IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 12, 2022, accepted May 5, 2022, date of publication May 12, 2022, date of current version May 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174548

Next-Generation Edge Computing Assisted Autonomous Driving Based Artificial Intelligence Algorithms

HATEM IBN-KHEDHER[1], MOHAMMED LAROUI[2,3,4], HASSINE MOUNGLA[3,4], (Member, IEEE), HOSSAM AFIFI[4], AND EMAD ABD-ELRAHMAN[5]

[1]Capgemini Engineering, Meudon, 75017 Paris, France
[2]Computer Science Department, Djillali Liabes University, Sidi Bel Abbes 22000, Algeria
[3]Laboratory of Informatics Paris Descartes (LIPADE), Universite de Paris Cite, 75006 Paris, France
[4]Telecom SudParis, Institut Polytechnique de Paris, 91764 Palaiseau, France
[5]National Telecommunication Institute (NTI), Cairo 11768, Egypt

Corresponding author: Emad Abd-Elrahman (emad.abdelrahman@nti.sci.eg)

ABSTRACT Edge Computing and Network Function Virtualization (NFV) concepts can improve network processing and multi-resources allocation when intelligent optimization algorithms are deployed. Multiservice offloading and allocation approaches pose interesting challenges in the current and next-generation vehicle networks. The state-of-the-art optimization approaches still formulate exact algorithms, and tune approximation methods to get sufficient solutions. These approaches are data-centric that aim to use heterogeneous data inputs to find the near optimal solutions. In the context of connected and autonomous vehicles (CAVs), these techniques show an exponential computational time and deal only with small and medium scale networks. Therefore, we are motivated by using recent Deep Reinforcement Learning (DRL) techniques to learn the behavior of exact optimization algorithms while enhancing the Quality of Service (QoS) of network operators and satisfying the requirements of the next-generation Autonomous Vehicles (AVs). DRL algorithms can improve AVs service offloading and optimize edge resources. An Optimal Virtual Edge Autopilot Placement (OVEAP) algorithm is proposed using Integer Linear Programming (ILP). Moreover, an autopilot placement protocol is presented to support the algorithm. Optimal allocation and Virtual Network Function (VNF) placement and chaining of the autopilot, based on several new constraints such as computing and networking loads, network edge infrastructure, and placement cost, are designed. Further, a DRL approach is formulated to deal with dense Internet of Autonomous Vehicle (IoAV) networks. Extensive simulations and evaluations are carried out. Results show that the proposed allocation strategies outperform the state-of-the-art solutions and give better performance in terms of Total Edge Servers Utilization, Total Edge Servers Allocation Time, and Successfully Allocated autopilots.

INDEX TERMS Edge computing, artificial intelligence, autonomous driving, optimization algorithms.

I. INTRODUCTION

Autonomous Vehicles (AV) [1]–[3] target the deployment of service chains including sensors sensing, computer vision, localization, High Definition (HD) Map building, path planning, and control. AV sensors include ultrasonic, cameras, radar, LiDAR, and Dedicated Short-Range Communications (DSRC) devices. Moreover, AV sensors generate heterogeneous data and need intelligent data fusion methods and machine learning techniques such as object classification, inference, and artificial intelligence models that ease the autonomous driving process. In recent AV sensors deployments, vehicles use a wide variety of sensing competences. Moderately, a vehicle has seventy sensors including ambient light sensors, accelerometers, gyroscopes, and moisture sensors. The combination of the Internet of Autonomous Vehicle (IoAV) sensors with different situational awareness, failures, and real-time response determines the AV complexities that they need to have a comprehensive software. Vehicles already had a number of functionalities for combining real-time sensing with perception and decision-making.

The associate editor coordinating the review of this manuscript and approving it for publication was Venkateshkumar M.

A self-driving process needs to solve complex AV issues through sensing, real-time object detection, classification, and segmentation [4]. The most important key issues in autonomous driving include safety, low latency [5], [6], high data rate, software accuracy, map completeness and correctness, augmented sensor fusion, etc.

Generally speaking, AVs gather a huge amount of raw data to figure out the world and incorporate data from other sensors, lasers, and radar to get a richer understanding of the environment. Then, it performs localization to figure out precisely its location in the world. Path planning and control modules are then enabled to chart a course through the world to reach the destination and makes appropriate decisions while driving such as steering wheel, hitting the throttle, hitting the brake, etc.

Current Internet of Things (IoT) platforms for AVs do not enable low-latency and real-time data processing, and require offloading data processing to near edge computing servers. In general, AVs connect to the edge platforms using 5G cellular networks to access real-time data analytics for required applications [7]. Edge servers can collaborate with other edge nodes in the vicinity, thereby creating a local distributed peer-to-peer (P2P) network beneath the cloud. Recently, the edge/cloud computing servers host AV service chains and provide feedbacks and decisions to the physical infrastructure.

In this paper, we contribute by virtualizing the main autopilot functions using Network Function Virtualization/Software Defined Networking (NFV/SDN) techniques. This step generates virtualized instances that can be deployed as a service in the network operator edges.

Moreover, despite the importance of the optimization task in virtualized architectures, such functions are missing in the global network architecture. Therefore, we try, in this paper, to contribute with an exact Optimal Virtual Edge Autopilot Placement (OVEAP) algorithm to decide about the optimal point of operations where the autopilot function should be hosted. Then, we proposed a Deep Reinforcement Learning (DRL) based Virtual Edge Autopilot Placement (DVEAP) algorithm to deal with large scale networks. These tools help network operators to manage their resources and collaborate efficiently with the Original Equipment Manufacturer (OEM).

A. PROBLEM STATEMENT

The main problem considered in this paper is the offloading approaches for AVs. It considers offloading autopilot functions, such as computer vision, perception, localization, planning, etc. to the near edge computing served by a 5G network operator. Moreover, the problem supports partial and/or full offloading, where the AV can offload one or multiple autopilot functions. The statement is described as follows: maximize the succeeded placement of the autopilot VNFs while minimizing the number of edge servers, where to place the offloaded autopilot functions at the distributed network edge, and how to guarantee an optimal quality of driving in terms of edge servers utilization and allocation, successfully allocated autopilots, average network delay, service time, and processing time.

We consider different types of constraints in our virtualization and offloading process. We introduce CPU, GPU, storage, RAM, Bandwidth, VNF chaining (without decomposition), minimum edge servers, and 5G link constraints. The proposed constraints are divided into different types: system type constraints, network type constraints, 5G link capacity, and Quality of Service/Quality of Experience (QoS/QoE) constraints. Moreover, edge type constraints include maximum number of simultaneous connections per server.

Recall that none of the previously described contributions considered the virtualization of a vehicle autopilot, neither the QoS/QoE. OVEAP optimization will maximize the total number of offloaded autopilots for AVs connected over 5G New Radio (NR) protocol. The objective is to minimize the cost of computing resources (i.e. active Mobile Edge Computing (MEC) servers). The main goal of our study is to propose optimization algorithms for edge-assisted autonomous vehicles and the underlying Artificial Intelligence (AI) mechanisms to optimize edge computing resources and vehicle networks.

B. MAIN CONTRIBUTIONS

The main contributions in this paper are as follows:

1) Design and propose an end-to-end, reliable and low-latency communication architecture that allows the allocation of compute-intensive autonomous driving services, in particular autopilot, to shared resources on edge servers and improve the level of performance for autonomous vehicles.
2) Propose an Advanced Autonomous Driving (A2D) communication protocol that supports dense moving vehicles.
3) Use of edge assisted Integer Linear Programming (ILP) techniques to allocate autopilot resources on the optimal edge computing servers.
4) Introduce an edge based Deep Reinforcement Learning (DRL) as a new approach to automate and optimize the allocation of heterogeneous autopilot computing and networking resources, extract knowledge from disseminated data, and recommend autopilot allocation strategies according to different metrics. The autopilot's Virtual Network Functions (VNFs) allocation is compared to standard optimization techniques in order to show how deep learning techniques will be used to solve generalized compute-intensive autonomous driving services optimization problems.

The rest of the paper is as follows. Section II highlights the related work background to ease the understanding of the paper contribution and give a detailed related work in the field of edge-assisted autonomous driving architectures, communication protocols, and optimization algorithms. Section III describes our proposed Advanced Autonomous Driving

(A2D) communication protocol. Section VI introduces the proposed mathematical programming approach for A2D. Section V enhances the optimization module with an Edge based Deep Reinforcement Learning (EDRL) approach. Section VI evaluates the proposed approaches and the work is concluded with some research directions in the final section (i.e. Section VII).

II. RELATED WORK BACKGROUND

Different techniques, standards, and norms adopted by AI, 5G, V2X, IoV, and Edge/Cloud computing in the area of edge assisted autonomous driving. Recent applications include the use of Artificial Intelligence (AI) techniques and Deep Learning (DL) models in self-driving cars, autonomous driving services management, edge computing assistance, and online optimization (e.g., Tesla AI [8], DeepRM, DeepMind, Deep Traffic). The popular Reinforcement Learning (RL) methods are Trust Region Policy Optimization (TRPO), Policy Gradient (PG), and Q-Learning (QL). More importantly, the rapid development of Internet of Vehicle (IoV) is also making autonomous driving a reality [9]. This section categorizes the AVs domain work into two sides as follows:

A. AUTONOMOUS VEHICLES AND EDGE COMPUTING CONVERGENCE

In [10], the authors present the state-of-the-art approaches that leverage the edge-computing paradigm in the autonomous driving field. However, it missed the discussion about current edge AI work and optimal resources allocation design.

Datta *et al.* [11] formulate some research and engineering challenges for developing cloud-based environment for connected car services. The test-bed services are running in a virtualized environment and are deployed using Micro-services. It leverages edge servers for vehicular data annotation and local processing with actuation. The work missed the use case of autonomous vehicles. Further, the authors do not take into account recent communication technologies that might enhance the overall performance metrics such as end-to-end delay, latency, and safety.

Liu *et al.* [12] set up an end-to-end prototype, which supports Wi-Fi, LTE, and DSRC communication technologies. The authors evaluate the performance in terms of network latency, power dissipation, and system utilization. They propose different communication and networking schemes that connect On Board Units (OBU) nodes to network gateways. Moreover, they implement communication prototypes using ROS messages. However, the prototype does not include the communication between OBUs, gateways (RSUs), and edge servers.

In [13], the authors design and propose a low power edge computing system for real-time autonomous robots and vehicles services. They propose an offloading strategy that decides when and where to offload autonomous driving tasks. Their work is only focusing on minimizing the power consumption of the edge platform. However, we are considering relevant metrics such as servers utilization, multi resources' utilization, QoS, and safety.

In [14], the authors addresses the issue of how to process large data volumes and still meet the objectives of the connected and autonomous vehicle driving. Therefore, they propose the introduction of edge and fog computing nodes as an assistance layer of processing. Further, they rise the problem of how to process large data volumes as quickly as possible. For this purpose, they propose moving machine learning models and functions to where data is generated and not collected such camera devices. Still, the authors do not take into consideration virtualized architectures at the network edge that can add flexibility, programmability, and control of the compute-intensive embedded-autopilot modules.

In [15], the authors propose an end-to-end machine learning algorithm for the entire autonomous driving procedures. The approach uses deep neural networks models to map directly collected IoAV sensory inputs, such as front-facing camera images, to driving actions such as steering angle. The work is of practical interest, however, it missed virtualized edge computing facility that helps in machine learning training and IoAV resources allocation.

In [16], the authors propose an autonomous vehicle world model that represents the vehicle's view of its road environment. The model takes as an input the heterogeneous information gathered from in-vehicle sensors, V2X communication, and a priori information (e.g., roads and intersections information) and outputs real time events that trigger and feed the decision-making module. They have used Cyclab [17], an open-source 3D simulator, to simulate the proposed framework. The work does not take benefit neither from edge computing nor from AI techniques. Moreover, the authors used a static world model that does not take into consideration navigation safety, communication latency, and performance indicators to evaluate the world model.

In [18] the authors propose a three-tier architecture for Vehicular Edge Computing (VEC) domain. It supports a high level of scalability, real-time data delivery, and mobility. The authors leverage SDN and NFV virtualization techniques to add more flexibility, control and a global view of the moving vehicles. This latter uses V2X communication technologies such as DSRC, LTE, and 5G to reach either the edge servers or the centralized cloud. They rise serious VEC technical issues and challenges such as latency, scheduling, load balancing, offloading, resource management, and security/privacy. The authors survey the main challenges and opportunities when vehicle network meets edge computing. However, the work missed an overall architecture that defines the most appropriate network modules and edge techniques. Moreover, the authors should take into consideration the V2X communication protocols in presence of the NFV/SDN paradigms. They also missed optimization techniques related to the network/edge convergence according to safety indicators when designing the allocation loop of autonomous vehicles.

In [19], the authors present an edge-cloud computing model for autonomous vehicles using the open-source software platform Autoware [20]. They believe that their proposed edge-cloud computing model for Autoware-based autonomous vehicles reduces the execution time and the total deadline miss. Among the main missing modules in their platform, the work consider neither the in-vehicle computing resources management, nor the Vehicle to Edge (V2E) communications.

In [21], the authors propose surrogate: an edge architecture for self-driving cars with OpenStack and ETSI open-source MANO. It aims at virtualizing the in-vehicle OBUs at the distributed edge platform and managing Multi-Access Edge Computing (MEC) layers that process real-time vehicle requests. The work suffers from optimal virtual OBU (vOBU) management and orchestration algorithms at the virtualized edge surrogates. Moreover, vOBU manager module needs to take into account solver instances related to the IoAV network scale and driving conditions.

In [22], the authors describe how to build a self-driving car, applying AI and ML techniques to train and test until the car drives safely. They are collaborating with Waymo Company, which offers cars having 4 million miles of driving and 2.5 billion simulated miles. Then, AI/ML models are feeded with the gathered data for training and knowledge extraction. The work missed the edge computing assistance for efficient computing and scalable processing. We believe that the work is interesting and autopilots can take benefit from these results.

In [23], the authors studied the problem of V2X service placement. They proposed an ILP technique to decide where to offload services, taking into consideration the limited computing resources, available at the network edges. They introduce decision variables to indicate the optimal service placement. Then, they proposed a greedy approach to deal with large network scales. The main objective of the proposed approaches is to minimize the overall delay experienced by vehicles. The authors do not take into consideration the AV requirement in terms of reliability, latency, and data rate. Moreover, ILP formulations need heuristics to deal with large autonomous vehicle network scales. In our contribution, we virtualize the entire autopilot function while ensuring V2X services constraints.

In [24], the authors implement a unified autonomous driving cloud infrastructure that supports heterogeneous applications. It can efficiently gather huge amount of raw data and perform distributed simulation in order to stress offloading algorithms. It can also perform offline DL model training and augmented HD Map generation. The authors rise the issue of heterogeneous applications deployment (i.e., simulations, offline DL model training, HD map generation) that need different infrastructure and orthogonal requirements. Indeed, deployed applications share all gathered data as inputs while assuring the required storage cost. Therefore, they propose a unified cloud infrastructure that supports heterogeneous AV applications. They introduce some design considerations in implementing the unified infrastructures including Spark RDD, Alluxio, and heterogeneous computing substrates. Despite the relevant work, the authors do not take into consideration the AV services placement, orchestration, and management which represent key modules in edge intelligence over virtualized AI-IoT architectures.

In [25] the authors propose a cloud based self-driving car which can optimize the in-vehicle data storage issues. They propose to free autonomous vehicles from all data and download everything from the cloud as per the need of the travel. Their solution allows to free vehicles from raw data and rely on a centralized cloud infrastructure for the drive. The authors assume a persistent network connectivity to the cloud and a sufficient in-vehicle storage to back the data in the case of limited network availability. The proposed cloud infrastructure is not clear and need to integrate scheduling algorithms that allocate gathered data to CPU cores and servers. Moreover, it missed distributed edge computing servers that efficient process sensitive application data.

In [26] the authors proposed Carcel, a cloud-assisted system for autonomous driving. The cloud platform has access to data from AV sensors and the roadside infrastructure environment. It assists autonomous vehicles to detect/avoid obstacles such as pedestrians and other vehicles that may not be directly detected by the AV sensors. It helps autonomous vehicles to plan efficient paths. Then, the authors introduce a cloud-based planner module along with request, sender, and receiver modules. They implement the planner module within the cloud using the Robot Operating System (ROS). The cloud-assisted system tracks request messages from the cloud, and accordingly transmits the sensor information in the form of UDP packets to the cloud. Their proposed metrics are the response time and the distance to pedestrian. We believe that the work is of practical interest, however, it missed virtualization techniques and VNF manager module that ease the allocation procedure of the autopilot chain on the cloud. Moreover, the edge/fog facility is missed from the overall architecture.

In [27] the authors explore a distributed computing architecture that addresses on-vehicle and off-vehicle computation as will be needed to support connected and autonomous driving. They suggest local/edge computation rather than offloaded to cloud servers in order to reduce the end-to-end latency. We believe that AI approaches may ease edge resources' management to satisfy Connected Autonomous Vehicles (CAV) requirements in terms of safety, latency, and bandwidth.

Different efforts have been proposed for Vehicular Edge Computing, where the edge servers provide different services according to the application. For instance, Ye *et al.* [28] proposed a service offloading framework in a fog computing environment, where the vehicles are used as mobile fog servers to provide services to connected end-users and also execute offloaded tasks from roadside cloudlets. The Genetic Algorithm (GA) is used as an allocation strategy to achieve tasks offloading with the least cost. The proposed strategy

TABLE 1. State-of-the-art approaches in edge computing assisted autonomous driving field.

Work	*Approach*	*Metrics*	*Limitations*
[4]	A survey that presents AI techniques and edge computing paradigm in IoV	Safety	Virtualization issues Edge-cloud cooperation.
[41]	A survey that studies MEE and MEC approaches in Autonomous Vehicle	Safety	autopilot, intersection movement awareness, and autonomous valet parking, MEC computing architecture missed emerging virtualization and optimization techniques.
[10]	This survey presents the state of the art approaches that leverages the edge-computing paradigm in the autonomous driving field	Delay	Edge AI and optimal resources allocation design.
[11]	Edge computing for offloading strategies	Delay	AV use cases and the V2X communication technologies
[14]	Illustration of different use cases and the presence of an edge data-offloading module	Latency	Artificial intelligence at the cloud layer. The execution of these offloading strategies
[12]	Different V2X communication prototypes	E-E latency	The communication between OBUs/ RSUs, and edges or clouds
[13]	They are focusing only on minimizing the power consumption on the edge platform	Power	Edge computing for autopilot services virtualization Performance evaluation in terms of safety
[14]	Computing resources migration to data	Computing delay	autopilot
[16]	Vehicle world model	Confidence level	Static world model, navigation safety and communication latency indicators
[18]	Architectures for SDN based V2X services	Computing	V2X communication protocols in presence of the SDN paradigm Management issues and safety indicators
[42]	MEC capability to process and broadcast precise positioning	Positioning	The field of MEC may include other uses cases such as virtual or programmable autopilots.
[43]	Computing architecture partitioning and offloading algorithm in connected and autonomous vehicle applications	Service chain delay	Optimal partitioning and offloading strategies in edge-computing servers
[20]	Edge-cloud computing model for autonomous vehicles using Autoware open source	Real time processing	Vehicle computing resources management and vehicle to edge communications
[21]	SURROGATES platform NFV-based OBU instances launching according to ETSI-MANO standards	NFV	Optimal OBU management (optimization) algorithm at the edge virtualization domain
[44]	End to end self-driving car	DNN	Edge assistance to ease processing of intensive autonomous driving functions such as perception procedures
[45]	IETF research group on AV-edge	Delay and safety	Challenges and potential solution of using edge computing in processing IoT AV data.
[22]	Waymo company presents a real self-driving test to train AI models	AI and NN	Edge assistance
[23]	Mathematical modelling of the V2X service placement	Placement time	autopilot virtualization and optimization Heuristics formulation to deal with large autonomous vehicle network scale.

achieve good results in terms of minimizing data transmission cost. However, the impact of obstacles and congestion in communication is not studied in this work. Moreover, we are motivating to use fog and edge computing capabilities to ensure the vehicle self-driving itself.

B. ARTIFICIAL INTELLIGENCE AND EDGE COMPUTING CONVERGENCE

In [29], the authors introduce the paradigm of *edge intelligence* that introduces the convergence of Edge Computing (EC) and Artificial Intelligence (AI). They categorize the utilization of Machine Learning (ML) on the wireless edge into three parts: resource management, networking, and localization. In [30], the authors study the converge of edge computing and deep learning techniques. In [31], the authors propose the use of deep learning for the Internet of Things (IoT) with Edge Computing. In [32] the authors use artificial intelligence methods in recent 5G wireless networking scenarios. In [33], the authors explore the role of Artificial Intelligence (AI), Machine Learning (ML), and Deep Reinforcement Learning (DRL) in the evolution of smart cities. In [34], the authors introduce AI as a Service (AIaaS) on Software-Defined Infrastructure (SDI). In [35], the authors propose an intelligent robust routing using artificial intelligence approaches. In [36] the authors integrates AI modules in the Network Simulator (NS) to simulate real environments and agents spaces. Recent studies [37], [38] focus on better clarification of the convergence of AI, edge computing, DL, and network telecommunication with respect to 5G standards and 6G vision. Besides, a huge effort is devoted to ensure the edge intelligence convergence through universal virtualized architectures and AI techniques [39], [40].

Literature lacks intelligent AI/DRL techniques that may improve the edge computing resources management in the autonomous driving context.

Table 1 summarizes the state-of-the-art approaches in the autonomous vehicle field using edge-computing facility.

Hereafter, we propose our autopilot protocol along with a reference architecture. Then, exact and artificial intelligence based optimization algorithms for compute intensive autonomous driving services allocations are proposed according to a reference architecture.

III. PROPOSED AUTOPILOT PROTOCOL

A. SYSTEM DESIGN

Distributed edge computing enables the multiplexing of heterogeneous and virtualized networks over different AI-IoT architectures corresponding to the isolated tenants and domains to satisfy different application requirements.

The physical infrastructure in next generation networking architectures consists of isolated logical networks, including IoT, IoV, and IoAV which is the topic of this paper. It is worth

mentioning that Unmanned Aerial Vehicles (UAV), Cloud-Radio Access Network (C-RAN), 5G New Radio (NR) [46], and SDN/NFV customers such as vCDN operators [47], Mobile Virtual Network Operator (MVNO) [48] represent also potential end-users in our proposed AI-driven networking architecture.

B. PROPOSED ADVANCED AUTONOMOUS DRIVING ARCHITECTURE USING EDGE ARTIFICIAL INTELLIGENCE

We propose an end-to-end, reliable and low latency communication architecture that allows the allocation of compute-intensive autonomous driving services, in particular autopilot, to share the resources on edge servers and improve the level of performance for autonomous vehicles. In Figure 1, we highlight the proposed Advanced Autonomous Driving (A2D) architecture for the proposed autopilot use case.

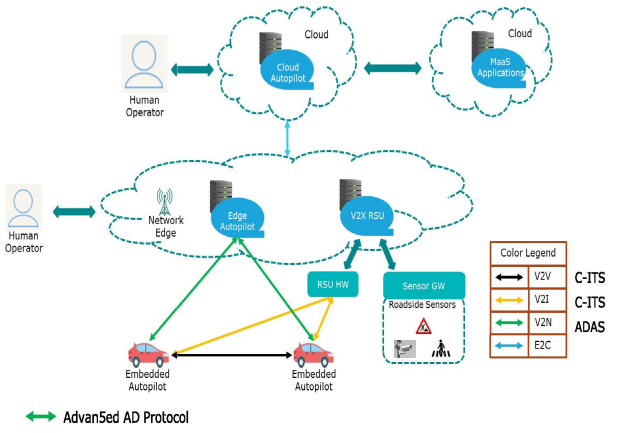


FIGURE 1. Autopilot use case in the artificial intelligence defined optimization framework.

The A2D architecture consists of three main layers/entities as follows:

- The **Centralized Cloud Computing Layer**: It acts as the cloud autopilot and is responsible for processing Non-Real-Time (NRT) autopilot VNFs.
- The **Distributed Network Edges Layer**: It is an intermediate layer that connects OBU vehicles to the cloud. This layer consists of distributed edge servers that assure the cooperation between the virtualized OBUs or vehicles. Moreover, it is responsible for processing and analyzing the offloaded VNFs according to vehicle requests requirements and available resources in the edge servers. It is worth mentioning that resources include computing (CPU, GPU, FPGA), radio (Resource Block, SNR, MCS, CQI), RAM, and storage. The network edge cooperates with the distributed edges and the cloud. It can execute the VNF migration and outsourcing in case of local resources miss.
- The **Autonomous Vehicle Layer**: It is the layer of autonomous vehicles that requests autopilot services chains offloading due to the local resources scarcity.

C. INTEGRATED AUTOPILOT ARCHITECTURE

The main autopilot service chain includes mapping engine (HD Map), localization, perception, decision-making, path-planning, control commands, and Human-Machine Interface (HMI). Each of them has an augmented version responsible for AI insertion. The autopilot layer is running on the top of a Real-Time Operating System (RTOS) such as ROS and ROS2 that schedules incoming tasks on GPU cores. It must ensure low latency by minimizing the makespan of all the jobs and data parallelism techniques to minimize the total missed deadlines. The computing platform of the integrated autopilot includes different Telecommunication Units (TCU) as follows:

- *V2X Modem:* it represents the V2X communication module that uses Dedicated Short Range (DSRC) implementing VANET modules with respect to the IEEE 802.11p standard.
- *Long Term Evolution (LTE) Modem:* it consists of the LTE cellular network modules implemented. One could use OpenAirInterface (OAI) to simulate the LTE-Advanced communication protocols.
- *5G Modem:* Is implements the 5G New Radio (NR) standardized protocols. One could use the open source OpenAirInterface5G [46] software which implements the Baseband Unit (BBU) functions and integrates different Remote Radio Heads (RRH) considered as Software Defined Radio (SDR) modules.
- *WI-FI Modem:* It implements a set of wireless network protocols based on the IEEE 802.11 family of standards.
- *Controlled Area Network (CAN) Modem:* It consists of an in-vehicle robust bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer. It is standardized with ISO 11898. It connects all the vehicle devices in order to facilitate the vehicle control and management.
- *Computing Platform:* It consists of a heterogeneous set of CPU, GPU, TPU and FPGA substrate.

We propose to leverage an edge computing infrastructure to virtualize the above integrated autopilot. The virtualized autopilot includes mainly OpenStack layers for VNFs management [49] and Kubernetes cluster for containers orchestration [50]. It is easily deployed using automation services according to the resources' availability at the network edge. In case of resources miss, the cloud computing servers are used. It is a centralized entity that assures autopilots applications orchestration, and multi-edge management.

D. PROPOSED ADVANCED AUTONOMOUS DRIVING COMMUNICATION PROTOCOL

The main steps of the proposed protocol are detailed hereafter (see Figure 1):

1) **Autopilot Slicing**: each autonomous vehicle can request for offloading some autopilot functions. It requests the near edge node, representing by gNB or

RSU (i.e., a 4G/5G base station) to enable local edge resource discovery and VNFs allocation.

2) **Resources Discovery in Connected Edge Nodes**: when the access point receives autopilot functions offloading request, it generates VNF components or slices. Then, it selects a set of connected edge nodes that can satisfy each VNF requirements in terms of CPU, GPU, RAM, storage. The selected set of connected edge nodes, called *Virtual Edge Servers (VES)*. Resource discovery procedure is based on the computing and networking capabilities of the servers.
3) **Autopilot VNFs Offloading/Allocation**: when the *VES* is selected, the access point starts the process of VNF offloading by allocating each slice a free device resources (from the selected *VES* that can satisfy the slice request requirements). It is worth mentioning that an optimization algorithm is used to select the optimal points of operations where VNFs can be offloaded according to the aforementioned system and network requirements. Still, the cloud computing may represent a solution in the case of edge resources miss. This case may occur when the access point cannot select a VES that can meet the demands of the autonomous vehicles set.
4) **VNF Components Graph:** this is the optimization results of the allocation procedures that indicate the placement of each VNF component. After launching the VNFs in the *VES*/*cloud* servers, optimal control commands are sent directly to the access point.
5) **Results Forwarding:** in the last step, the access point forwards optimal control commands to the autonomous vehicle while satisfying its requirements.

For the sake of clarity, we show in Figure 2 the main communication steps between the edge computing and the connected autonomous vehicle as follows:

- Connected autonomous vehicles send instantaneous states such as position, speed, and next decision of the autopilot to the Edge/Cloud.

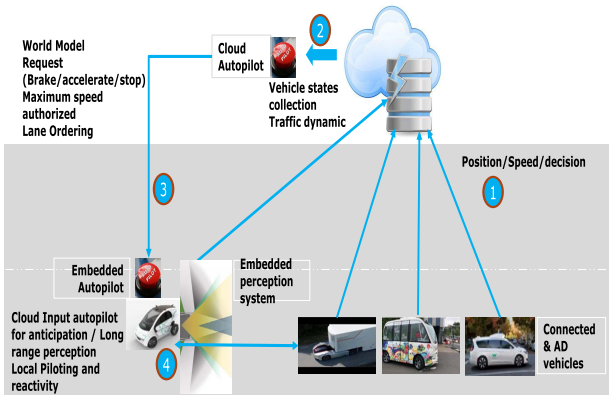


FIGURE 2. The communication steps between the centralized Edge and the distributed connected AD vehicles.

- The Edge/Cloud Autonomous Driving (AD) service collects the raw data, creates the world model for each section of the road, and communicates with Cloud AD Autopilot.
- The Edge/Cloud AD Autopilot sends the global model, generates a high level request for each autonomous vehicle node such as speed request and lane request positioning.
- The Integrated Autopilot merges the Edge/Cloud autopilot inputs and embedded/local inputs to decide to anticipate and act locally.

As explained above, the A2D protocol needs some intelligent optimization algorithms that allocate autopilot VNFs to the optimal/near-optimal edge servers.

IV. OVEAP OPTIMIZATION MODEL

The optimization of autopilot's VNFs placement in edge computing architectures has achieved more attentiveness. It is similar to the placement of Virtual Machines (VMs), where the VNFs are composed of containers or VMs that can execute the needed network functions. We propose a mathematical programming model based on Integer Linear Programming (ILP) in order to model the autopilot VNFs (i.e., a service chain: traffic flow automation between the instantiated functions) offloading in the virtualized edge architecture. The algorithm takes as inputs the system capacity in terms of storage, networks and computing. It aims then to optimally place autopilot VNFs upon the virtual edges. Autopilot VNFs are offloaded in order to increase the safety and decrease the end-users and network devices loads. The optimization algorithm for service offloading in VMEC is modeled, implemented, and evaluated in the next subsections.

A. OVEAP MULTI-RESOURCES AWARE MATHEMATICAL FORMULATION

For the sake of better clarifying the mathematical formulation, we propose the notation of the main parameters. Let *MEC* be the set of homogeneous edge servers in terms of vCPU, vGPU and vStorage resources. We use *EA* as a set of autopilots, where each one of them, has a chain of VNFs to be processed at the edge periodically.

In this section, we specify the parameters and the constraints that are defined and proposed in formulating the optimization/analytic model. This formulation takes as input the multi-resources requirements of the autonomous vehicles and determines the placement of autopilot VNFs to the optimal location. OVEAP will speed up the processing of virtual functions by allocating the available resources while ensuring that it does not exceed the edge server capacity. We quote in Table 2 the main system parameters and decision variables of the proposed mathematical programming approach.

The binary variable x indicates the placement of the autopilot VNF on the MEC Server $mec \in \mathcal{MEC}$. It represents a Service Instantiate Graph (SIG) that defines the optimal points

TABLE 2. A2D mathematical notation.

Notations	*Definition*
$\mathcal{EA}$	The set of autopilots in terms of services Chain.
$\mathcal{VNF}$	The set of autopilot VNFs. Each autopilot VNF is composed of slices.
$\mathcal{AV}$	The set of Autonomous Vehicles.
$\mathcal{MEC}$	The set of Mobile Edge Computing Servers available at the network edge.
$\mathcal{L}_{ea}$	The number of VNFs in each autopilot $ea \in \mathcal{EA}$. It represents the length of the autopilot.
$\mathcal{C}^{mec}$	The maximum computing capacity vCPU available in the MEC server $mec \in \mathcal{MEC}$.
$\mathcal{G}^{mec}$	The maximum computing capacity vGPU available in the MEC server $mec \in \mathcal{MEC}$.
$\mathcal{S}^{mec}$	The maximum storage capacity available in the MEC server $mec \in \mathcal{MEC}$.
$\mathcal{R}^{mec}$	The maximum memory capacity available in the MEC server $mec \in \mathcal{MEC}$.
$\mathcal{B}^{mec}$	The maximum bandwidth (streaming) capacity available in the MEC server $mec \in \mathcal{MEC}$.
$c_{ea,vnf}$	Required vCPU resources for the VNF vnf of the autopilot services chain ea.
$g_{ea,vnf}$	Required vGPU resources for the VNF vnf of the autopilot services chain ea.
$s_{ea,vnf}$	Required vStorage resources for the VNF vnf of the autopilot services chain ea.
$r_{ea,vnf}$	Required vRAM resources for the VNF vnf of the autopilot services chain ea.
$b_{av}^{\mathcal{L}_{ea}}$	Required Vehicle to Network (V2N) bandwidth used for sending the Last autopilot functionality to the Autonomous Vehicle.
Decision variables	*Definition*
$x_{ea,vnf}^{mec}$	A binary variable that allocates the autopilot VNF $vnf \in \mathcal{VNF}$ of the autopilot services chain $ea \in \mathcal{EA}$ to the MEC server $mec \in \mathcal{MEC}$.
y^{mec}	A binary variable that indicates if the MEC server $mec \in \mathcal{MEC}$ is used to process autopilot VNFs. It indicates also if it satisfies the autonomous vehicle request $av \in \mathcal{AV}$.

of operations where Autopilot VNFs should be allocated.

$$x_{ea,vnf}^{mec} = \begin{cases} 11 & \text{if the VNF } vnf \text{ of the Autopilot} \\ & ea \in \mathcal{EA} \text{ is allocated on the MEC server } mec \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

Further, a binary variable y is needed to track the MEC server's computing and networking utilization. It is formulated as follows:

$$y^{mec} = \begin{cases} 11 & \text{if the MEC server } mec \in \mathcal{MEC} \text{ is used} \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

The general ILP algorithm tries to maximize the succeeded placement of the autopilot VNFs while minimizing the number of active MEC servers. Then, the objective function can be formulated as follows:

$$\max \left(OBJ = \sum_{ea \in \mathcal{EA}} \sum_{mec \in \mathcal{MEC}} x_{ea,\mathcal{L}_{ea}}^{mec} - \sum_{mec \in \mathcal{MEC}} y^{mec} \right) \quad (3)$$

According to the proposed OVEAP protocol, the autopilot VNFs placement procedure is constrained by system infrastructure, resources (CPU, GPU, RAM, Storage), and network constraints. Hereafter, we formulate the algorithm constraints related to the autonomous driving service offloading in virtual edge computing.

- **CPU constraint:** The allocation of autopilot VNFs should not exceed the maximum vCPU computing capacity of each MEC servers

$$\sum_{ea \in \mathcal{EA}} \sum_{vnf \in \mathcal{VNF}} c_{ea,vnf} x_{ea,vnf}^{mec} \leq \mathcal{C}^{mec} y^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (4)$$

- **GPU constraint:** The allocation of autopilot VNFs should not exceed the maximum vGPU computing capacity of each MEC servers

$$\sum_{ea \in \mathcal{EA}} \sum_{vnf \in \mathcal{VNF}} g_{ea,vnf} x_{ea,vnf}^{mec} \leq \mathcal{G}^{mec} y^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (5)$$

- **Storage constraint:** The allocation of autopilot VNFs should not exceed the maximum vStorage computing capacity of each MEC servers

$$\sum_{ea \in \mathcal{EA}} \sum_{vnf \in \mathcal{VNF}} s_{ea,vnf} x_{ea,vnf}^{mec} \leq \mathcal{S}^{mec} y^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (6)$$

- **RAM constraint:** The allocation of autopilot VNFs should not exceed the maximum vRAM computing capacity of each MEC server

$$\sum_{ea \in \mathcal{EA}} \sum_{vnf \in \mathcal{VNF}} r_{ea,vnf} x_{ea,vnf}^{mec} \leq \mathcal{R}^{mec} y^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (7)$$

- **Bandwidth constraint:** Constraint (8) assures that the cost of sending ea commands after the processing of all the chain, by a MEC server mec should be less than or equal to the maximum MEC server $N2V$ (or $V2N$) bandwidth capacity.

$$\sum_{av \in AV} \sum_{ea \in EA} b_{av}^{\mathcal{L}_{ea}} x_{ea,\mathcal{L}_{ea}}^{mec} \leq \mathcal{B}^{mec} y^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (8)$$

- **No parallelism per VNF constraint:** Constraint (9) guarantees that only one optimal MEC server should process each autopilot VNF

$$\sum_{mec \in \mathcal{MEC}} x_{ea,vnf}^{mec} \leq 1, \quad \forall ea \in \mathcal{EA}, vnf \in \mathcal{VNF} \quad (9)$$

- **Chaining constraint:** Constraint (10) assures the chaining between autopilot VNFs allocation

$$\sum_{mec \in \mathcal{MEC}} x_{ea,vnf}^{mec} = \sum_{mec \in \mathcal{MEC}} x_{ea,vnf+1}^{mec}, \quad \forall ea \in \mathcal{EA}, vnf \in \mathcal{VNF} \setminus \{\mathcal{L}_{ea}\} \quad (10)$$

- **Minimum server's constraint:** Constraint (11) assures a minimum selection of MEC servers available to process autopilot VNFs.

$$y^{mec} \leq \sum_{ea,vnf} x_{ea,vnf}^{mec}, \quad \forall mec \in \mathcal{MEC} \quad (11)$$

- **AV-RSU/gNB link constraint:** Constraint (12) guarantees the 5G quality on link capacities between MEC servers sources *mec* and autonomous vehicles sinks *av*. Indeed, network links should allow IoV data to be sent and control commands to be received.

$$b_{av}^{\mathcal{L}_{ea}} \leq L^{mec,av} \quad \forall mec, av \in \mathcal{AV} \cup \mathcal{MEC} \tag{12}$$

- **Latency constraint:** Constraint (13) assures that the autopilot VNFs are processed before the deadline d

$$\sum_{ea \in \mathcal{EA}} x_{ea,vnf}^{mec} t_{ea,vnf} \leq d \quad \forall ea \in \mathcal{EA}, vnf \in \mathcal{VNF} \tag{13}$$

where $t_{ea,vnf}$ is the processing time matrix of the autopilot VNF (ea, vnf).

- **Non-negativity constraints:** Used variables x and y are binary in order to decide efficiently (without relaxation) about the autopilot VNFs allocation.

$$x_{ea,vnf}^{mec} \in \{0, 1\} \quad \forall ve \in \mathcal{MEC},\ ea \in \mathcal{EA},\ vnf \in \mathcal{VNF} \tag{14}$$

$$y^{mec} \in \{0, 1\}, \quad \forall mec \in \mathcal{MEC},\ ea \in \mathcal{EA},\ vnf \in \mathcal{VNF} \tag{15}$$

B. OVEAP: COMPLEXITY AND TRIGGERS

1) OVEAP COMPLEXITY

OVEAP algorithm is a non-deterministic polynomial time approach which is feasible with a few instances. It is an NP problem that has an exponential number of feasible solutions.

2) OVEAP TRIGGERS

OVEAP algorithm is proposed to be executed in an autopilot manager entity with respect to ETSI standards. It has to control, manage, and orchestrate the VNFs running autopilot nodes. This offloading is executed after the following triggers:

- System resources (computing, storage, and memory) constraints through $c_{ea,vnf}$, $g_{ea,vnf}$, $s_{ea,vnf}$ and $r_{ea,vnf}$ parameters.
- Networking constraint through $L^{mec,av}$ and $b_{av}^{\mathcal{L}_{ea}}$ parameters.
- Autopilot offloading requests prediction: OVEAP predicts the incoming autopilots as well as the corresponding VNFs. Then, it makes the decision about the optimal points of placements.

Once satisfying the above requirements, the algorithm is executed periodically.

Exact optimization algorithms are non-deterministic polynomial time approaches. In the following section, we propose artificial intelligence techniques to solve the above optimization problem. Autopilot VNFs are items to be offloaded into distributed edge servers according to the resources discovery procedure.

V. DVEAP: DRL-BASED VIRTUAL EDGE AUTOPILOT PLACEMENT ALGORITHM

Artificial Intelligence defined optimization tries to replace the tedious process of ILP by recent AI techniques. We insert Deep Learning modules at the network edge that collect, process, and analyze the raw data. Then, online decisions are taken in order to self-organize the network. Still, data-centers are used to process the big data that does not require real-time optimization. The AI-driven placement of autopilot VNFs at the network edge provides a lot of benefits, including safety and efficient processing. This leads to a minimum latency and enhances the overall path planning and driving qualities.

A. DVEAP: THE PROPOSED RL MODEL

We consider distributed edge servers with multi-resources (CPU, GPU, RAM, Storage, and Bandwidth) that represent the network environment. Then, autopilot VNFs are jobs that arrive to the cluster with an online fashion in discrete time-steps as shown in Figure 3. At each time step, the cluster manager/scheduler chooses a VNF to place according to the Deep-Q-Network (DQN) agent. This latter predicts the near optimal actions using Deep Neural Network. We assume that the VNF demands in terms of required resources are known upon arrival.

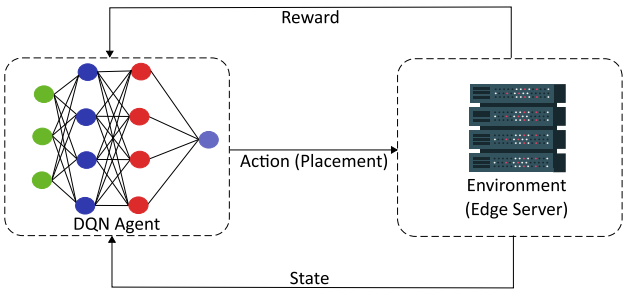


FIGURE 3. Reinforcement learning technique: Placement on virtual edge server use case.

1) THE STATE SPACE

We represent the state space s_t as the current placement of autopilot VNFs on server slots.

2) THE ACTION SPACE

The action space a_t is the placement of a computing-intensive autopilot VNF on a server slot. The placement takes into consideration the server capacity (in terms of slots). In fact, the agent will not place the autopilot VNF on a server slot if it is occupied by running another VNF. The action is monotype where the agent process VNFs one by one until processed all the incoming user requests.

3) THE REWARD SPACE

The proposed reward r is the placement cost of such a VNF. It is measured as the number of servers used after performing

an action. It is formulated as follows

$$r_t = \begin{cases} i & \text{if } i \text{ Edge Servers are Occupied} \\ 0 & \text{Otherwise.} \end{cases}$$

As shown in the above equation 16, the main objective of the agent is to minimize the total opened servers. In other words, the agent objective will be minimizing the total discounted cumulative rewards.

B. DVEAP: THE PROPOSED DQN-DRL

In the DRL algorithm, we use Deep Neural Networks (DNN) to approximate the above RL model while considering the same state-action-reward state spaces. A succession of layers of neural networks are used to map the input state to the output action. In Algorithm 1, we describe the pseudocode of our proposed DRL algorithm.

Algorithm 1 DRL-based Virtual Edge Autopilot Placement (DVEAP) at The Network Edge

1: **Input:** $\mathcal{EA}$, $\mathcal{VNF}$, $\mathcal{L}_{ea}$, $\mathcal{MEC}$
2: **Output:** Q^* (i.e., x and y)
3: Initialize a reply memory D
4: Initialise action-value Q with random weights
5: Observe the initial state s
6: **repeat**
7: Select a server a.
- with probability ϵ select a random server
- Otherwise select the server that has the $\max_{a'} Q(s, a')$

8: Place the autopilot SFC's VNF on the Edge Computing Server a.
9: Observe the incurred allocation cost r and the new edge state s' (new allocation and another incoming autopilot SFC's VNFs)
10: Store the experience $\{s, a, r, s'\}$ in the replay memory.
11: Sample a random transition from the replay memory.
12: Calculate the target for each mini-batch transition $(r + \gamma \times \max_a Q(s', a'))$
13: Train the Q network using the following loss $Loss = \frac{1}{2} * (r + \gamma \times \max_{a'} Q(s', a') - Q(s, a))^2$
14: s = s'
15: **until** No incoming autopilot VNFs from all the SFC

We use the Stochastic Gradient Descent (SGD) algorithm [51] to perform Deep-Q-Network (DQN) agent training. Then, we tune the main hyperparameter to decide about optimal DNN configurations such as epoch/iteration numbers, optimizer parameters, and action selection strategies. As shown in Algorithm 1 Line 13, SGD algorithm uses the following Bellman equation in order to minimize the loss function (squared error) between target and current Q-values: $Q(s, a) = r + \gamma \times \max_{a'} Q(s', a')$. Then, DNN weight are updated using back-propagation process. The training procedure is an offline procedure which is performed before the deployment of the DQN-DRL algorithm at the edge network. Then, the algorithm will be used for real time resources allocation. Network operators can update the pretrained model according to their needs. It is worth mentioning that the training time depends on the available resources and data types. In our case, we use the structured data types which do not require a significant time.

The DRL based approach consists in reducing ILP time and RL state space complexities by reducing the number of iterations to be considered in the optimization while including more parameters.

VI. PERFORMANCE EVALUATION: DRL-DVEAP VERSUS ILP-OVEAP

In this section, we evaluate our proposed solutions as follows:

- Optimal autopilot resources allocation strategy is evaluated in small scale IoAV networks.
- For the sake of dealing with large-scale networks, where many AV autopilots require offloading at the same time, AI-DRL method is evaluated and compared with the optimal approach.

We evaluate the proposed algorithms (ILP and DRL) using different optimization tools. CPLEX [52] is used to evaluate the exact ILP model, while TensorFlow and Keras [53] are used to configure and implement the DRL algorithm. As explained above, we consider the following autopilot VNFs: *vPerception*, *vLocalisation*, *vPlanner*, and *vControl*. Indeed, the autopilot chain is composed of a set of VNFs. Recall that the objective is to place each autopilot VNF in the edge server, while assuring the chaining of all the VNFs of the same autopilot. Tables 3 and 4 show the different configurations used in the evaluation.

TABLE 3. Small scale configuration (AES: Autopilot Edge Servers).

Autonomous Vehicles	*Autopilot Chain*
Vehicle 1	*vPerception*
Vehicle 2	*vPerception, vLocalisation*
Vehicle 3	*vPerception, vLocalisation, vPlanner*
Vehicle 4	*vPerception, vLocalisation, vControl*
Vehicle 5	*vPerception, vLocalisation, vPlanner, vControl*
Embedded Autopilots	*Number of servers*
Embedded vehicle	1
Edge	*Capacity (servers/slots)*
Edge 1	10/5
Edge 2	10/10

TABLE 4. Summary of the parameters (AES: Autopilot edge servers).

Parameter	*Min value*	*Max value*
Number of AES	3	5
Number of Autonomous vehicles	5 (small scale)	100
Available vCPU in AES	50 (slot)	200 (slot)
Available vGPU in AES	10 (slot)	150 (slot)
Available vRAM in AES	50 (Gigabytes)	200 (Gigabytes)
Available vStorage in AES	10 (Terabytes)	100 (Terabytes)
Available Bandwidth in AES	10 (Mbps)	150 (Mbps)
Latency in Autopilot Chain	1/8 (ms)	1 (ms)

It is worth mentioning that we evaluated the proposed algorithms in small and large scale using simulation. We simulate the datacenter using the "x86" architecture, the "Linux" operating system and the "Xen" hypervisor. For the sake of better clarification, network and system parameters are shown in Table 5. Recall that the simulation considers the offloading of sensor data to the edge. This latter creates the world model according to the vPerception VNF. Then, the vLocalisation adds the position of the vehicle in the environment. The vPlanner traces the chart between the source and the destination and the vControl executes the path by sending the control command to the embedded vehicle.

TABLE 5. Summary of simulation parameters.

Network Parameters	*Value*			
Wireless channel model	The 5G NR protocol [54]			
Mobility model	Low, Medium, and High mobilty			
VNF autopilot Parameters	vPerception	vLocalisation	vPlanner	vControl
Usage percentage (percent)	50	20	20	10
Task interarrival time (ms)	2	2	2	2
Active period duration (ms)	0.5	0.2	0.2	0.1
Upload data size (MB)	10-50	✗	✗	✗
Download data size (MB)	✗	✗	✗	1-10
Vm utilization of tasks (percent)	1-20	1-5	1-30	1-10

A. KEY PERFORMANCE INDICATORS (KPIs)

For the interest of assessing the efficiency of the proposed approaches (OVEAP ILP and DVEAP DRL), we propose different KPIs as follows:

- **Total Edge Servers Utilization (TESU):** it represents the number of servers allocated for the autopilot service functions chain.
- **Total Edge Servers Allocation Time (TESAT):** it represents the required time for autopilot service functions chain allocation.
- **Successfully Allocated Autopilots:** it represents the number of successfully offloaded autopilots at the network edge.
- **Average Network Delay:** It represents the average network delay between autonomous vehicles and edge computing servers.
- **Service Time:** it represents the end-to-end response time including autopilot VNF submission, resources discovery, VNF offloading, and computation results forwarding to the end-user. It is coupled with the availability of resources and the offloading decisions, where an efficient service is characterized by low execution time.
- **Processing Time:** this KPI represents the duration needed by the OVEAP controller to complete the execution of all submitted autopilot VNFs. In general, the availability of resources among virtual edge servers is an important metric that enhances the efficiency of processing through decreasing the processing time.

B. COMPUTING ARCHITECTURES BASED ANALYSIS

To quantify the behavior of OVEAP, we compare our proposed OVEAP optimization algorithm with state-of-the-art computing architectures. We quote the relevant architecture as follows:

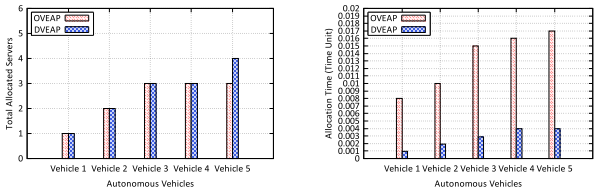
1) **Embedded Computing:** This architecture allows the local execution of autopilot modules, while edge computing is still enabled to receive VNFs.
2) **Edge Computing:** This architecture prioritizes edge computing servers for autopilots services offloading.
3) **Cloud Computing:** This architecture allows only the use of the centralized cloud computing.

C. OBTAINED RESULTS

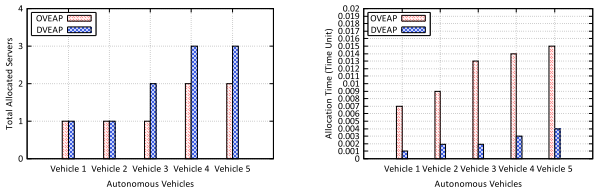
In this section, we introduce two scenarios in order to evaluate the optimization algorithms. The first scenario targets small networks, while the second deals with large scale networks.

1) THE SMALL-SCALE SCENARIO

For the sake of better selecting the appropriate allocation strategy (OVEAP or DVEAP), different network scales (i.e. small and large) are considered as follows. In Figure 4, we show the total resources' utilization at the network edge for *Edge*1 and *Edge*2 respectively in small scale network. In Figures 4a and 4c, we plot autonomous vehicles configurations against the *TESU* metric. Results show that the efficiency of the proposed DRL-DVEAP algorithm, since it converges to the exact ILP OVEAP in terms of placement cost. In addition, Figures 4b and 4d show that DVEAP algorithm gives a non-significant placement time (in terms of a few micro seconds) compared to OVEAP that still has a feasible placement time.



(a) Autopilot Edge Server Allocation Cost (Edge 1) **(b)** Autopilot Edge Server Allocation Time (Edge 1)



(c) Edge Server Allocation Cost (Edge 2) **(d)** Edge Server Allocation Time (Edge 2)

FIGURE 4. Total resources utilization of autopilot edge servers in small-scale network at Edge 1 and 2 (Time unit is in second).

2) THE LARGE-SCALE SCENARIO

We first considered the entire autopilot allocation. In Figure 5, we plot the main KPIs against the number of autonomous vehicles. We consider the exact ILP OVEAP approach taking into account the above constraints.

In Figure 5a, we plot the network average delay between autonomous vehicles and edge computing servers.

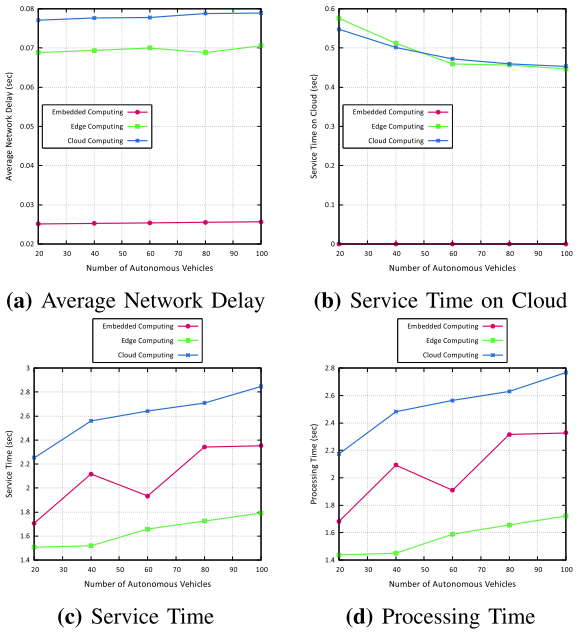
In Figure 5b, we show that the edge computing reduces the computing load on cloud computing when the autonomous vehicles are increasing.

In Figure 5c, we show that the edge computing reduces the average service time comparing to the embedded and cloud computing architectures. Indeed, embedded autopilot offload the heavy VNF to the near edge for efficient edge processing.

In Figure 5d, we show that edge computing reduces the processing comparing to cloud computing.

a: THE IMPACT OF AUTOPILOT VNFs ALLOCATION ON THE PROCESSING TIME

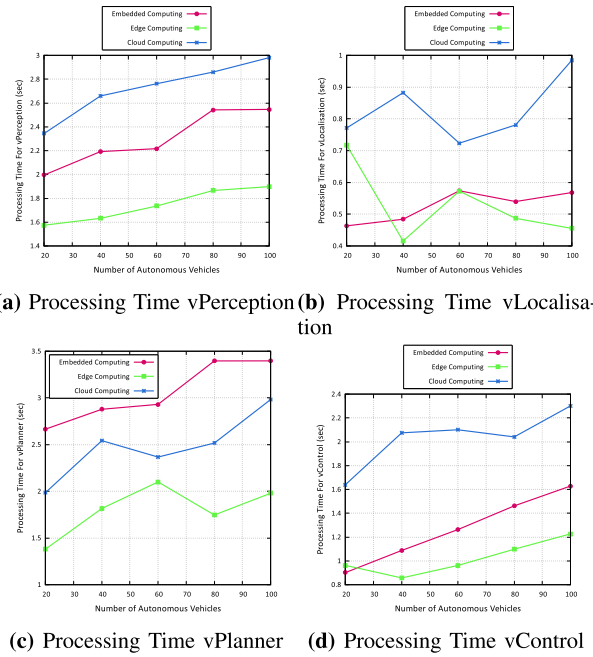
In Figure 6, we show the processing time variation when increasing the number of autonomous vehicle. Results show



(a) Average Network Delay **(b)** Service Time on Cloud

(c) Service Time **(d)** Processing Time

FIGURE 5. **Performance evaluation in large-scale networks.**



(a) Processing Time vPerception **(b)** Processing Time vLocalisation

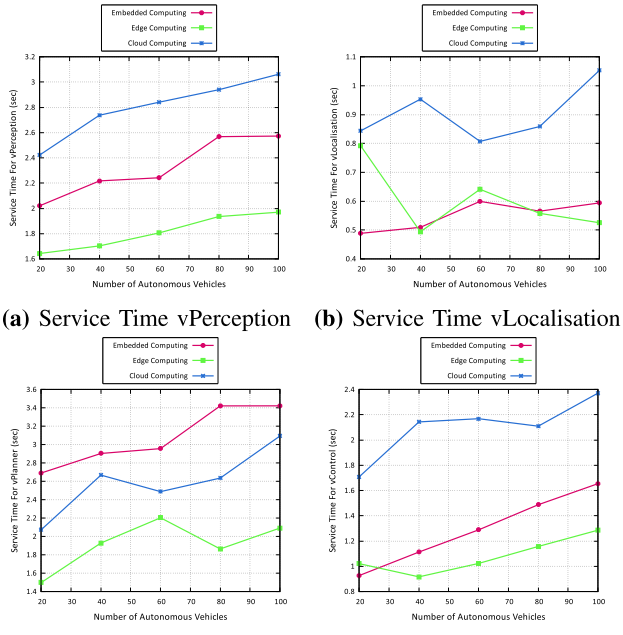
(c) Processing Time vPlanner **(d)** Processing Time vControl

FIGURE 6. **Processing time evaluation in large-scale networks.**

the proposed edge computing architecture reduces the processing time of the autopilot vPerception, vLocalisation, vPlanner, and vControl. Moreover, we show that the vPerception and the vPlanner functions require more significant processing time than other autopilot VNFs.

b: THE IMPACT OF AUTOPILOT VNFs ALLOCATION ON THE SERVICE TIME

In Figure 7, we show the service time variation when increasing the number of autonomous vehicle. Results prove the



(a) Service Time vPerception **(b)** Service Time vLocalisation

(c) Service Time vPlanner **(d)** Service Time vControl

FIGURE 7. **Service time evaluation in large-scale networks.**

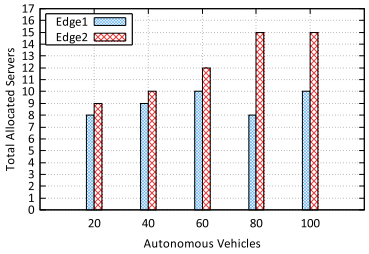


FIGURE 8. **Total resources utilization in large-scale network using DVEAP approach.**

feasibility and the efficiency of the proposed edge computing architecture by reducing the service time shown. This result validates the edge assistance autopilot function's offloading, since that this KPI reflects the total application time.

In Figure 8, we quantify the behavior of the DRL DVEAP algorithm in a large-scale network according to the different edges configurations. We plot autonomous vehicle number ranging from 20 to 100 against *TESU*. Results show that increasing the computing capacity helps in better offloading autopilot functions.

In Figure 9, we show the limit of the DRL approach in a very dense network constituted by a *hundred* of autonomous vehicles requiring services offloading. Results show that most of the autopilots are successfully allocated on MEC servers.

In Figure 10, we evaluate the DRL approach in terms of VNF offloading time. The result proves the feasibility and the efficiency of the proposed algorithm in large scale.

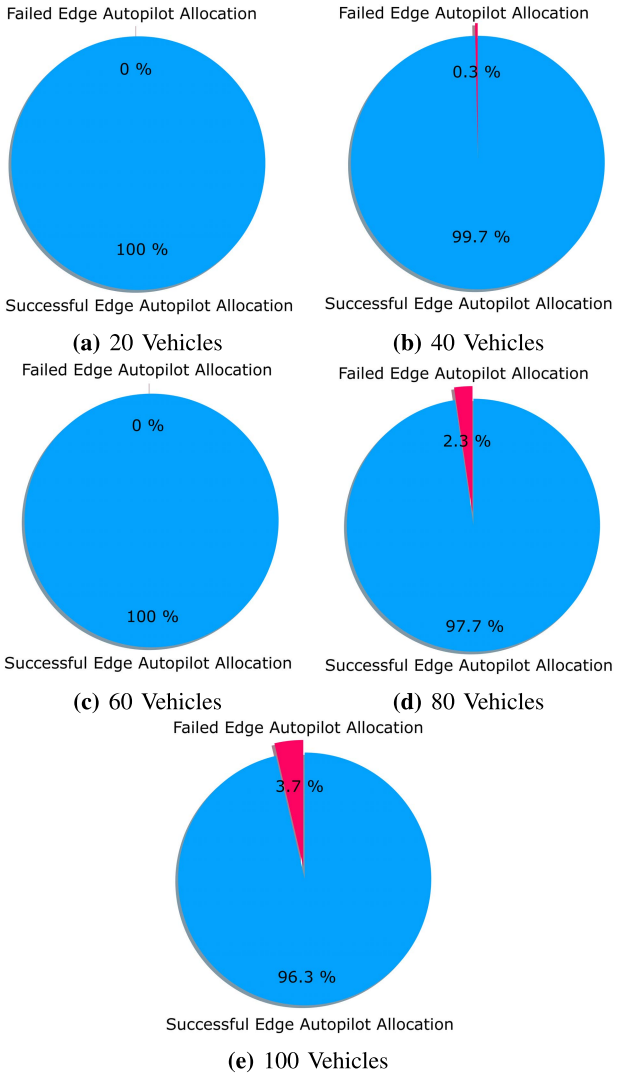


FIGURE 9. Total resources utilization in large-scale network using DVEAP approach.

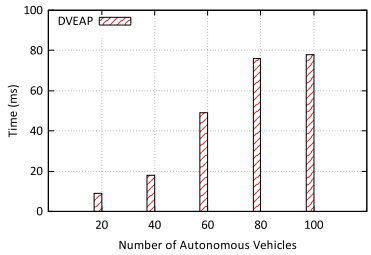


FIGURE 10. Autopilots VNFs offloading time based on the DVEAP model.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an Artificial Intelligence approach for autopilot placement (offloading) at the network edge. First, we have proposed an end-to-end architecture for edge computing assisted autonomous driving. Then, we have proposed an optimal allocation approach (OVEAP) that decides about optimal autopilot functions placement. Further, to deal with dense IoAV networks, a deep reinforcement learning approach (DRL-DVEAP) is formulated and implemented. Based on different configurations of edge environments, the proposed DRL achieves a good result in terms of offloading cost and processing time. In the future work, we will focus on autopilot VNF migration in case of near edge discovery scenario.

REFERENCES

[1] L. L. Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey on imitation learning techniques for end-to-end autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 1, 2022, doi: 10.1109/TITS.2022.3144867.

[2] H. Alghodhaifi and S. Lakshmanan, "Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches," *IEEE Access*, vol. 9, pp. 151531–151566, 2021.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[4] H. Khayyam, B. Javadi, M. Jalili, and R. N. Jazar, "Artificial intelligence and Internet of Things for autonomous vehicles," in *Nonlinear approaches in Engineering Applications*. Cham, Switzerland: Springer, 2020, pp. 39–68.

[5] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–7.

[6] A. Belogaev, A. Elokhin, A. Krasilov, E. Khorov, and I. F. Akyildiz, "Cost-effective V2X task offloading in MEC-assisted intelligent transportation systems," *IEEE Access*, vol. 8, pp. 169010–169023, 2020.

[7] F. Riquelme, R. Olivares, F. Muéz, X. Molinero, and M. Serna, "Hypo-driver: A multiview driver fatigue and distraction level detection system," *Comput., Mater. Continua*, vol. 71, no. 1, pp. 1999–2007, 2022.

[8] *Artificial Intelligence & Autopilot*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.tesla.com/AI

[9] P. Bhardwaj, "An Overview of ADAS in internet of vehicles," in *Internet of Vehicles and Its Applications in Autonomous Driving* (Unmanned System Technologies), N. Gupta, A. Prakash, R. Tripathi, Eds. Cham, Switzerland: Springer, Jan. 2021, pp. 77–92, 10.1007/978-3-030-46335-9_6.

[10] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[11] S. Kanti Datta, M. Irfan Khan, L. Codeca, B. Denis, J. Harri, and C. Bonnet, "IoT and microservices based testbed for connected car services," in *Proc. IEEE 19th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2018, pp. 14–19.

[12] L. Liu, B. Wu, and W. Shi, "A comparison of communication mechanisms in vehicular edge computing," in *Proc. 3rd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2020, pp. 1–7.

[13] J. Tang, S. Liu, L. Liu, B. Yu, and W. Shi, "LoPECS: A low-power edge computing system for real-time autonomous driving services," *IEEE Access*, vol. 8, pp. 30467–30479, 2020.

[14] *Automotive Edge Computing Consortium: Driving the Vision of Network and Computing Infrastructure for Connected Car Big Data*. Accessed: Mar. 1, 2022. [Online]. Available: https://aecc.org/

[15] J. Janai, F. Guney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Found. Trends Comput. Graph. Vis.*, vol. 12, nos. 1–3, pp. 1–308, 2020.

[16] A. Furda and L. Vlacic, "An object-oriented design of a world model for autonomous city vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 1054–1059.

[17] *How-to: Cycabtk Simulator*. Accessed: Mar. 1, 2022. [Online]. Available: https://robotml.github.io/HowToUseRobotML/ExamplesOfUse/HowToCycabTK.ht%ml

[18] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–19, Feb. 2019.

[19] H. Chishiro, K. Suito, T. Ito, S. Maeda, T. Azumi, K. Funaoka, and S. Kato, "Towards heterogeneous computing platforms for autonomous driving," in *Proc. IEEE Int. Conf. Embedded Softw. Syst. (ICESS)*, Jun. 2019, pp. 1–8.

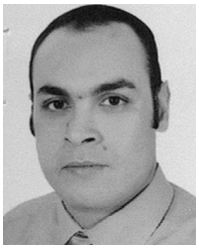
[20] E. Molina, O. Lazaro, M. Sepulcre, J. Gozalvez, A. Passarella, T. P. Raptis, A. Ude, B. Nemec, M. Rooker, and F. Kirstein, "The autoware framework and requirements for the cognitive digital automation," in *Proc. Work. Conf. Virtual Enterprises*. Springer, 2017, pp. 107–117.
[21] J. Santa, P. Fernández, J. Ortiz, R. Sanchez-Iborra, and A. Skarmeta, "SURROGATES: Virtual OBUs to foster 5G vehicular services," *Electronics*, vol. 8, no. 2, p. 117, Jan. 2019.
[22] V. G. Cerf, "A comprehensive self-driving car test," *Commun. ACM*, vol. 61, no. 2, p. 7, Jan. 2018.
[23] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1380–1392, Apr. 2021.
[24] S. Liu, J. Tang, C. Wang, Q. Wang, and J.-L. Gaudiot, "Implementing a cloud platform for autonomous driving," 2017, *arXiv:1704.02696*.
[25] N. S. Yeshodara, N. S. Nagojappa, and N. Kishore, "Cloud based self driving cars," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Oct. 2014, pp. 1–7.
[26] S. Kumar, S. Gollakota, and D. Katabi, "A cloud-assisted design for autonomous driving," in *Proc. 1st, Ed., MCC Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 41–46.
[27] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 148–153.
[28] D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *Proc. IEEE 3rd Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2016, pp. 247–251.
[29] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.
[30] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.
[31] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
[32] C.-X. Wang, M. D. Renzo, S. Stanczak, S. Wang, and E. G. Larsson, "Artificial intelligence enabled wireless networking for 5G and beyond: Recent advances and future challenges," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 16–23, Feb. 2020.
[33] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of artificial intelligence and machine learning in smart cities," *Comput. Commun.*, vol. 154, pp. 313–323, Mar. 2020.
[34] S. Parsaeefard, I. Tabrizian, and A. Leon-Garcia, "Artificial intelligence as a service (AI-aaS) on software-defined infrastructure," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2019, pp. 1–7.
[35] N. Chen, T. Qiu, X. Zhou, K. Li, and M. Atiquzzaman, "An intelligent robust networking mechanism for the Internet of Things," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 91–95, Nov. 2019.
[36] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "Ns3-Ai: Fostering artificial intelligence algorithms for networking research," in *Proc. Workshop (Ns)*, Jun. 2020, pp. 57–64.
[37] P. Chemouil, P. Hui, W. Kellerer, Y. Li, R. Stadler, R. Stadler, Y. Wen, and Y. Zhang, "Special issue on artificial intelligence and machine learning for networking and communications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1185–1191, Jun. 2019.
[38] L. Zhang, Y.-C. Liang, and D. Niyato, "6G visions: Mobile ultra-broadband, super Internet-of-Things, and artificial intelligence," *China Commun.*, vol. 16, no. 8, pp. 1–14, Aug. 2019.
[39] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.
[40] Y. Ren, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Green intelligence networking for connected and autonomous vehicles in smart cities," *IEEE Trans. Green Commun. Netw.*, early access, Feb. 1, 2022, doi: 10.1109/TGCN.2022.3148293.
[41] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.
[42] D. Hetzer, M. Muehleisen, A. Kousaridas, S. Barmpounakis, S. Wendt, K. Eckert, A. Schimpe, J. Löfhede, and J. Alonso-Zarate, "5G connected and automated driving: Use cases, technologies and trials in cross-border environments," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, pp. 1–19, Dec. 2021.
[43] S. Baidya, Y.-J. Ku, H. Zhao, J. Zhao, and S. Dey, "Vehicular and edge computing for emerging connected and autonomous vehicle applications," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[44] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Müller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.
[45] J. Hong, Y.-G. Hong, X. de Foy, M. Kovatsch, E. Schooler, D. Kutscher. (Jan. 2022). *IoT Edge Challenges and Functions, Internet-Draft Draft-IRTF-T2TRG-IoT-Edge-04, Internet Engineering Task Force, Work in Progress*. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-iot-edge-04
[46] H. Khedher, S. Hoteit, P. Brown, R. Krishnaswamy, W. Diego, and V. Veque, "Processing time evaluation and prediction in cloud-RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[47] H. Khedher, H. Afifi, and H. Moustafa, "Optimal placement algorithm (OPA) for IoT over ICN," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 372–377.
[48] M. Laroui, H. Ibn-Khedher, M. A. Cherif, H. Moungla, H. Afifi, and A. E. Kamel, "SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning," *Trans. Emerg. Telecommun. Technol.*, p. e4211, Jan. 2021, 10.1002/ett.4211.
[49] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online VNF lifecycle management in an MEC-enabled 5G IoT architecture," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4183–4194, May 2020.
[50] Y. Han, S. Shen, X. Wang, S. Wang, and V. C. M. Leung, "Tailored learning-based scheduling for kubernetes-oriented edge-cloud system," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.
[51] B.-C. Zhou, C.-Y. Han, and T.-D. Guo, "Convergence of stochastic gradient descent in deep neural network," *Acta Mathematicae Applicatae Sinica, English Ser.*, vol. 37, no. 1, pp. 126–136, Jan. 2021.
[52] *IBM ILOG CPLEX Optimization Studio Community Edition, Version 12.5*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio
[53] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *J. Educ. Behav. Statist.*, vol. 45, no. 2, pp. 227–248, 2020.
[54] A. P. K. Reddy, M. S. Kumari, V. Dhanwani, A. K. Bachkaniwala, N. Kumar, K. Vasudevan, S. Selvaganapathy, S. K. Devar, P. Rathod, and V. B. James, "5G new radio key performance indicators evaluation for IMT-2020 radio interface technology," *IEEE Access*, vol. 9, pp. 112290–112311, 2021.

HATEM IBN-KHEDHER received the Diploma degree in electrical, electronics, computer and telecommunications engineering from the National Engineering School of Sfax (ENIS), Tunisia, in 2014, the M.S. degree in network and computer science from the Telecom SudParis, in 2015, and the Ph.D. degree in computer science and networks from University Pierre and Marie Curie, in April 2018. He collaborated with Intel Laboraties, USA, in 2016, and Iowa State University, in 2017. Since March 2018, he was a Postdoctoral Researcher at the Laboratoire des Signaux et Systemes (L2S), Centrale-Supelec. In 2019, he was a Research and Development Engineer at IRT SystemX. In 2020, he did an ATER at Université de Paris. He is currently an Engineer with Capgemini Engineering.

MOHAMMED LAROUI received the B.Sc. and M.Sc. degrees in computer science from Djillali Liabes University, Sidi Bel Abbés, Algeria, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in computer science from Paris University and Djillali Liabes University. He visited Telecom SudParis, Paris, France, in 2017, as a Researcher, where he worked on vehicular networks. His research interests include cloud/edge and fog computing, vehicular networks, IoT, next-generation networking, and internet.

HASSINE MOUNGLA (Member, IEEE) has been an Associate Professor with the University of Paris Descartes and a member of the Paris Descartes Computer Science Laboratory (LIPADE), since October 2008. He was a Researcher at INRIA, in 2008, and a Research Fellow at CNRS-LIPN Laboratory, Paris Nord University, in 2007. His research interests include wireless area body networking (WBAN) for medical and health applications, wireless sensor networking, QoS in WSN, middleware for 5G mobile, and sensor networks. He participated and still participates in several national and international research projects. He is with the Technical Program Committee of different ACM and IEEE conferences, including Globecom, ICC, WCNC, PIMRC, IWCMC, and chaired some of their sessions. He is also a reviewer on a regular basis for major international journals. He is a member of IEEE Communication Society.

HOSSAM AFIFI received the Ph.D. degree in computer science from Inria Sophia Antipolis, in 1992. He visited Washington University, St. Louis, as a Postdoctoral Researcher, where he worked on IP switching techniques. He was appointed as an Assistant Professor with ENST Bretagne (now, Mines Atlantic), in the field of high speed networking. He is currently a Professor with Télécom SudParis-Institut polytechnique de Paris, SAMOVAR Laboratory, where he works on machine learning for mobile and security protocols. After his tenure and a sabbatical with Nokia Research Labs, Mountain View, USA, he took the current position with Telecom SudParis since 2000. His current interests include vehicular and user-centric behaviors.



EMAD ABD-ELRAHMAN received the B.Sc. degree in electronics engineering from Mansoura University, Egypt, in 1999, the M.Sc. degree in electronics engineering from the Computers and Systems Department, Mansoura University and National Telecommunication Institute, Egypt, in 2004, and the Ph.D. degree in computer science and telecommunication from the University of UPMC and Telecom SudParis, in 2012. In 2008, he joined the University of UPMC-France (Paris-6) and IMT (Institute Mines- Telecom) Telecom SudParis. He spent three years as a Guest Researcher at the RST Department, Telecom SudParis (IMT)-CEA Saclay-France, from 2014 to 2016. He has been an Associate Professor with the National Telecommunication Institute, Cairo, Egypt, since 2018. His current research interests include networking, optimization, multimedia, multi-modal traffic in ITS, virtualization SDN/NFV, and cloud computing. He is involved in many European and French projects, including the UP-TO-US, DVD2C, and CA-ITS.