasi, and M. A. Imran, “Urban air mobility—A 6G use case?,” Front. Commun. Netw., vol. 2, Aug. 2021, Art. no. 729767.

[2] S. S. Ahmed et al., “The flying car—Challenges and strategies toward future adoption,” Front. Built Environ., vol. 6, p. 106, Jul. 2020.

[3] G. Pan and M.-S. Alouini, “Flying car transportation system: Advances, techniques, and challenges,” IEEE Access, vol. 9, pp. 24586–24603, 2021.

[4] A. A. Zaid, B. E. Y. Belmekki, and M.-S. Alouini, “Technological trends and key communication enablers for eVTOLs,” 2021, arXiv:2110.08830.

[5] N. Saed, T. Y. Al-Naffouri, and M.-S. Alouini, “Wireless communication for flying cars,” Front. Commun. Netw., vol. 2, p. 16, Jun. 2021.

[6] S. Al-Rubaye and A. Tsourdos, “Connectivity for air mobility transportation in future networks,” IEEE Future Netw. Tech. Focus, no. 12, pp. 1–6, Apr. 2021. [Online]. Available: https://futurenetworks.ieee.org/tech-focus/april-2021/connectivity-for-air-mobility-transportation-in-future-networks

[7] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghomimid, “A comprehensive survey on the E2E 5G network slicing model,” IEEE Trans. Netw. Service Manag., vol. 18, no. 1, pp. 49–62, Mar. 2021.

[8] B. Brik, K. Boutiba, and A. Ksentini, “Deep learning for 5G open radio access network: Evolution, survey, case studies, and challenges,” IEEE Open J. Commun. Soc., vol. 3, pp. 228–250, 2022.

[9] “O-RAN: Towards an open and smart RAN,” O-RAN Alliance, Alfter, Germany, White Paper, Oct. 2018.

[10] A. Ndikumana, K. K. Nguyen, and M. Cheriet, “Age of processing-based data offloading for autonomous vehicles in MultiRATs open RAN,” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 11, pp. 21450–21464, Nov. 2022.

[11] C.-Y. Yang, N. Nikaein, and T. Spyropoulos, “Radio access network resource slicing for flexible service execution,” in Proc. IEEE Comput. Commun. Workshops (INFOCOM WKSHPS), 2018, pp. 668–673.

[12] F. Fossati, S. Moretti, P. Perny, and S. Secci, “Multi-resource allocation for network slicing,” IEE/ACM Trans. Netw., vol. 28, no. 3, pp. 1311–1324, Jun. 2020.

[13] Z. Shu and T. Taleb, “A novel QoS framework for network slicing in 5G and beyond networks based on SDN and NFV,” IEEE Netw., vol. 34, no. 3, pp. 256–263, May/Jun. 2020.

[14] D. H. Kim, S. M. A. Kazmi, A. Ndikumana, A. Manzoor, W. Saad, and C. S. Hong, “Distributed radio slice allocation in wireless network virtualization: Matching theory meets auctions,” IEEE Access, vol. 8, pp. 73494–73507, 2020.

[15] X. Cheng, Y. Wu, G. Min, A. Y. Zomaya, and X. Fang, “Safeguard network slicing in 5G: A learning augmented optimization approach,” IEEE J. Sel. Areas Commun., vol. 38, no. 7, pp. 1600–1613, Jul. 2020.

[16] Q. Liu, T. Han, and E. Moges, “EdgeSlice: Slicing wireless edge computing network with decentralized deep reinforcement learning,” 2020, arXiv:2003.12911.

[17] N. Salhab, R. Rahim, R. Langar, and R. Boutaba, “Machine learning based resource orchestration for 5G network slices,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), 2019, pp. 1–6.

[18] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, “Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning,” IEEE Access, vol. 8, pp. 68183–68198, 2020.

[19] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, “Energy-efficient deep reinforcement learning assisted resource allocation for 5G-RAN slicing,” IEEE Trans. Veh. Technol., vol. 71, no. 1, pp. 856–871, Jan. 2022.

[20] P. Naik et al., “Closed-loop automation for 5G slice assurance,” in Proc. 14th Int. Conf. COMmun. Syst. NETw. (COMSNETS), 2022, pp. 424–426.

[21] M. Xie et al., “Towards closed loop 5G service assurance architecture for network slices as a service,” in Proc. Eur. Conf. Netw. Commun. (EuCNC), 2019, pp. 139–143.

[22] M. A. Habibi, B. Han, F. Z. Youssaf, and H. D. Schotten, “How should network slice instances be provided to multiple use cases of a single vertical industry?” IEEE Commun. Stand. Mag., vol. 4, no. 3, pp. 53–61, Sep. 2020.

[23] Z. Yuan and G.-M. Muntean, “AirSlice: A network slicing framework for UAV communications,” IEEE Commun. Mag., vol. 58, no. 11, pp. 62–68, Nov. 2020.

[24] A. E. Garcia et al., “Performance evaluation of network slicing for aerial vehicle communications,” in Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops), 2019, pp. 1–6.

[25] H. D. R. Albonda and J. Pérez-Romero, “An efficient RAN slicing strategy for a heterogeneous network with eMBB and V2X services,” IEEE Access, vol. 7, pp. 44771–44782, 2019.

[26] W. Wu et al., “Dynamic RAN slicing for service-oriented vehicular networks via constrained learning,” IEEE J. Sel. Areas Commun., vol. 39, no. 7, pp. 2076–2089, Jul. 2021.

[27] D. Tamang, S. Martiradonna, A. Abrardo, G. Mandó, G. Roncella, and G. Boggia, “Architecting 5G RAN slicing for location aware vehicle to infrastructure communications: The autonomous tram use case,” Comput. Netw., vol. 200, Dec. 2021, Art. no. 108501.

[28] Y. Sahraoui et al., “A cooperative crowdsensing system based on flying cars,” in Proc. IEEE 14th Int. Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), 2021, pp. 1–6.

[29] “3GPP TS 23.501 (2019), system architecture for the 5G system; stage 2,” 3GPP, Sophia Antipolis, France, 3GPP Rep. TS 23.501, 2019.

[30] “3GPP TS 23.501 (2019), system architecture for the 5G system; stage 2,” 3GPP, Sophia Antipolis, France, 3GPP Rep. TS 23.501, 2019.

[31] M. Caminati, M. Kerber, C. Lange, and C. Rowat, “Vickrey-clark-groves (VCG) auctions,” Dept. Econ., Univ. Birmingham, U.K., 2016.

[32] L. Blunrosen and N. Nisan, “Combinatorial auctions,” Algorithmic Game Theory, vol. 267, Cambridge, U.K.: Cambridge Univ. Press, 2007, p. 300.

[33] M. Chiang, Networked Life: 20 Questions and Answers, Cambridge, U.K.: Cambridge Univ. Press, 2012.
[33] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, “Collaborative cache allocation and computation offloading in mobile edge computing,” in Proc. 19th Asia-Pacific Netw. Operat. Manage. Symp. (APNOMS), 2017, pp. 366–369.

[34] E. Elamarian and B. Sudhakar, “Greedy based round robin scheduling solution for data traffic management in 5G,” in Proc. Int. Conf. Smart Syst. Inventive Technol. (ICSIIT), 2019, pp. 773–779.

[35] P. K. Korrai, E. Lagunas, A. Bandi, S. K. Sharma, and S. Chatzinotas, “Joint power and resource block allocation for mixed-numerology-based 5G Downlink under imperfect CSI,” IEEE Open J. Commun. Soc., vol. 1, pp. 1583–1601, 2020.

[36] J. F. Sekaran, H. Kalvran, and L. Irudhayaraj, “Modeling and analysis of GPS–GLONASS navigation for car like mobile robot,” J. Elect. Eng. Technol., vol. 15, no. 2, pp. 927–935, 2020.

[37] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, “Deep learning based caching for self-driving cars in multi-access edge computing,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 5, pp. 2862–2877, May 2021.

[38] A. Ndikumana and C. S. Hong, “Self-driving car meets multi-access edge computing for deep learning-based caching,” in Proc. Int. Conf. Int. Netw. (ICOIN), 2019, pp. 49–54.

[39] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A review of auto-scaling techniques for elastic applications in cloud environments,” J. Grid Comput., vol. 12, no. 4, pp. 559–592, 2014.

[40] D. Horgan et al., “Distributed prioritized experience replay,” 2018, arXiv:1803.00933.

[41] B. R. Kiran et al., “Deep reinforcement learning for autonomous driving: A survey,” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[42] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A theoretical analysis of deep Q-learning,” in Proc. Learn. Dyn. Control, 2020, pp. 486–489.

[43] A. Nagpal and G. Gabrani, “Python for data analytics, scientific and technical applications,” in Proc. Amity Int. Conf. Artif. Intell. (AICAI), 2019, pp. 140–145.

[44] G. Brockman et al., “OpenAI gym,” 2016, arXiv:1606.01540.

[45] L. Peng and C. Murray, “VeRoViz: A vehicle routing visualization toolkit.” 2020. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3746037

[46] E. Bilgin, Mastering Reinforcement Learning With Python: Build Next-Generation, Self-Learning Models Using Reinforcement Learning Techniques and Best Practices, 1st ed. Birmingham, U.K.: Packt Publ., Dec. 18, 2020.

[47] E. Liang et al., ”Ray RLlib: A composable and scalable reinforcement learning library,” 2017, arXiv:1712.09383.

[48] K. Ramasubramanian and A. Singh, “Deep learning using keras and tensorflow,” in Machine Learning Using R. Berkeley, CA, USA: Springer, 2019, pp. 667–688.

[49] A. Ndikumana. “Two-level closed loops for RAN slice resources management.” Accessed: Jan. 30, 2023. [Online]. Available: https://docs.mosek.com/9.3/intro.pdf

[50] M. ApS. “MOSEK optimization suite.” 2017. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3746037

[51] F. Rezazadeh, H. Chergui, L. Christofi, and C. Verikoukis, “Actor-Critic based reinforcement learning with zero-touch joint resource and energy control in network slicing,” in Proc. IEEE Int. Conf. Commun. (ICC), 2021, pp. 1–6.

[52] M. Pugin, T. Zugno, M. Polese, and M. Zorzi. “Resource management for 5G NR integrated access and backhaul: A semi-centralized approach,” IEEE Trans. Wireless Commun., vol. 21, no. 2, pp. 753–767, Feb. 2022.

Kim Khoa Nguyen (Senior Member, IEEE) is an Associate Professor with the Department of Electrical Engineering and the Founder of the Laboratory on IoT and Cloud Computing, University of Quebec’s Ecole de technologie superieure. He was the Architect of the Canarie’s GreenStar Network and is the Canadian Architecture of large-scale projects with Ericsson, Ciena, Telus, InterDigital, and Ultra Electronics. In the past, he served as a CTO with Inocybe Technologies (currently, Kontron, Canada), a world’s leading company in software-defined networking solutions. He is the author of more than 100 publications, and holds several industrial patents. His expertise includes network optimization, cloud computing IoT, 5G, big data, machine learning, smart city, and high speed networks. He is the recipient of Microsoft Azure Global IoT Contest Award 2017 and Ciena’s Aspirational Prize 2018.

Mohamed Cheriet (Senior Member, IEEE) received the bachelor’s, M.Sc., and Ph.D. degrees in computer science from USTHB, Algiers, and the University of Pierre and Marie Curie, Paris VI, in 1984, 1985, and 1988, respectively. He was then a Postdoctoral Fellow with CNRS, Pont et Chaussées, Paris V, in 1988, and CENPARMI, Concordia University, Montreal, in 1990. Since 1992, he has been a Professor with the Systems Engineering Department, University of Quebec—Ecole de technologie superieure (ETS), Montreal, where he was appointed as a Full Professor in 1998. He was the Director with the LIVIA Laboratory for Imagery, Vision, and Artificial Intelligence from 2000 to 2006, and has been the Founder and the Director with the Synchronedia Laboratory for Multimedia Communication in Telepresence Applications since 1998. He has published more than 500 technical papers in the field and serves on the editorial boards of several renowned journals and international conferences. He contributed six patents (three granted), and the first standard ever, IEEE 1922.2, on real-time calculation of ICT emissions in April 2020, with his IEEE Emissions Working Group. His research has extensive experience in Sustainable and Intelligent Next Generation Systems. He is an Expert in computational intelligence, pattern recognition, machine learning, artificial intelligence and perception and their applications, more extensively in networking and image processing. In addition, He held a Tier 1 Canada Research Chair on Sustainable and Smart Eco-Cloud from 2013 to 2000, and lead the establishment of the first smart university campus in Canada, created as a hub for innovation and productivity at Montreal. He is the General Director of the FRQNT Strategic Cluster on the Operationalization of Sustainability Development, CIRODD from 2019 to 2026. He is the Administrative Director of the $12M CFI‘2022 CEOS® Net Manufacturing Cloud Network. He is the recipient of the 2016 IEEE J.M. Ham Outstanding Engineering Educator Award, the 2013 ETS Research Excellence Prize, for his outstanding contribution in green ICT, cloud computing, and big data analytics research areas, and the 2012 Queen Elizabeth II Diamond Jubilee Medal. He is the Founder and the Former Chair of the IEEE Montreal Chapter of Computational Intelligent Systems, a Steering Committee Member of the IEEE Sustainable ICT Initiative, and the Chair of ICT Emissions Working Group. He is a 2016 Fellow of the International Association of Pattern Recognition, a 2017 Fellow of the Canadian Academy of Engineering, a 2018 Fellow of the Engineering Institute of Canada, and a 2019 Fellow of Engineers Canada.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.