On the Design of Federated Learning in Latency and Energy Constrained Computation Offloading Operations in Vehicular Edge Computing Systems

Swapnil Sadashiv Shinde, *Student Member, IEEE*, Arash Bozorgchenani, *Member, IEEE*, Daniele Tarchi, *Senior Member, IEEE*, and Qiang Ni, *Senior Member, IEEE*

***Abstract*—With the advent of smart vehicles, several new latency-critical and data-intensive applications are emerged in Vehicular Networks (VNs). Computation offloading has emerged as a viable option allowing to resort to the nearby edge servers for remote processing within a requested service latency requirement. Despite several advantages, computation offloading over resource-limited edge servers, together with vehicular mobility, is still a challenging problem to be solved. In particular, in order to avoid additional latency due to out-of-coverage operations, Vehicular Users (VUs) mobility introduces a bound on the amount of data to be offloaded towards nearby edge servers. Therefore, several approaches have been used for finding the correct amount of data to be offloaded. Among others, Federated Learning (FL) has been highlighted as one of the most promising solving techniques, given the data privacy concerns in VNs and limited communication resources. However, FL consumes resources during its operation and therefore incurs an additional burden on resource-constrained VUs. In this work, we aim to optimize the VN performance in terms of latency and energy consumption by considering both the FL and the computation offloading processes while selecting the proper number of FL iterations to be implemented. To this end, we first propose an FL-inspired distributed learning framework for computation offloading in VNs, and then develop a constrained optimization problem to jointly minimize the overall latency and the energy consumed. An evolutionary Genetic Algorithm is proposed for solving the problem in-hand and compared with some benchmarks. The simulation results show the effectiveness of the proposed approach in terms of latency and energy consumption.**

***Index Terms*—Vehicular Edge Computing, computation offloading, Federated Learning, latency, energy consumption, Genetic Algorithm.**

I. Introduction

MODERN cities are characterized by the presence of an increased interest in mobility management for a higher urban life sustainability. The ongoing COVID-19 pandemic has increased the importance of a sustainable smart transport infrastructure [1]. In this context, road users are becoming smart, requiring a tighter interaction with the Internet and among them in order to strengthen vehicular requirements [2]. This has introduced the possibility of implementing several services and generated a large amount of data, hence requiring the introduction of suitable computation and storage resources, which vehicles are not capable of providing [3].

Due to a lack of available resources, Vehicular Users (VUs) tend to offload computation tasks to the nearby edge and cloud computing servers [4], [5]. Computation offloading enables several data-intensive and latency-critical services and applications with the potential of enhancing service quality for consumers. Within this scenario, Cloud Computing has been envisaged as one of the most important technologies for assisting Vehicular Networks (VNs) by executing services at remote cloud platforms. However, Cloud Computing results in high response time, which is not acceptable in latency-critical environments, such as VNs. To this aim, resorting to the Edge Computing technology, namely Vehicular Edge Computing (VEC), allowing to deploy edge servers in the proximity of the vehicles, by gaining from the presence of Road Side Units (RSUs) is a promising solution. Even though RSUs can reduce the connectivity latency due to their proximity, they still have limited computational and communication resources. This requires optimized resource management strategies by selecting the proper RSU and the amount of data to be offloaded. In addition, due to their small coverage range, only a limited number of vehicles/VUs can access RSU services without incurring an additional latency (due to, e.g., vehicle handover, service migration).

To cope with the limited RSU resources and to avoid the additional latency costs, the VUs can perform a partial computation offloading, where they can offload a portion of their computation loads towards edge servers and can compute the remaining tasks locally [5]–[7]. Finding a suitable RSU for the remote task processing and jointly optimizing the amount of data to be offloaded allow to reduce the overall latency. However, several factors have to be considered while solving the computation offloading problem in VNs, including VUs velocity, locations, RSU capacity, RSU coverage, RSU density, environmental conditions, nature of roads, etc. In addition, VUs

Manuscript received July 30, 2021; revised November 3, 2021 and December 6, 2021; accepted December 9, 2021. Date of publication December 14, 2021; date of current version February 14, 2022. The review of this article was coordinated by Prof. Swades De. *(Corresponding author: Daniele Tarchi.)*

Swapnil Sadashiv Shinde and Daniele Tarchi are with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40126 Bologna, Italy (e-mail: swapnil.shinde2@unibo.it; daniele.tarchi@unibo.it).

Arash Bozorgchenani and Qiang Ni are with the School of Computing and Communications, Lancaster University, Lancaster LA1 4YW, U.K. (e-mail: a.bozorgchenani@lancaster.ac.uk; q.ni@lancaster.ac.uk).

Digital Object Identifier 10.1109/TVT.2021.3135332

often demand services with a target Quality of Services (QoS) level, such as critical latency requirements, as well as an increased requirement of energy saving mechanisms. Modern and future vehicles are often based on electrical engine, not to name the recently introduced micro-mobility solutions, where energy saving is a condition. In addition, mobility in VNs adds a further complexity dimension during the service provision and needs to be handled carefully. Consequently, the computation offloading problem in VNs is a complex problem to be solved [8].

Machine Learning (ML) is a class of algorithms recently gaining lots of attention due to their ability in managing large amount of data for solving complex decision problems [9]. Their effectiveness has been recently demonstrated in wireless communication networks to solve complex problems, e.g., communication resource management and allocation, spectrum management, power control, base station switching [10], [11]. In traditional centralized ML approaches each agent sends its data to a centralized entity, e.g., a centralized cloud server or a base station, that is in charge of gathering data for the learning process. Several issues, like users' privacy concerns, limited communication resources and a limited amount of energy supply, have narrowed the use of centralized approaches in wireless communication environments. In [12], the authors proposed a novel approach, named Federated Learning (FL) with the potential of avoiding the previously discussed shortcomings. In the case of FL, instead of sending raw data to the centralized entity, users limit their communication only to the ML parameters of the locally trained model. The centralized servers collect the updates from the agents and create the new global update parameters, and send them back to the users. During the FL training phase, the device can participate in several communication rounds aiming at refining the ML model. Each communication round utilizes communication and computing resources.

When applying FL to the vehicular scenario some issues should be considered. On one side, VNs are affected by the limited coverage of RSUs, posing some challenges to the offloading procedure. On the other side, FL requires a centralized node able to gather the information from all the vehicles. In the case of RSUs acting as a centralized server for the FL process, only a reduced number of VUs can participate in the training process due to the RSU coverage limitations. In the past, it had been proven that FL process convergence cost (in terms of latency) can be largely impacted by the number of FL devices participating in it [13]. Recently, air-ground integrated networks are increasing their importance allowing the integration of an aerial communication network, composed of several Low and High Altitude Platforms (LAPs and HAPs), with the terrestrial communication networks, e.g., VNs [14]. HAPs, such as aircrafts, balloons, and airships, have several advantages including a larger coverage area, renewable power source, and long endurance, allowing to help VNs to cope with the stringent user requirements [15]–[17]. In addition, HAPs can be a viable option with respect to the satellite systems with a considerable reduction in the round trip time, low deployment costs, and favorable channel conditions. Thus HAPs can act as a powerful centralized entity with a more global view during the implementation of an FL in VNs scenarios.

During the FL process, resource-constrained wireless nodes are in charge of implementing ML algorithms to infer useful information from their datasets. FL process can converge to a predefined loss function value after a certain number of communication rounds [18], defining the FL process performance. However, if on one side each FL round helps in terms of convergence, it also introduces an additional cost in terms of computational and communication resources to the resource-constrained devices. As a result, there is a trade-off between the FL process accuracy and the training cost, which needs to be explored for resource-constrained communication networks like VNs.

First, we aim at defining a FL-based platform assisted by the HAPs, acting as FL servers, to find a solution for the computation offloading problem in VNs. To this aim, we have considered a three-layer network architecture composed of VNs, RSUs, and HAPs. The first layer contains several VUs, characterized by limited computation and storage resources. They can communicate with the RSUs within a specific range and with at least one HAP. The layer two is composed of RSUs having richer computational and storage capabilities. They can serve as a computation offloading platform for vehicles. In layer three, we use HAPs as FL servers.

The system aims at performing task offloading within a FL architecture. The FL framework enables learning information that is useful for the computation offloading decision phase. Although FL assists a better decision making in computation offloading procedure, it imposes some cost in terms of energy and delay. Since FL consumes some resources in terms of time and energy, hence impacting on the resources left to the users willing to offload, we aim at optimizing the resource sharing between learning and offloading phases. This is done by allocating sufficient time to both phases, by considering the learning convergence toward the optimal offloading parameters estimation. It has to be clarified that in this work we are not investigating the learning procedure, hence we resort to a simplified FL-inspired distributed approach for modeling the interactions of the FL process. In particular, given the limited available resources of each vehicle and RSUs, we aim at jointly optimizing the delay and energy performance of VNs for both FL and offloading phases.

The system has been solved by resorting to a clustering approach, where three policies have been defined. While in the first, all the VUs requesting the offloading service perform also the FL process, aiming at optimizing them jointly, in the second, a probabilistic clustering policy considers that only a subset of the VUs participates to the optimization process, modeling the possibility that some of the VUs are not capable of performing the FL. As third policy, we still select a subset of VUs while based on their position within the RSU coverage area. The clustering policies are also compared with a distributed policy where each VU acts independently. Further, a Genetic Algorithm (GA) from the family of evolutionary computing methods is considered for solving the previously introduced policies.

The main contributions of this paper can be summarized in:

- We define an air-ground integrated FL-inspired distributed learning platform for enabling a run-time evaluation of the

computation offloading parameters. We consider the HAPs as FL servers and VUs as distributed FL devices/clients, while the RSUs act as processing devices accepting computation offloaded by the VUs acting as sources.
- We model the joint learning and offloading process through its delay and energy consumption, and then we define the joint delay and energy minimization problem as a constrained non-linear optimization problem. The main aim is to select the optimal number of FL process iterations for each VU given a target delay requirement, while keeping the energy consumption under a certain level.
- Three RSU based clustering approaches are introduced for solving the problem (i.e., full clustering, probabilistic clustering, distance-based clustering) along with a VU-based distributed approach.
- A GA evolutionary computing method is proposed for solving clustered and distributed approaches. Two other benchmark methods and one simple Heuristic approach based on a reduced size solution space are also considered for performance comparison.
- Performance evaluation is carried out under different VEC environments where the effectiveness of the proposed scheme is shown.

The remaining parts of this paper are organized as follows. In Section II, the main related works in the area are discussed, while Section III presents the system model and define the optimization problem to be solved. In Section IV, the clustering policies and the GA-based solution method are discussed. In Section V, the numerical results obtained through computer simulations are provided and analyzed. Finally, in Section VI the conclusions are drawn.

II. Related Works

The importance of VEC in the VN scenarios has been highlighted in several survey papers; to this aim [19], [20] constitute two outstanding starting points for understanding the working scenario and main challenges. Among several challenges, the partial computation offloading problem in the VEC-enabled VNs for the latency-critical applications is considered by several authors. In [3], a joint load balancing and offloading problem is formulated as a utility maximization problem by considering the latency constraints and solved through a low complexity algorithm. The computation offloading problem in heterogenous VEC scenarios is studied in [5], where multi-armed bandit theory is applied, and online and off-policy learning algorithms are proposed for the network selection problem. Some attempts have also been made for optimizing the energy cost while performing computation offloading operations in VN. In [21], a low-complexity heuristic method is proposed based on the cost-effectiveness of allocated resources and energy consumption for computation offloading in VEC scenarios. In [22], authors have studied the energy-efficient workload offloading problem in VEC and propose a low-complexity distributed solution based on consensus alternating direction method of multipliers. Many authors have treated the latency and energy cost minimization problems separately even though it is proven that there is a clear trade-off between latency and energy consumed during computation offloading towards edge servers [23]. Only a few attempts have been made for joint minimization of energy and latency in vehicular scenarios. One such approach can be found in [24] where the energy-efficient dynamic computation offloading and resources allocation scheme for minimizing the joint energy and latency cost in a Vehicular Fog Computing scenario is proposed.

In the recent past, LAPs and HAPs usage as edge computing nodes in vehicular scenarios has been proposed by many researchers for improving the overall system reliability, service latency, and energy performance. In [25], authors have introduced a three-layer network architecture by integrating the HAP and terrestrial edge network for improving the delay performance of vehicular nodes through computation offloading and caching over HAPs. An edge computing enabled integrated space-air-ground network platform is proposed and analyzed in [26] for providing vehicular services into remote areas with lower latency and the reduced uses of satellite resources. In [27], a new energy-efficient, UAV-assisted edge computing framework allowing a joint optimization of the trajectory and CPU frequency of fixed-wing UAVs along with offloading scheduling is proposed. In [28], authors have proposed a UAV-assisted VEC system architecture for enabling 6 G vehicle-to-everything (V2X) applications. This work also highlights the main challenges in the UAV-assisted VEC systems including its use for achieving distributed intelligence at the edge, for implementing vehicular applications and services.

In recent times, several works have highlighted the importance of the FL process in VEC-enabled scenarios. In [29], the authors listed several applications, research challenges, and future directions for the FL research in VNs. A mobility-aware FL scheme for edge caching in VN is proposed in [30], which can protect users' privacy, reduce communication costs, and support the high mobility of vehicles. In [31], authors have provided a brief survey of applications and challenges while using the FL process in vehicular scenarios.

While implementing FL over resource-constrained networks, it is important to consider a trade-off between available resources and the FL performance. This issue has become clear since few years when researchers started to analyze the FL process optimization problem in terms of computing and communication resource allocation, user scheduling, energy performance, latency performance. Joint energy-efficient transmission and computing allocation for the FL process over wireless communication networks has been investigated in [18]. In [32], the authors have addressed the joint resource allocation and user selection problem for FL process performance improvement. In [33], joint user association, service sequence, and task allocation problem for minimizing the weighted sum of the energy and time consumption over a MEC-enabled balloon HAP network is addressed. In [34], FL device scheduling policies, taking into account the channel conditions and the significance of the local model updates, are provided for better network performance.

Lately, similar analyses have been carried out in the VN scenarios, with the aim of improving the FL process accuracy with a limited cost. In [35], the authors have studied FL in a VEC scenario and have proposed an approach for selecting the

best quality models during the training phase for tackling the diverse data quality and corresponding information asymmetry issue of the FL process. An edge computing-based joint client selection and networking scheme for vehicular IoT is presented in [36]. The importance of the trade-off between the accuracy of the global model and the communication overhead of FL in vehicular environments is also highlighted.

In several of these works, authors aimed at optimizing the FL process training phase without paying much attention to the application latency requirements. Also, the mobility scenarios in VNs can have a significant impact on the FL process and are required to be explored. Therefore we aim to perform a joint optimization of FL and computation offloading processes in VEC scenarios by considering the latency constraints while keeping the energy consumption reduced.

III. System Model and Problem Formulation

We consider an integrated air-ground network composed of one HAP, a set $\mathcal{V} = \{v_1, \ldots, v_m, \ldots, v_M\}$ of M VUs, and a set $\mathcal{R} = \{r_1, \ldots, r_n, \ldots, r_N\}$ of N RSUs, placed in the area, supposed to be modeled as a two-lane road scenario.

The generic mth VU, supposed to move in either of the two directions, is characterized by a processing capability equal to $c_{v,m}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$. Each VU is supposed to be able to communicate on a bandwidth $B^{\text{rsu}}_{v,m}$ during a terrestrial communication. On the other hand, while communicating with the HAP in the non-terrestrial communication network, it is supposed to communicate with a bandwidth $B^{\text{HAP}}_{v,m}$. Many new vehicular applications and services, including autonomous driving, online gaming, multimedia content streaming infotainment services, etc., come with strict application latency requirements. Such latency constraints need to be taken into account while solving vehicular networks problems [22], [37]. Therefore, in this work, the mth VU is supposed to generate tasks to be processed, where the task x_m is identified through the tuple $\langle D_{x_m}, \Omega_{x_m}, \bar{T}_{x_m} \rangle$ where D_{x_m} is the task size in Byte, Ω_{x_m} are the requested CPU execution cycles and $\bar{T}_{x_m}$ is the maximum latency of the requested service.

The nth RSU, supposed to be in a fixed position, is characterized by a processing capability equal to $c_{r,n}$ FLOPS per CPU cycle, with CPU frequency $f_{r,n}$, and communication capabilities, supposed to be identified through a communication technology able to work on a bandwidth $B_{r,n}$ and covering an area with radius $R_{r,n}$. Each RSU provides computation offloading services to the VUs within its coverage area. In addition, the area is supposed to be under the coverage of one HAP equipped with an edge computing server with much superior computation capabilities compared with the RSU/VUs [25]. Moreover, we consider multi-beam antenna forming techniques, placed at an altitude of h_{HAP} above the ground, where each antenna beam is supposed to cover a geographical area of radius R_{HAP} and having a communication bandwidth B_{HAP}. In the following we will refer to a single beam as the coverage of the HAP. It should be noted that though HAP coverage is reduced to a single beam for notation simplicity, our approach can easily be scaled for the overall HAP coverage with multiple beams.

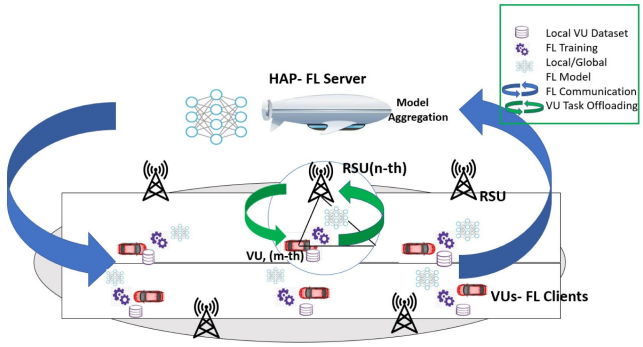

Fig. 1. System Architecture.

Fig. 1 highlights the main system elements of considered network architecture. It is worth to be noted that we have considered only one centralized HAP in the following. In a realistic scenario, for having a better fault tolerance, it can be complemented by ground-based 5 G base stations (5G-gNB) or replaced by a decentralized HAP network, composed by multiple HAPs.

A. Vehicular Mobility Model

The generic mth VU is supposed to be located at position $\{x_{v,m}(t), y_{v,m}(t)\}$ at time t. Each vehicle is supposed to move along the x-axis, defining the directions of the two-lane road, with a speed $\bar{v}_m$, supposed to be constant. The nth RSU is considered to be in a fixed position $\{x_{r,n}, y_{r,n}\}$. Hence, it is possible to define the remaining distance within which the mth VU remains under the coverage of the nth RSU as,

$$\Pi_{m,n} = \sqrt{R^2_{r,n} - (y_{r,n} - y_{v,m})^2} \pm (x_{r,n} - x_{v,m}) \qquad (1)$$

where $R_{r,n}$ is the coverage radius of the nth RSU and $\pm$ identifies the two possible directions taken by the mth VU. It is worth to be noticed that the dependency on t in (1) has been omitted for a better clarity; moreover, we assume that it is calculated at a time instant t when the mth VU requests an offloading service. In addition, it is worth to be noticed that (1) is valid only if the mth VU is within the nth RSU coverage area.

The time available by the mth VU before leaving the nth RSU coverage, i.e., *sojourn time*, is defined as [6]:

$$T^{\text{soj}}_{m,n} = \frac{\Pi_{m,n}}{\bar{v}_m} \qquad (2)$$

Similarly, supposing that the center beam of the HAP coverage on the ground is in a fixed position $\{x_{\text{HAP}}, y_{\text{HAP}}\}$, it is possible to define the remaining distance within which the mth VU remains under the HAP beam coverage as:

$$\Pi_{m,\text{HAP}} = \sqrt{R^2_{\text{HAP}} - (y_{\text{HAP}} - y_{v,m})^2} \pm (x_{\text{HAP}} - x_{v,m}) \qquad (3)$$

which is valid only if the mth VU is within the HAP coverage area as well. Hence, the HAP sojourn time can be defined as:

$$T^{\text{soj}}_{m,\text{HAP}} = \frac{\Pi_{m,\text{HAP}}}{\bar{v}_m}. \qquad (4)$$

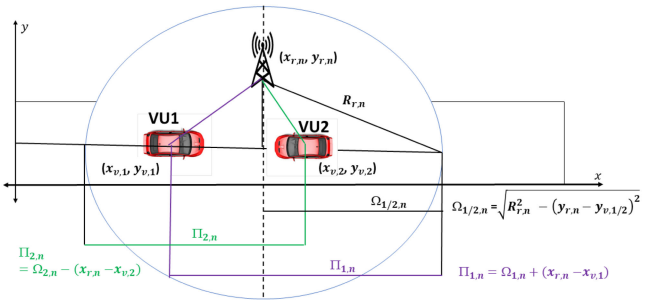

Fig. 2. VU Mobility Scenarios and Corresponding Distance Matrices.

In Fig. 2, the working scenario is depicted, where we suppose, as an example, the presence of two VUs (i.e., VU1 and VU2) moving in opposite directions, where VU1 is traveling towards right and VU2 moving towards left. The available distance for each of the VUs before passing through the coverage area of RSU is determined by using (1). A similar analysis can be performed for determining the distances concerning the HAP.

B. Partial Offloading Model

The VUs are supposed to be able to offload their tasks to the RSUs within their connecting area. In order to minimize the time spent for the offloading process we assume that the mth VU is able to split its task in two portions and offload a portion $\alpha_m \in [0,1]$ to any of the RSUs within its connecting area,[1] while the remaining $(1-\alpha_m)$ can be locally computed [23].

1) Task Offloading Process: When mth VU selects the nth RSU for offloading its task, the process is composed of the data transmission toward the selected RSU, task processing at the RSU, and the reception of computed data back at the VU. Each of these steps consume some amount of time and energy, as detailed in the following.

In the following, we resort to the Shannon capacity formula for evaluating the data rate between any node i and j as a function of the distance between them, defined as:

$$r_{i,j}(B_i, d_{i,j}) = B_i \log_2\left(1 + \frac{P_i^{\text{tx}} \cdot h(d_{i,j})}{N_0}\right) \tag{5}$$

where P_i^{tx} is the transmission power of the generic device i, $h(d_{i,j})$ is the channel gain at a distance $d_{i,j}$ between the device i and the device j, and $N_0 = N_T B_i$ is the noise power, where N_T and B_i are the noise power spectral density and bandwidth associated to the ith device during communication.

a) VU-RSU Communication: The total time and energy required for full offloading of task x_m from the mth VU towards the nth RSU is given by,

$$T_{m,n}^{x_m,\text{tx}} = \frac{D_{x_m}}{r_{m,n}(B_{v,m}^{\text{rsu}}, d_{m,n})}, \quad E_{m,n}^{x_m,\text{tx}} = P_m^{\text{tx}} \cdot T_{m,n}^{x_m,\text{tx}}, \tag{6}$$

where, $r_{m,n}(B_{v,m}^{\text{rsu}}, d_{m,n})$ is the data rate between mth VU and nth RSU, that depends on the available radio resources i.e., $B_{v,m}^{\text{rsu}}$, and the distance between two devices, i.e., $d_{m,n}$. Also, P_m^{tx} is the transmission power of m-th VU while transmitting data towards the RSU.

b) RSU Computation: The task computation at the RSU side depends on the CPU execution cycles requested by the task, i.e., Ω_{x_m} and available RSU processing resources, i.e., $c_{r,n}$ and $f_{r,n}$; hence, the processing time can be modeled as:

$$T_n^{x_m,\text{c}} = \frac{\Omega_{x_m}}{c_{r,n} f_{r,n}}. \tag{7}$$

Here, we assume that the RSUs are connected to the electrical grid, hence their energy cost is negligible, while the VUs are in idle state, whose energy consumption can be neglected as it can be considered an unavoidable basic energy consumption.

c) RSU-VU Communication: The completion of the task offloading process is performed by sending back the result to the VU. The time and energy required by the mth VU for receiving the result from nth RSU is given by,

$$T_{m,n}^{x_m,\text{rx}} = \frac{D_{x_m,\text{rx}}}{r_{n,m}(B_{r,n}, d_{n,m})}, \quad E_{m,n}^{x_m,\text{rx}} = P_m^{\text{rx}} \cdot T_{m,n}^{x_m,\text{rx}}, \tag{8}$$

where, $D_{x_m,\text{rx}}$ is the task size processing result at the RSU side, and $r_{n,m}(B_{r,n}, d_{n,m})$ is the downlink data rate between the nth RSU and the mth VU. P_m^{rx} is the reception power of m-th VU while receiving data from the RSU.

In general, RSUs are located in the proximity of VUs, resulting in negligible task propagation time during uplink and downlink communication. Therefore, in this work, we do not consider the propagation time when modeling the delay of the task offloading process. Thus, the total time and energy required for the complete task offloading process is,

$$\hat{T}_{m,n}^{\text{off}} = T_{m,n}^{x_m,\text{tx}} + T_n^{x_m,\text{c}} + T_{m,n}^{x_m,\text{rx}} \tag{9}$$

$$\hat{E}_{m,n}^{\text{off}} = E_{m,n}^{x_m,\text{tx}} + E_{m,n}^{x_m,\text{rx}} \tag{10}$$

Since we assume that the mth VU offloads a portion α_m of the task x_m towards the nth RSU, the overall time required to perform the offloading process is:

$$T_{m,n}^{\text{off}}(\alpha_m) = \alpha_m \cdot \hat{T}_{m,n}^{\text{off}} \tag{11}$$

where we suppose that both communication and processing latency terms scale linearly. Similarly, the overall energy consumed by the mth VU for performing the offloading process is:

$$E_{m,n}^{\text{off}}(\alpha_m) = \alpha_m \cdot \hat{E}_{m,n}^{\text{off}} \tag{12}$$

2) Local VU Computation Process: Each VU is able to locally compute its task and the amount of time and energy required is based on its processing resources, i.e., c_m and f_m. Hence, the local processing time and energy consumption is:

$$T_m^{x_m,\text{c}} = \frac{\Omega_{x_m}}{c_{v,m} f_{v,m}}, \quad E_m^{x_m,\text{c}} = P_m^{\text{c}} \cdot T_m^{x_m,\text{c}}, \tag{13}$$

where, P_m^{c} is the computational power used during local task computation at the mth VU. Due to the partial offloading, the amount of time and energy required for the local computation at the mth VU is:

$$T_m^{\text{loc}}(\alpha_m) = (1-\alpha_m) T_m^{x_m,\text{c}} \tag{14}$$

[1]More specifically to be managed by the co-located server.

$$E_m^{\text{loc}}(\alpha_m) = (1-\alpha_m)E_m^{x_m,\text{c}} \tag{15}$$

where α_m is the portion of the task to be offloaded by the mth VU; hence, the overall processing time for the task x_m results:

$$T_m^{x_m}(\alpha_m) = \max\left\{T_{m,n}^{\text{off}}(\alpha_m), T_m^{\text{loc}}(\alpha_m)\right\} \tag{16}$$

where $T_{m,n}^{\text{off}}(\alpha_m)$ is the time needed for offloading the portion of a task $\alpha_m \cdot x_m$ to the nth RSU, while $T_m^{\text{loc}}(\alpha_m)$ is the time for locally processing the remaining task $(1-\alpha_m)\cdot x_m$ by the mth VU. We suppose that offloading and local computation can be performed in parallel. Similarly, the overall processing energy for the task x_m results:

$$E_m^{x_m}(\alpha_m) = E_{m,n}^{\text{off}}(\alpha_m) + E_m^{\text{loc}}(\alpha_m) \tag{17}$$

where $E_{m,n}^{\text{off}}(\alpha_m)$ is the energy consumed offloading the portion of a task $\alpha_m \cdot x_m$ to the nth RSU, while $E_m^{\text{loc}}(\alpha_m)$ is the energy consumed during locally processing the remaining task $(1-\alpha_m)\cdot x_m$ on mth VU.

3) Partial Offloading Problem: The partial computation offloading problem corresponds to set the offloading parameters α_m in an optimal way such that service latency ($\bar{T}_{x_m}$), sojourn time ($T_{m,n}^{\text{soj}}$) and the overall energy consumption constraints are respected. To this aim, we assume that in an energy efficient partial offloading operation, the energy spent for the task x_m ($E_m^{x_m}(\alpha_m)$) with offloading parameter α_m, is less than the amount of energy required to completely compute it locally (i.e., $E_m^{x_m,\text{c}}$), since otherwise offloading would not be beneficial. Therefore, the joint latency and energy constrained optimization problem corresponds to find the optimal $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m, \ldots, \alpha_M\}$ parameters such that:

$$\mathbf{P1}: \ \mathcal{A}^* = \text{argmin}_{\mathcal{A}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left(\eta_1 T_m^{x_m}(\alpha_m) + \eta_2 E_m^{x_m}(\alpha_m)\right) \right\} \tag{18}$$

subject to the following constraints,

$$T_m^{x_m}(\alpha_m) \le \bar{T}_{x_m}, \qquad \forall m \tag{19a}$$

$$T_{m,n}^{\text{off}}(\alpha_m) \le T_{m,n}^{\text{soj}}, \qquad \forall m, \forall n \tag{19b}$$

$$E_m^{x_m}(\alpha_m) \le E_m^{x_m,\text{c}}, \qquad \forall m \tag{19c}$$

$$\sum_{n=1}^{N} a(m,n) \le 1, \qquad \forall m \in M \tag{19d}$$

$$\sum_{m=1}^{M} a(m,n) \cdot B_{v,m}^{\text{rsu}} \le B_{r,n}, \qquad \forall n \in N \tag{19e}$$

$$0 \le \alpha_m \le 1, \qquad \forall m \in M \tag{19f}$$

$$0 \le \eta_1, \eta_2 \le 1, \tag{19g}$$

where (19a) shows that the total task processing time for each VU should be limited by the task latency requirement and (19b) represents that each VU should complete the computation offloading process while it is in the RSU coverage for avoiding additional latency costs. From (19c), the overall processing energy of task should be upper bounded by the amount of energy required to compute it locally. A binary assignment variable $a(m,n)$ is considered equal to 1 if mth VU is assigned to the nth RSU, and 0 otherwise. According to (19d), each VU can offload tasks to no more than one RSU, while (19e) shows that the bandwidth resources available for all active VUs[2] in a particular RSU coverage is upper bounded by its bandwidth. Eq (19f) limits the offloading parameter value between 0 and 1. Moreover, in (19g), η_1 and η_2 are two weight coefficients between 0 and 1, for balancing latency and energy consumption.

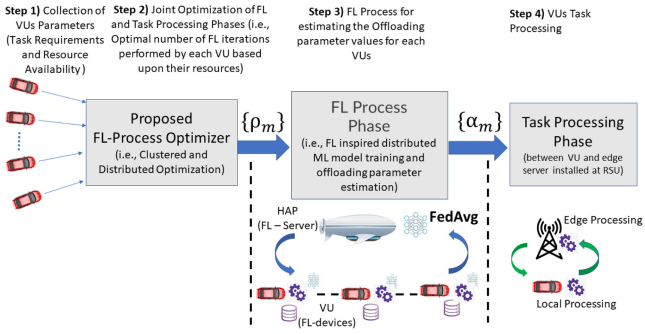

Fig. 3. Proposed scheme for the joint FL and task-offloading processes optimization.

In a real scenario the amount of data to be offloaded from each VU towards an RSU while respecting the system constraints is hard to be estimated; several factors, including VUs position, velocity, directions, RSU resources, task requirements, surrounding environmental conditions, make the problem hard to be solved. Many of these parameters are hard to be accessed given their stochastic behaviors. Therefore, finding a set of optimal offloading parameters ($\mathcal{A}^*$) in a highly dynamic environment like VN is a challenging problem to be solved, and advanced optimization methods are needed. Belonging to the class of the ML approaches, FL has been recently introduced as an effective way for performing data augmentation and significantly reducing the communication overhead in comparison with direct data-sample exchanges, allowing also to enhance VUs privacy issues. In order to properly address the latency and energy constrained offloading problem defined in (18), we propose to exploit a FL framework for estimating the set $\mathcal{A}$, composing the offloading portions of all VUs, based on the VU side parameters.

In Fig. 3 we provide a more detailed step-by-step view of the considered joint FL and task offloading process optimization problem and the proposed solution methodologies; it is possible to notice that the VUs parameters act as input for the FL-inspired distributed process (Step 1), whose goal is to properly set the number of iterations to be performed (Step 2) in order to have a proper solution for setting the offloading parameters (Step 3) to be later used by each VU (Step 4).

[2]We assume that only a subset of the VUs, named active, have data to process, and potentially to be offloaded to an RSU.

C. Federated Learning Model

FL is based on the idea that the same ML algorithm is present at both FL server and FL clients sides, where a centrally located FL server assists distributed clients during the learning process. Instead of only executing the ML algorithm in a centralized server node, it is executed in a federated way among all the involved nodes through the exchange of a set of parameters defining the weights of the implemented ML algorithm. To do this, the FL process is composed of several steps: information exchange between FL-server and devices for initializing the learning model over devices, local device training, parameters exchange over wireless links between devices and the FL-server, parameter collection and aggregation on the server [38]. In the FL process, we assume that the HAP acts as FL server, assisting the VUs acting as FL clients for making the offloading decision. For each offloading request, the VUs perform numerous FL iterations with the HAP aiming at properly setting the offloading portion toward the selected RSU. It has to noticed that the HAP computation infrastructure can be implemented by resorting to the function virtualization approach through different virtualization technologies, e.g., virtual machines, containers, hypervisors, for performing the FL process. Moreover, the interaction with FL-clients can happen through predefined interfaces (e.g., implementing the REST API technology) allowing a smarter interaction [39]. However, such considerations are beyond the scope of this work, that instead mainly focuses on the optimization of the joint FL-offloading framework.

Even though FL allows to reach a global optimum in distributed environments, the dynamicity of VN scenarios introduces an additional challenge. Indeed, FL process cannot be considered as a granted process, as it consumes resources by itself. Hence, FL is executed at the cost of a reduction of resources that can be given to the offloading process. It is however, clear from past studies that the number of FL iterations required for reaching a predefined convergence value can be upper bounded [18], [38], [40], [41] depending on several factors, including the ML model, number of users participating in the training process, number of local iteration on the device, type of radio environment, quality of data, etc. Therefore, without loss of generality, we consider that after ρ^{opt} FL iterations each VU will be able to estimate the optimal offloading parameter α_m^{opt}, where $\rho^{\text{opt}} = \frac{\mathcal{K}}{\bar{M}}$; $\mathcal{K}$ can be considered as a numerical constant setting the overall number of FL iterations required to achieve the convergence, while $\bar{M}$ is the number of VUs participating in the FL training process, so higher the participating VUs, lower the required iterations, respecting the FL process behavior.

In this work, we assume the learning process converges after ρ^{opt} FL iterations, when each VU is able to estimate the offloading parameters[3]. For the purpose of this paper, we consider that, in case we stop the FL process in advance, some estimation error should be considered, as later explained. Since each FL iteration requires a certain amount of communication and computational resources, performing ρ^{opt} iterations over all VUs can be challenging and sometimes might not be feasible given the limited VUs resources and the latency constraints imposed by both service requirements and sojourn time. The additional energy cost of each FL iteration can also limit the number of FL iterations performed by VUs. Therefore, in this work we consider that the generic m-th VU is able to perform up to ρ_m FL iterations with $\rho_m \leq \rho^{\text{opt}}$. The set $\mathcal{I} = \{\rho_1, \ldots, \rho_m, \cdots \rho_M\}$ contains the number of FL iterations performed by each VU.

In order to understand the impact of the FL process we can now introduce the FL iterations latency, the corresponding energy consumption and the joint optimization model.

1) FL Computation Model: The FL computation corresponds to the local training of the ML model based on the on-device dataset. In local device training, the mth VU has to compute the local parameter set $w_{v,m}^{it}$ through the dataset having size K_m data samples; if we assume that, for every iteration, the total number of FLOPs required for each data sample d is ψ_d, the time and energy consumed during FL process at the mth device is given by [42]:

$$T_m^{\text{FL,c}} = \frac{\sum_{d=1}^{K_m} \psi_d}{c_{v,m} f_{v,m}}, \quad E_m^{\text{FL,c}} = P_m^{\text{c}} \cdot T_m^{\text{FL,c}}. \tag{20}$$

We suppose for simplicity that the on-device FL processing time and energy is the same for every iteration. Conversely, the FL server is limited to the model aggregation, whose time and energy is considered as negligible given the abundant available resources at HAP.

2) FL Communication Model: In FL, the devices communicate the local model updates towards the HAP in uplink and receive back the updated global model parameters in downlink. Both uplink and downlink communication processes are characterized by transmission and propagation delays, due to the high distance between VUs and HAP. The propagation time required for each FL iteration is given by,

$$T_{m,it}^{\text{FL,prop}} = 2 \cdot \frac{d_{m,\text{HAP}}}{\sigma}, \quad \forall m \tag{21}$$

where σ is the propagation speed in the considered transmission medium, $d_{m,\text{HAP}}$ is the distance between the mth VU and the HAP, which can be calculated by using HAP altitude (h_{HAP}) and the mth VU location through simple algebraic passages, and the multiplication by 2 is due to the two-way propagation delay. During the FL processing, at each iteration it the mth VU sends the parameters set $w_{v,m}^{it}$ to the HAP. Supposing that $|w_m^{it}|$ represents the data size of the parameters set expressed in bits [32], the uplink transmission time and energy for the FL parameters in the itth iteration is:

$$T_{m,it}^{\text{FL,tx}} = \frac{|w_{v,m}^{it}|}{r_{m,\text{HAP}}^{it}(B_{v,m}^{\text{HAP}}, d_{m,\text{HAP}})}, \quad E_{m,it}^{\text{FL,tx}} = P_m^{\text{tx}} \cdot T_{m,it}^{\text{FL,tx}}, \tag{22}$$

where, $r_{m,\text{HAP}}^{it}$ is the uplink transmission rate between mth VU and the HAP during the itth iteration, which is a function of the VUs bandwidth ($B_{v,m}^{\text{HAP}}$), and the distance ($d_{m,\text{HAP}}$) between the mth VU and the HAP, modeled through the Shannon capacity formula under Rice fading conditions [43]. Since the HAP is accessed by multiple VUs, we assume for simplicity that the HAP bandwidth is equally shared among the connected VUs.

[3]In the case of a practical system, the convergence can be bounded by some stopping criteria, e.g., loss function value.

Also, P_m^{tx} is the VUs, transmission power while communicating with HAP.

In general, HAP needs to wait for all training VUs to transmit their model parameters before performing the averaging operation. Therefore, the FL transmission time for the itth iteration is given by,

$$T_{it}^{\text{FL,tx}} = \max_{m} \left\{ T_{m,it}^{\text{FL,tx}} \right\}, \quad \forall m \tag{23}$$

The HAP performs the aggregation of the received model parameters (e.g., FedAvg [38]) to create a global parameter vector w_G^{it} for the next iteration and transmits it back towards VUs over the downlink communication links. Therefore, in downlink, the global parameters transmission time and energy are given by,

$$T_{m,it}^{\text{FL,rx}} = \frac{|w_G^{it}|}{r_{\text{HAP},m}^{it}(B_{\text{HAP}}, d_{\text{HAP},m})}, \; E_{m,it}^{\text{FL,rx}} = P_m^{\text{rx}} \cdot T_{m,it}^{\text{FL,rx}} \tag{24}$$

where $r_{\text{HAP},m}^{it}$ is the downlink transmission rate between the HAP and the mth VU during the itth iteration when the global parameter set is broadcast. P_m^{rx} is the power consumed while receiving data from HAP. Hence, the total time and energy required for a single FL iteration can be detailed as:

$$T_{m,it}^{\text{FL}} = T_m^{\text{FL,c}} + T_{m,it}^{\text{FL,prop}} + T_{it}^{\text{FL,tx}} + T_{m,it}^{\text{FL,rx}} \tag{25}$$

$$E_{m,it}^{\text{FL}} = E_m^{\text{FL,c}} + E_{m,it}^{\text{FL,tx}} + E_{m,it}^{\text{FL,rx}} \tag{26}$$

D. Joint Offloading and Federated Learning Model

Since the FL process is based on multiple iterations for exchanging the ML model parameters, it is possible to write the total time and energy for the FL process when focusing on the mth VU as,

$$T_m^{\text{FL}}(\rho_m) = \sum_{it=1}^{\rho_m} T_{m,it}^{\text{FL}}, \qquad E_m^{\text{FL}}(\rho_m) = \sum_{it=1}^{\rho_m} E_{m,it}^{\text{FL}} \tag{27}$$

where ρ_m is the number of FL iterations performed by mth VU, $T_{m,it}^{\text{FL}}$ is the amount of time spent, and $E_{m,it}^{\text{FL}}$ is the amount of energy consumed for the itth iteration of the FL process depending on both FL communication and computation performance. The time needed for completing both FL iterations and task processing has to be constrained by the maximum service latency requirement, given by:

$$T_m(\rho_m, \alpha_m) = T_m^{\text{FL}}(\rho_m) + T_m^{x_m}(\alpha_m) \leq \bar{T}_{x_m} \tag{28}$$

Also the energy consumed for completing both FL iterations and task processing has to be constrained by the energy required to compute a complete task locally, given by:

$$E_m(\rho_m, \alpha_m) = E_m^{\text{FL}}(\rho_m) + E_m^{x_m}(\alpha_m) \leq E_m^{x_m,\text{c}} \tag{29}$$

Due to the dynamicity of the vehicular environment, computation offloading and FL process latencies should also be bounded by the VUs sojourn times under RSU and HAP beam coverage. Since the HAP is acting as an FL server, the whole FL phase should be completed by the HAP sojourn time, hence:

$$T_m^{\text{FL}}(\rho_m) \leq T_{m,\text{HAP}}^{\text{soj}} \tag{30}$$

In addition, each VU should finish the offloading process within the RSU sojourn time. Thus,

$$T_m^{\text{FL}}(\rho_m) + T_{m,n}^{\text{off}}(\alpha_m) \leq T_{m,n}^{\text{soj}} \tag{31}$$

It is worth to be noticed that the sojourn time does not affect the overall processing time, while only the offloading time, since the local computation can be performed also out of the RSU coverage.

1) Problem Formulation: Following (11), (14), (16), (25), (27), and (28) the total time $T_m(\rho_m, \alpha_m)$ required for both phases (i.e., FL and task processing) can be determined. Similarly, from (12), (15), (17), (26), (27), and (29) the total energy $E_m(\rho_m, \alpha_m)$ required for both phases can be calculated. The proposed optimization model aims at minimizing the total time and energy by properly setting the offloading parameters and the FL iterations used for determining the offloading parameters itself. Hence, the problem in (18) can be rewritten as:

$$\mathbf{P2}: (\mathcal{I}^*, \mathcal{A}^*) = \text{argmin}_{\mathcal{I},\mathcal{A}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left(\eta_1 \cdot T_m\left(\rho_m, \alpha_m\right) + \eta_2 \cdot E_m\left(\rho_m, \alpha_m\right) \right) \right\} \tag{32}$$

subject to the constraints (19d)-(19g), (28)-(31), and,

$$\sum_{m=1}^{M} B_{v,m}^{\text{HAP}} \leq B_{\text{HAP}} \tag{33a}$$

$$0 \leq \rho_m \leq \rho^{\text{opt}} \qquad \forall v_m \in \mathcal{V} \tag{33b}$$

where (28) is the service latency requirement reformulating (19a) including the FL processing time. Also, (29) is the reformulated energy constraint defined in (19c) with FL process energy. Eq (30) provides an upper bound for the FL process depending on the HAP sojourn time and (31) is the reformulated version of (19b), defining the upper bound of both task offloading and FL process as the RSU sojourn time: each vehicle should offload the computation data to the RSU and receive results before it leaves its coverage area. According to (33a), the sum of bandwidth resources available for all VUs in non-terrestrial communication links should be upper bounded by the HAP bandwidth resources. Eq (33b) upper bounds the number of iterations performed by each VU to ρ^{opt}.

2) Federated Offloading Parameter Estimation: Solving the problem defined in (32) requires finding two sets of optimization variables $(\mathcal{I}, \mathcal{A})$ and thus is hard to be solved. However, $(\mathcal{I}, \mathcal{A})$ are not two separate sets of variable. As more iterations are performed, higher is the reliability with which the offloading parameter is estimated through the FL process. Hence, the offloading parameter α_m can be modeled as function of the number of FL iterations performed with the aim of estimating the optimal α_m^{opt}, i.e., $\alpha_m = \alpha_m(\rho_m)$. Without loss of generality, we assume in the following that in case the m-th VU cannot participate in the FL process, the offloading parameter is $\alpha_m^0 = \alpha_m(\rho_m = 0)$, while in case it can perform ρ^{opt} iterations, the estimated offloading parameter is $\alpha_m^{\text{opt}} = \alpha_m(\rho^{\text{opt}})$. In any other case, the estimated value α_m is a function of ρ_m FL iterations that are

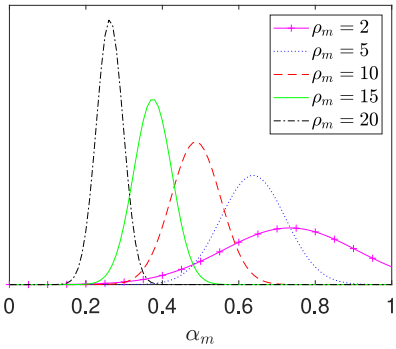

Fig. 4. Truncated Normal Distribution of α_m as a function of the FL iterations.

performed by the mth VU. The exact relationship between α_m and ρ_m is hard to be set since it depends on several factors such as FL environment, number of VUs participating in the FL process, the communication medium between FL clients and server, etc. To the best of our knowledge there is no model in the literature aiming at setting the aforementioned relationship. Therefore, without loss of generality, we consider here that the estimated α_m can be modeled as a stochastic value whose distribution follows a *truncated normal distribution* with mean μ and variance σ^2, where $0 \leq \alpha_m \leq 1$, since α_m is bounded between 0 and 1 by definition. Therefore, it is possible to define the probability density function $f_{\alpha_m}(\cdot)$ of α_m as,

$$f_{\alpha_m}(\alpha_m;\mu,\sigma) = \begin{cases} \frac{1}{\sigma}\frac{\xi\left(\frac{\alpha_m-\mu}{\sigma}\right)}{\Delta\left(\frac{1-\mu}{\sigma}\right)-\Delta\left(\frac{-\mu}{\sigma}\right)} & \text{if } 0 \leq \alpha_m \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

where, $\xi(\cdot)$ and $\Delta(\cdot)$ are, respectively, the probability density function of the related standard normal distribution and its cumulative distribution function, i.e.,

$$\xi(\omega) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{\omega^2}{2}\right)}, \quad \Delta(\kappa) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{\kappa}{\sqrt{2}}\right)\right).$$

In this work we assume that the mean value of the distribution of α_m, i.e., μ, and its variance, σ^2, are equal to

$$\mu = \alpha_m^{\text{opt}}(\rho_m), \quad \sigma^2 = \left(\gamma \cdot \frac{\rho^{\text{opt}} - \rho_m}{\rho^{\text{opt}}}\right)^2 \tag{35}$$

where γ is a numerical constant, used for controlling the variance of the model. It is worth to be noticed that the variance is defined in a way that higher ρ_m, lower is the variance. This corresponds to say that increasing the number of iterations reflects in a more reliable estimation of α_m provided that $\rho_m \leq \rho^{\text{opt}}$. Moreover, the higher the iterations to be performed, the higher is the time spent in the FL phase, so the lower is the time left for the offloading phase. This is the reason why μ is also function of the iterations. This is consistent with the FL process where more FL iterations turn out in a better estimation of the offloading parameter. During simulations, $f_{\alpha_m}(\alpha_m;\mu,\sigma)$ is used for estimating the α_m for every mth VU, whose quality will depend upon the number of FL iterations performed compared with ρ^{opt}. A qualitative representation is reported in Fig. 4 with $\rho^{\text{opt}} = 25$, where as the number of iterations increases, both the distribution

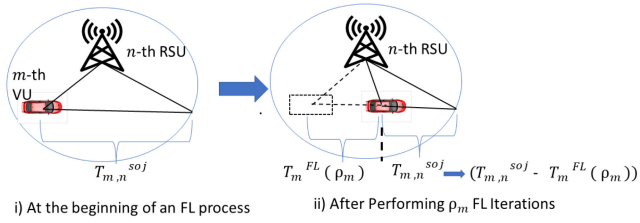

Fig. 5. FL process impact over the offloading parameter value.

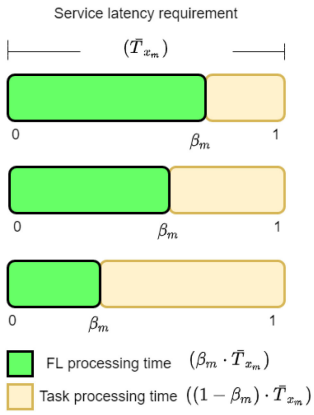

Fig. 6. FL and Task processing time sharing.

variance and the average optimal offloading parameter become smaller, leaving less time for the offloading operation.

According to (31) both FL and task offloading processes should be completed within available sojourn time. Fig. 5 shows the impact of the constraint (31) on the considered vehicular environment. In particular, we can notice that at the beginning, the joint FL and offloading process is bounded by the sojourn time. As the VU moves, despite some iterations that are performed, the remaining time for completing the FL process and starting the offloading is reduced, due to the lower remaining sojourn time.

From the previous description, it is clear that given a certain amount of time, we have to trade-off between offloading and FL processes. Let us introduce now a new parameter, named $\beta_m \in [0,1]$, modeling the portion of time allocated for the FL process of the mth VU. If $\beta_m = 0$, the whole time is allocated for the task processing phase, while if $\beta_m = 1$ the mth VU uses the whole available time for the FL phase. Considering the target latency of the tasks generated by each VU as a reference time interval, it is possible to set the maximum number of possible iterations for the FL process:

$$\rho_m(\beta_m) \text{ s.t. } T_m^{\text{FL}}(\beta_m) = \sum_{it=1}^{\rho_m(\beta_m)} T_{m,it}^{\text{FL}} \leq \beta_m \cdot \bar{T}_{x_m} \tag{36}$$

where $\mathcal{B} = \{\beta_1, \ldots, \beta_m, \ldots, \beta_M\}$. Each VU performs numerous FL iterations aiming at finding the optimal offloading amount to be transferred towards RSU, where any additional FL iteration reduce the variance in (34), i.e., its reliability, while reducing its average value.

Fig. 6 shows the available resources for both phases as a function of β_m. As β_m increases VU spends more time on

the FL process through additional iterations, which reduces the available time for the processing phase since both phases should be completed within the requested service latency.

In the end, the optimization problem defined in (32) can be rewritten as,

$$\mathbf{P3}: \mathcal{B}^* = \text{argmin}_{\mathcal{B}} \left\{ \frac{1}{M} \sum_{m=1}^{M} (\eta_1 T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 E_m(\rho_m(\beta_m), \alpha_m(\beta_m))) \right\} \quad (37)$$

subject to the constraints (19d)–(19g), (28)–(29), (33), (36) and,

$$0 \leq \beta_m \leq 1 \qquad \forall v_m \in \mathcal{V} \quad (38)$$

where (36) limits the maximum number of FL iterations performed by each VU based on the available FL process time and, according to (38), β_m can take any value between 0 and 1.

IV. Proposed Solutions

The solution space dimension for the problem **P3** can be estimated as $\mathcal{SP} = \Theta^{(M)}$, where Θ is the number of possible values taken by β_m, i.e., the smaller step size for β_m discretization, the bigger is the solution space. In a certain service area, the number of VUs requesting services can also be huge. Therefore, despite being simplified with respect to **P2**, solving **P3** for the whole set $\mathcal{V}$, even for a discrete solution space, is computationally expensive and requires exploring a huge solution space $\mathcal{SP}$; thus sub-optimal approaches operating on a subspace of $\mathcal{SP}$ are required. In order to address the problem, first, we propose an RSU-based clustering approach where each RSU performs the optimization for the VUs under its coverage. As a second scheme, we consider a distributed approach, where each VU performs the optimization by itself without considering the surrounding VUs. In both cases, a GA is proposed as the solution methodology.

In order to simplify the problem we assume that each VU will be assigned to the nearest RSU for computation offloading, hence:

$$a(m,n) = 1 \iff n = \text{argmin}_{n'}\{d_{m,n'}\} \quad \forall m, n' \quad (39)$$

where $d_{m,n'}$ is the distance between the mth VU and the n'th RSU. Each VU starts by downloading the FL model from the HAP and infers the initial offloading parameter α_m^0, supposed to be a random value between 0 and 1.

A. Clustered Approach

In the cluster-based sub-optimal approach we assume that it is possible to find β_m by considering the active VUs (i.e., VUs requesting offloading services) under each RSU coverage, where M_n corresponds to the VUs managed by the nth RSU. The RSU communication and computing resources are supposed to be equally shared among all active VUs in its coverage area. The solution vector $\mathcal{B}_n = \{\beta_1, \beta_2, \ldots, \beta_{M_n}\}$ is composed by the M_n values for all the VUs connected to nth RSU. We aim to determine $\mathcal{B}_n^*$, the optimal parameter set for the nth RSU. The overall optimal $\mathcal{B}^*$ can be determined by merging the solutions from all RSUs, i.e., $\mathcal{B}^* = \cup_n \mathcal{B}_n^*$. The problem originally formulated in (37) is thus modified as

$$\mathcal{B}_n^* = \text{argmin}_{\mathcal{B}_n} \left\{ \frac{1}{M_n} \sum_{m=1}^{M_n} [\eta_1 \cdot T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 \cdot E_m(\rho_m(\beta_m), \alpha_m(\beta_m))] \right\} \quad (40)$$

Algorithm 1: Clustered Approach.

Input: $N, M, \{d_{m,n}\}$
Output: $\mathcal{B}^*$
1: $a(m,n) = 1 \iff n = \text{argmin}_{n'}\{d_{m,n'}\} \quad \forall m.$
2: Find $M_n = \sum_{m=1}^{M} a(m,n), \forall n$
3: **for all** $n = 1, \cdots N$ **do**
4: **for all** $m \in M_n$ **do**
5: Find $\mathcal{B}_n^*$ by solving (40)
6: **end for**
7: **return** $\mathcal{B}^* = \{\mathcal{B}_1^*, \ldots, \mathcal{B}_n^*, \cdots \mathcal{B}_N^*\}$

In Algorithm 1 the steps used for the RSU based clustered optimization are presented. At the beginning, VUs are assigned to the RSUs based on the minimum distance criterion in (39), from which the number of VUs requesting services from each RSU is determined (Line 1-2). After this, the optimal set of $\mathcal{B}_n$ values for all the VUs associated with a given RSU is determined by using (40) (Line 3-6). In the end, the algorithm returns the solution set of all RSUs (Line 7).

1) Clustering Policies: In order to better understand the impact of the clustered approach we have considered three different clustering policies.

a) Full Clustering Policy (FC): In this policy, all the active M_n VUs of nth RSU cluster participate to the FL process before performing offloading, hence,

$$M_n = \sum_{m=1}^{M} a(m,n) \quad \forall n$$

b) Probabilistic Clustering Policy (PC): In this approach, we randomly classify the M_n VUs of nth RSU cluster into two subgroups $\hat{M}_n^1$ and $\hat{M}_n^2$. VUs belonging to $\hat{M}_n^1$ perform FL process with optimal β_m^* determined through (38) while VUs in $\hat{M}_n^2$ performs offloading with initially estimated offloading parameter α_m^0, i.e., $\beta_m = 0$. By this policy we would like to understand the impact of the VUs when participating to the FL process. The classification of VUs into two subgroups is based on a Bernoulli distribution where the probability of the m-th VU being in $\hat{M}_n^1$ is p, i.e., $P(m \in \hat{M}_n^1) = p$ and the probability being in $\hat{M}_n^2$ is $(1-p)$, i.e., $P(m \in \hat{M}_n^2) = (1-p)$.

c) Distance-Based Clustering Policy (DBC): In this approach the selection of the VUs is based on the available distance before they move out of the RSU coverage. In this policy we would like to give more importance to those VUs staying longer within the same RSU coverage; hence, we select them

for performing FL. Therefore for the nth RSU we have:

$$\hat{M}_n^1 = \left\{ m | \Pi_{m,n} \geq \hat{\Pi},\ m \in M_n \right\}$$

where $\hat{M}_n^1$ are VUs that perform FL iterations before the computation offloading process with optimal β_m determined through (40). $\hat{\Pi}$ is the distance bound used for partitioning VUs into two groups. The remaining VUs, will not participate into the FL training process, i.e., $\beta_m = 0$ and given by,

$$\hat{M}_n^2 = \left\{ m | m \notin \hat{M}_n^1,\ m \in M_n \right\}$$

such that $M_n = \hat{M}_n^1 \cup \hat{M}_n^2$

B. Distributed Approach

Due to its dynamic nature, predicting the exact VUs number and their characteristics even if within the same RSU is a difficult task. Some of the main reasons include VUs unpredictable velocity, directions, drivers' behaviors, different types of vehicles, etc. Moreover, in many situations, privacy-protective VUs are reluctant to share their information with surrounding nodes, limiting the VUs capability for understanding the surrounding environment. In this situation, each VU has to offload computation data towards RSU without knowing how many other VUs have already requested the services from that particular RSU with certain assumptions over available RSU resources. In such situations, VUs can act selfishly and assume that no other VUs have requested services from a selected RSU and its complete resource pool can be used. Here, we propose a VU-based distributed approach where VU makes similar assumptions while offloading data towards RSU nodes. Thus, in the VU-based distributed approach, we consider that VUs are not aware of nearby competing VUs and perform the optimization without considering them. The problem originally formulated in (37), is modified as

$$\beta_m^* = \text{argmin}_{\beta_m} \{ \eta_1 \cdot T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) + \eta_2 \cdot E_m(\rho_m(\beta_m), \alpha_m(\beta_m)) \} \ \forall m \quad (41)$$

It is possible to notice that in this case we suppose no mutual influence among different VUs.

C. Genetic Algorithm

We propose a GA-based solution for solving both cluster-based and distributed approaches. GAs are evolutionary search methods inspired by the theory of natural selection and genetics. GA process begins with an initial population space ($\mathcal{PS}$) that constitutes the set of possible solutions (i.e., individuals), each having a chromosome ($\mathcal{C}$). Through an iterative process, involving the creation of a new $\mathcal{PS}$ with possibly better individuals at each step, the sub-optimal solution is obtained. The evaluation process includes the analysis of each $\mathcal{C}$ of the current $\mathcal{PS}$ through a fitness function ($\mathcal{FF}$), the selection of a parent $\mathcal{C}$ is based on a selection function (S_f), then the formation of new individuals by using mutation and crossover GA operators. In the mutation process, a new $\mathcal{C}$ is formed by altering some of the genes in the selected solution from ($\mathcal{PS}$), while, in the crossover process, two chromosome sets with good fitness function constitute a $\mathcal{C}$ for the next generation by combining their genes. Each evaluation creates a better solution set and finally ends by providing a solution point with a higher fitness value. More comprehensive information on GA and evolutionary algorithms can be found in [44], while here we focus on the main elements for the sake of brevity.

a) Chromosome: In this work, we have considered $\mathcal{C}$ constituted by set of $\beta_m \in \mathcal{B}_n$ values for the nth RSU. Thus, each $\beta_m \in [0, 1]$ acts as a gene for $\mathcal{C}$.

b) Fitness Function: The $\mathcal{FF}$ allows to model the problem to be minimized considering also the constraints; hence, it is defined by using the objective function in (40), later written as $f(\mathcal{B}_n)$ for the nth RSU, plus three additional penalty functions related to the constraints in (28), (29) and (31). The fitness function $\mathcal{FF}(\mathcal{B}_n)$ is:

$$\mathcal{FF}(\mathcal{B}_n) = f(\mathcal{B}_n) + \Upsilon_1 \cdot \max(0, C1(\mathcal{B}_n)) + \Upsilon_2 \cdot \max(0, C2(\mathcal{B}_n)) + \Upsilon_3 \cdot \max(0, C3(\mathcal{B}_n)) \quad (42)$$

where Υ_1, Υ_2 and Υ_3 are the weighting coefficients for the penalty values, and:

$$C1(\mathcal{B}_n) = \sum_{M_n} \left(T_m(\rho_m(\beta_m), \alpha_m(\beta_m)) - \bar{T}_{x_m} \right)$$

$$C2(\mathcal{B}_n) = \sum_{M_n} \left(E_m(\rho_m(\beta_m), \alpha_m(\beta_m)) - E_m^{x_m, \mathrm{c}} \right)$$

$$C3(\mathcal{B}_n) = \sum_{M_n} \left(T_m^{\mathrm{FL}}(\rho_m(\beta_m)) + T_{m,n}^{\mathrm{off}}(\alpha_m(\beta_m)) - T_{m,n}^{\mathrm{soj}} \right)$$

where $C1(\mathcal{B}_n)$ is the additional fitness penalty for VUs not performing FL and offloading process within the service latency requirement, $C2(\mathcal{B}_n)$ is the penalty for not respecting the energy constraint defined in (29) and $C3(\mathcal{B}_n)$ is the supplementary penalty for VUs not performing the offloading process before moving out of RSU coverage.

c) Selection: The selection function S_f used for the parent selection is based on the roulette wheel selection technique, where the selection probability for an individual to be selected depends upon its fitness score. It should be noted that since our problem is latency and energy minimization, parents with the lowest fitness are selected at each round for reproduction stage.

d) Crossover: In the crossover operator, new chromosomes ($\mathcal{C}_1^{\text{new}}, \mathcal{C}_2^{\text{new}}$) are generated by alternating genes of the parents ($\mathcal{C}_1^{\text{old}}, \mathcal{C}_2^{\text{old}}$) from a crossover point. Thus, child chromosomes can be written as

$$\mathcal{C}_1^{\text{new}} = \Phi \mathcal{C}_1^{\text{old}} + (1 - \Phi) \mathcal{C}_2^{\text{old}}, \quad \mathcal{C}_2^{\text{new}} = \Phi \mathcal{C}_2^{\text{old}} + (1 - \Phi) \mathcal{C}_1^{\text{old}}$$

where Φ is the crossover point uniformly distributed in $[\Lambda, (1 + \Lambda)]$, i.e., $\Phi \sim U(-\Lambda, 1 + \Lambda)$

e) Mutation: We have used a Gaussian mutation technique where selected genes (β_m) from a child $\mathcal{C}$ can be altered by adding a random value from a Gaussian distribution, i.e., $\beta_m \rightarrow \beta_m + \nu$, where, ν is a random variable with a Gaussian distribution, i.e., $\nu \sim \mathcal{N}(\mu, \sigma^2)$.

Algorithm 2: The proposed GA-based Approach.

Input: $\mathcal{FF}, G_{\max}, M_n, \Phi, \nu$
Output: $\mathcal{B}_n^*$

1: Generate the initial population space $\mathcal{PS}$ with each $\beta_m \in [0,1]$
2: **while** $i \leq G_{\max}$ **do**
3: **function** Evaluate$\mathcal{PS}$
4: Find $\mathcal{FF}(\mathcal{C})$, $\mathbf{foralldo}\mathbf{C} \in \mathcal{PS}$.
5: **end Function**
6: **function** Search$\mathcal{PS}$
7: Select better fit individuals using S_f
8: **end Function**
9: **function** Create$\mathcal{PS}$
10: Generate new $\mathcal{C}$s through Crossover and Mutation (using Φ, ν).
11: Integrate $\mathcal{C}$s with current $\mathcal{PS}$ and sort them using fitness scores i.e., $\mathcal{FF}(\mathcal{C})$
12: **end Function**
13: Replace current $\mathcal{PS}$ with new best set of $\mathcal{C}$s.
14: $i = i + 1$
15: **end while**
16: **return** $\mathcal{B}_n^*$

Algorithm 2 shows the steps used during the implementation of GA for the clustered approach. The main GA steps include the evaluation of $\mathcal{PS}$ (Line 3-5), selection of better fit individuals as parent $\mathcal{C}$s (Line 6-8), generation of new possibly better fit $\mathcal{C}$s for the next generation (Line 9-12). The algorithm terminates after a maximum number of iterations $G_{\max}$ are reached. A similar process can be used for the distributed case by considering the individual VUs, where GA performs optimization for each $m \in M_n$ separately. It is worth to be noticed that, when GA is applied to the clustered approaches a set of VUs participates in the GA process. GA process can produce, for some of these VUs, a solution with $\beta_m = 0$, corresponding to exclude such VUs from the FL process. Thus, inherently, GA process is also able to optimize the clusters' size by including/excluding VUs from the FL process, if there is an advantage in terms of cost.

D. Limited Search-Based Heuristic Approach (LS-HuA)

In order to compare the results with a simpler while sub-optimal solution, we propose also an intuitive heuristic approach where we consider a reduced-size solution space $\hat{\mathcal{SP}}$ through a user-defined parameter θ_{hu} representing the number of possible values taken by the parameter β_m. In this way we are going to optimize the problem while considering only a subset of possible solutions. For example, in case of $\theta_{hu} = 5$, $\beta_m \in \{0, 0.2, 0.4, 0.6, 0.8\}$. We do not consider the case with $\beta_m = 1$, since it corresponds to completely assign the time interval to the FL process, resulting in an always infeasible solution for active VUs having tasks to be offload. The smaller values of θ_{hu} reduces the simulation time, while limiting the accuracy of a solution provided. On the other hand, larger values of θ_{hu} allow the user to search over the larger $\mathcal{SP}$ for finding an optimal solution (i.e., exhaustive search). Also in this case $\beta_m = 0$ corresponds to exclude the mth VU from the FL process.

Algorithm 3: Limited Search-based Heuristic Approach.

Input: M_n, θ_{hu}
Output: $\mathcal{B}_n^{hu}$

1: Create $\hat{\mathcal{SP}} = \{\beta_n\}$ of size $\theta_{hu}^{(M_n)}$ with all possible solution points to be searched in the reduced-size solution space
2: Initialize $f_{hu} = \infty$,
3: **for all** $\mathcal{B}_n \in \hat{\mathcal{SP}}$ **do**
4: Use (40) for finding total cost $f(\mathcal{B}_n)$
5: Determine all constraint functions values
6: **if** $f(\mathcal{B}_n) \leq f_{hu}$ and all constraints are satisfied **then**
7: $f_{hu} = f(\mathcal{B}_n)$ and $\mathcal{B}_n^{hu} = \mathcal{B}_n$
8: **end if**
9: **end for**
10: **if** $f_{hu} = \infty$ (i.e., no feasible solution found) **then**
11: $\mathcal{B}_n^{hu} = \{0\}_{1 \times M_n}$
12: **end if**
13: **return** $\mathcal{B}_n^{hu}$

Algorithm 3 lists the steps followed during the search process. It includes the creation of a reduced search space (Line 1), initializing the cost function value (f_{hu}) that stores the optimal cost for each iteration (Line 2), and iterating over all possible solution points $(\mathcal{B}_n)$ from $\hat{\mathcal{SP}}$ for finding the best possible solution (Line 3-12). In the end, the algorithm returns the best possible solution point $\mathcal{B}_n^{hu}$ found through iterations. In case there is no feasible solution available, VU decides to offload without performing any FL iteration.

E. Optimal Offloading Parameter

Here we aim at finding a closed form expression for the optimal offloading parameter $\alpha_m^{\text{opt}}(\beta_m)$ having set β_m. It should be noticed that this particular analysis is carried out by considering that all system parameters are known in advance, which is not the case in reality given the uncertainty of the environment. Thus the results are used for comparison.

In case we fix β_m it is possible to obtain the optimal offloading parameter $\alpha_m^{\text{opt}}(\beta_m)$ by resorting to the equality conditions in (16), (29) and (31). Resorting to (16), the optimal offloading parameter $(\alpha_m^{\text{T}_1})$ implies that:

$$T_{m,n}^{\text{off}}(\alpha_m^{\text{T}_1}) = T_m^{\text{loc}}(\alpha_m^{\text{T}_1}) \tag{43}$$

Exploiting (11) and (14), we have the following:

$$T_{m,n}^{\text{off}}(\alpha_m^{\text{T}_1}) = \alpha_m^{\text{T}_1} \cdot \hat{T}_{m,n}^{\text{off}}, \quad T_m^{\text{loc}}(\alpha_m^{\text{T}_1}) = (1 - \alpha_m^{\text{T}_1}) \cdot T_m^{x_m,\text{c}}$$

Hence, exploiting (43) we have,

$$\alpha_m^{\text{T}_1} = \frac{T_m^{x_m,\text{c}}}{T_m^{x_m,\text{c}} + \hat{T}_{m,n}^{\text{off}}} \tag{44}$$

In addition, the equality condition in (31) allows to achieve an optimal offloading parameter $\alpha_m^{\text{T}_2}(\beta_m)$ so that,

$$T_m^{\text{FL}}(\rho_m(\beta_m)) + T_{m,n}^{\text{off}}(\alpha_m^{\text{T}_2}(\beta_m)) = T_{m,n}^{\text{soj}}$$

where,

$$T^{\text{off}}_{m,n}(\alpha^{\text{T}_2}_m(\beta_m)) = \alpha^{\text{T}_2}_m(\beta_m) \cdot \hat{T}^{\text{off}}_{m,n}$$

which returns,

$$\alpha^{\text{T}_2}_m(\beta_m) = \frac{T^{\text{soj}}_{m,n} - T^{\text{FL}}_m(\rho_m(\beta_m))}{\hat{T}^{\text{off}}_{m,n}} \tag{45}$$

In case the mth VU performs the FL process for a longer time and goes out of the coverage of RSU, i.e., $T^{\text{soj}}_{m,n} < T^{\text{FL}}_m(\rho_m(\beta_m))$, it will not be able to offload any data towards the RSU, i.e., $\alpha^{\text{T}_2}_m(\beta_m) = 0$. Hence, (45) can be rewritten as,

$$\alpha^{\text{T}_2}_m(\beta_m) = \max\left\{0, \frac{T^{\text{soj}}_{m,n} - T^{\text{FL}}_m(\rho_m(\beta_m))}{\hat{T}^{\text{off}}_{m,n}}\right\} \tag{46}$$

Following the energy constraint defined in (29) equality holds for a particular optimal offloading parameter $\alpha^{\text{E}_1}_m(\beta_m)$ and can be written as,

$$E^{\text{FL}}_m(\rho_m(\beta_m)) + E^{x_m}_m(\alpha^{\text{E}_1}_m(\beta_m)) = E^{x_m,\text{c}}_m$$

Exploiting (12) and (17), we have the following:

$$E^{x_m}_m(\alpha^{\text{E}_1}_m(\beta_m)) = \alpha^{\text{E}_1}_m(\beta_m) \cdot \hat{E}^{\text{off}}_{m,n} + (1 - \alpha^{\text{E}_1}_m(\beta_m)) \cdot E^{x_m,\text{c}}_m$$

that returns,

$$\alpha^{\text{E}_1}_m(\beta_m) = \frac{-E^{\text{FL}}_m(\rho_m(\beta_m))}{\hat{E}^{\text{off}}_{m,n} - E^{x_m,\text{c}}_m} \tag{47}$$

In such cases where $\hat{E}^{\text{off}}_{m,n} > E^{x_m,\text{c}}_m$, performing computation offloading is not an option since it requires additional energy and that results into $\alpha^{\text{E}_1}_m(\beta_m) = 0$. Therefore, (47) can be modified to,

$$\alpha^{\text{E}_1}_m(\beta_m) = \max\left\{0, \frac{-E^{\text{FL}}_m(\beta_m)}{\hat{E}^{\text{off}}_{m,n} - E^{x_m,\text{c}}_m}\right\} \tag{48}$$

In the end, (44), (46), and (48) are considered for finding $\alpha^{\text{opt}}_m(\beta_m)$ as:

$$\alpha^{\text{opt}}_m(\beta_m) = \min\left\{\alpha^{\text{T}_1}_m, \alpha^{\text{T}_2}_m(\beta_m), \alpha^{\text{E}_1}_m(\beta_m)\right\} \tag{49}$$

This procedure is used as a reference value in the following for testing the effectiveness of the estimated offloading parameters that depends on the FL and, in turns, on its iterations.

V. Numerical Results

Numerical results are obtained through computer simulations with Matlab. A variable number of VUs between 100 and 1000 are considered, assuming that each one is generating tasks with a probability equal to 0.2, while the remaining have no task to be offloaded. VUs are uniformly distributed in a two-lane road and travel in either directions with a velocity $\bar{v}_m$ equal to 10 m/s. Moreover, 80 RSUs are randomly placed on either sides of the lanes. The task latency requirement $(\bar{T}_{x_m})$ has been set to 2 s; this value is consistent with other works in the literature [22], [37] considering similar scenarios and applications. The other parameters considered in simulation are listed in Table I.

The GA weight coefficients are $\Upsilon_1, \Upsilon_3 = 10$, $\Upsilon_2 = 1$, while the crossover function parameter is $\Lambda = 0.1$, and the mutation function parameters are $\mu = 0.02$, $\sigma = 0.1$. Moreover, we set an initial population of 30 chromosomes and G_{max}=50, α^0_m is uniformly distributed between 0 and 1, while $\mathcal{K}$ is 2000, and both $|w^i_m|$ and $|w_G|$ have size 1000 bits. The parameter γ is set to 0.4 while estimating offloading parameters. In DBC policy $\hat{\Pi}$ is equal to $R_{r,n}/2$, while $p = 0.5$ is used in PC. Also, the numerical value used for both η_1 and η_2 is 0.5. Finally, we have considered $\theta_{hu} = 6$ when evaluating the results for the LS-HuA.

TABLE I
Simulation Parameters

Simulation parameters	
HAP Beam Coverage (R_{HAP})	2 Km
RSU Coverage ($R_{r,n}$)	25 m
Task Size (D_{x_m})	2.5 MB
Task Computation (Ω_{x_m})	$10^3 \times D_{x_m}$ FLOPS
Task Results ($D_{x_m,\text{rx}}$)	0.5 MB
VU Flops ($c_{v,n} \cdot f_{v,n}$)	8 GFLOPS
VU Tx. Energy (P^{tx}_m)	1.3 W [23]
VU Rx. Energy (P^{rx}_m)	1.1 W [23]
VU Comp. Energy (P^{c}_m)	0.9 W [23]
RSU Flops ($c_{r,n} \cdot f_{r,n}$)	80 GFLOPS
HAP Beam Bandwidth (B_{HAP})	100 MHz
RSU Bandwidth ($B_{r,n}$), $\forall n$	10 MHz
HAP Altitude (h_{HAP})	20 Km [17]

In the following, we present the results by comparing the proposed GA approach with LS-HuA and two static benchmarks:

- *Computation Offloading without Performing any FL Iterations (Without FL):* In this approach, each VU decides to offload data without performing any FL iteration. Therefore, the offloading operation is performed with α^0_m without adding any FL cost. Since the initial value of offloading parameter may or may not be optimal, this approach cannot guaranty the optimal performance. Though this approach can have a reduced cost, VU performs the offloading operation without taking into account the available time and energy resources which may diminish performance in terms of constraint failures.
- *Computation Offloading by Performing Complete FL Iterations (Complete FL):* In this particular method, each VU performs the ρ^{opt} FL iterations before offloading data towards RSU. Thus the offloading operation is performed with α^{opt}_m, as defined in (49), when β_m is such that $\rho_m = \rho^{\text{opt}}$. Though VUs can perform offloading with optimal offloading parameters, it is not always feasible to perform ρ^{opt} FL iterations with limited service time, sojourn time, and energy of VU, which limits the performance of this approach.

These two benchmarks do not consider the available resources of VUs while making computation offloading decisions and may have a sub-optimal performance over long-term simulations. In the following figures, GA-FC, GA-DBC, GA-PC, and GA-D are the acronyms used for the Genetic Algorithm technique with FC, DBC, PC, and Distributed Clustering approaches, respectively.

1) Avg. Latency and Energy Cost With Varying VUs: In Fig. 7, the average cost in terms of joint latency and energy consumed for both FL and task processing phases using a variable number of VUs is shown. The results show that GA and LS-HuA techniques have a considerable advantage over the

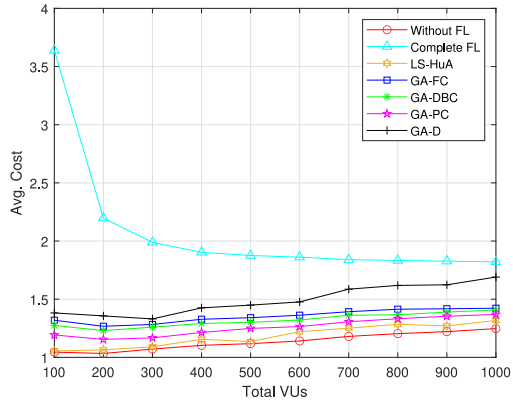

Fig. 7. Cost Function.

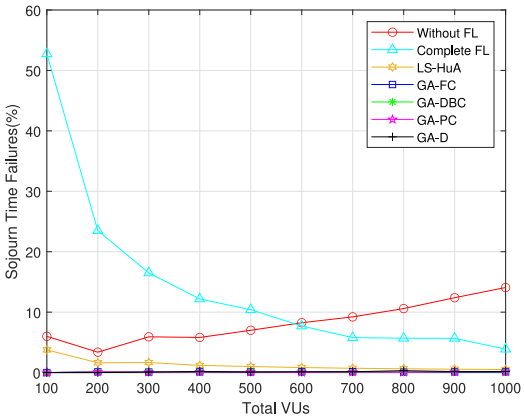

Fig. 8. Percentage of VUs with sojourn time constraint violation.

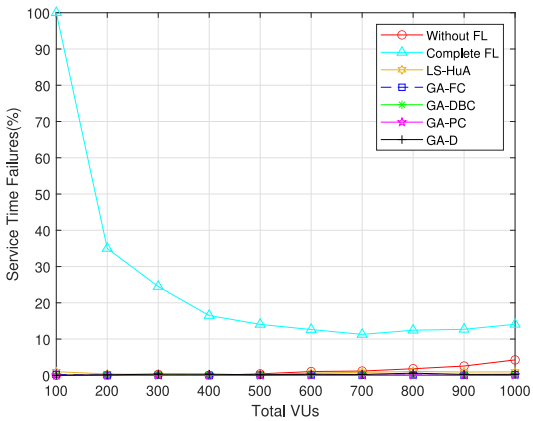

Fig. 9. Percentage of VUs with service time constraint violation.

Complete FL approach with reduced cost values. Even though the *Without FL* approach has the minimum cost among all the proposed methods, it cannot guarantee a reliable performance in terms of service latency, sojourn time, and energy constraints, as shown and discussed later in Figs. 8– 10. The proposed Clustered GA approaches (i.e., GA-FC, GA-DBC, GA-PC), thanks to a better knowledge of the surrounding environment, performs FL and task processing with a lower cost along with better reliability, which can benefit several latency-critical services demanded by VUs having limited energy resources. Since the required

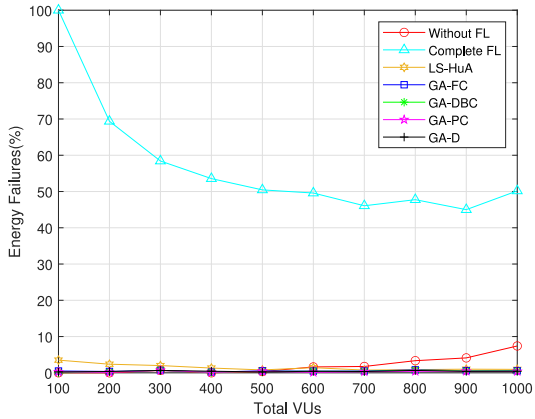

Fig. 10. Percentage of VUs violating the Energy Constraint in (29) .

FL iterations to achieve model convergence reduces with the participation of more VUs, the cost of the *Complete FL* process decreases with increasing VUs, but still it fails to achieve the overall performance of proposed GA methods.

2) Performance in Terms of Sojourn Time Failures: Fig. 8 shows the percentage of number of VUs failing to perform the offloading operation before leaving the RSU coverage. According to constraint (31), each VU should complete both FL and task offloading processes within available sojourn time. The two benchmark methods lack suitable flexibility while performing the offloading operations as both methods do not utilize the available latency resources properly while performing the offloading operations. That results in higher failures since they are not able to perform both FL and task offloading operations in a limited sojourn time. It should be noted that the complete FL approach has a falling curve, which is due to the fact that by the increase in the number of VUs in a service area, a shorter time will be required to achieve FL convergence. On the other hand, both GA schemes and LS-HuA approach perform an adequate number of FL iterations, before performing the offloading operations, and, as a result, have very few failures with reduced cost. The performance of the *Without FL* worsens with an increasing number of VUs, and at a certain point, it has even higher failures than the *Complete FL* approach.

3) Performance in Terms of Service Time Outage: Fig. 9 shows the percentage of VUs failing to perform both FL and the task processing operation within a demanded service latency. The significant performance improvement in terms of a reduced number of failures can be observed in the GA and LS-HuA results, comparing with the benchmark methods. This is mainly because of the improper allocation of VUs available resources towards FL and task processing phases in the benchmark methods. These results also highlight the importance of proper allocation of VU resources for the FL and task processing phases (estimation of $\mathcal{B}$), for improving the overall VNs performance.

4) Performance in Terms of Energy: Fig. 10 shows the percentage of VUs violating the energy constraint in (29). Since each FL iteration costs energy, performing ρ^{opt} iterations for each VU before offloading during the *Complete FL* approach

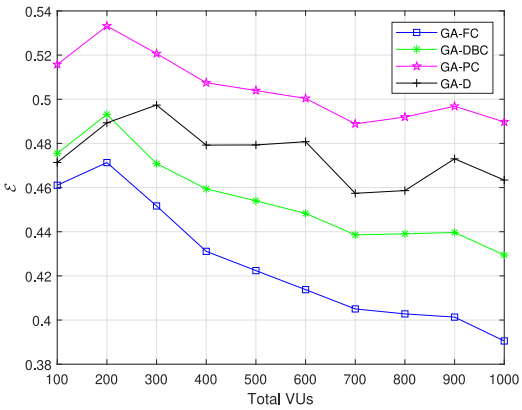

Fig. 11. Avg. Offloading Error.

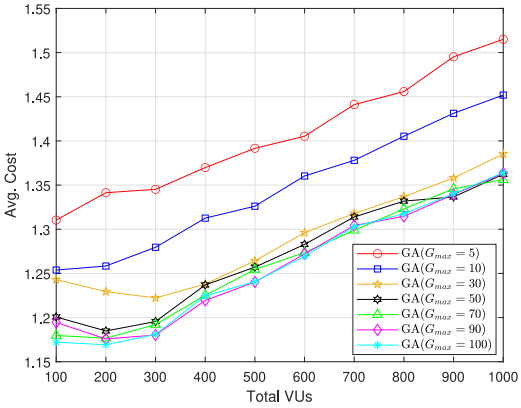

Fig. 12. GA Performance Vs Iterations.

decreases its reliability in terms of respecting VUs energy constraint and can be seen from these results. On the other hand, GA and LS-HuA approaches have better performance since they allocate a proper number of FL iterations before offloading. Thus these results highlight the importance of performing an adequate number of FL by taking into account the VUs available resources to achieve a reliable performance with FL in dynamic VNs.

5) Offloading Performance: Fig. 11 shows the average error when estimating the offloading parameters. For a given set M of VUs, the error in the estimation process is measured by using the Root Mean Square Error (RMSE) as,

$$\mathcal{E}(M,\mathcal{B}^*) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left|\left(\alpha_m^{\text{opt}}(\beta_m^*)\right)^2 - \left(\alpha_m(\beta_m^*)\right)^2\right|}$$

where $\alpha_m^{\text{opt}}(\beta_m^*)$ is the offloading parameter estimated in (49), while $\alpha_m(\beta_m^*)$ is derived through (34). The value $\mathcal{E}$ decreases for the GA-FC approach with higher values of M, as the number of surrounding VUs increases. Other clustered approaches have reduced offloading performance since only a lower number of VUs participate in the optimization process before performing offloading. Also, with limited available information, the distributed approach fails to adapt itself properly.

6) Impact of GA Iterations: In the case of GA, the performance can be improved by increasing the number of iterations of the GA. In Fig. 12, we compare the performance in terms of average latency and energy cost by considering a different number of GA iterations in the GA-FC policy. It can be seen that as the number of iterations increases, GA performance improves. However, after a certain number of iterations (i.e., 50), performance of the GA process becomes stable, thus, performing a higher number of iterations can only increase the time complexity of the GA process by several folds, without major gains in terms of offloading solution.

VI. Conclusion

In this work, we performed the optimization for a joint FL and task processing problem over the integrated air-ground network of HAP-assisted VN. For this, we first modeled the computation offloading problem in the vehicular scenario in which each VU can offload a portion of their tasks to the surrounding RSUs. Next, an integrated air-ground network-based FL platform was introduced, where powerful HAPs act as an FL server to assist several VUs (i.e., FL clients) in estimating the better offloading parameters. A joint computation offloading and FL process optimization problem aiming at minimization of overall latency and energy cost was formulated. The proposed solution methods include the RSU cluster-based approach with several clustering policies and distributed approaches. An evolutionary search-based GA was proposed to find both allocated time for the two phases and estimating the offloaded portions for the VUs. Simulation results demonstrate that our proposed GA-based approaches, when compared with other benchmark solutions, show a network-wide performance improvement.

As future directions of this work, we point out the extension to autonomous driving scenarios, where VUs data can be analyzed for solving vehicular problems through the proposed FL platform. Some other challenges to be faced include (i) a proper RSU selection for offloading, (ii) the possibility of considering a network of multiple decentralized HAPs for a higher fault-tolerance, (iii) the optimization of the number of VUs participating in the FL process considering their available resources, (iv) the extension to intermediate FL layers (e.g., LAPs, UAVs, RSUs) for reducing the communication/computation costs during FL data processing and communication (i.e., Hierarchical FL).

References

[1] N. Lyons and G. Lzroiu, "Addressing the COVID-19 crisis by harnessing internet of things sensors and machine learning algorithms in data-driven smart sustainable cities," *Geopolitics, History, Int. Relations*, vol. 12, no. 2, pp. 65–71, Oct. 2020.

[2] B. Ji *et al.*, "Survey on the internet of vehicles: Network architectures and applications," *IEEE Commun. Standards Mag.*, vol. 4, no. 1, pp. 34–41, Mar. 2020.

[3] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2018.

[4] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[5] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3082927.

[6] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Mobile edge computing partial offloading techniques for mobile urban scenarios," in *Proc. IEEE Glob. Commun. Conf.*, Abu Dhabi, UAE, 2018, pp. 1–6.
[7] F. Mashhadi, S. A. S. Monroy, A. Bozorgchenani, and D. Tarchi, "Optimal auction for delay and energy constrained task offloading in mobile edge computing," *Comput. Netw.*, vol. 183, 2020, Art. no. 107527.
[8] P. Liu, J. Li, and Z. Sun, "Matching-based task offloading for vehicular edge computing," *IEEE Access*, vol. 7, pp. 27628–27640, Feb. 2019.
[9] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 94–101, Jun. 2018.
[10] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, Oct. 2019.
[11] H. Fang, X. Wang, and S. Tomasin, "Machine learning for intelligent authentication in 5G and beyond wireless networks," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 55–61, Oct. 2019.
[12] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Workshop Private Multi-Party Mach. Learn.*, 2016, https://arxiv.org/abs/1610.05492
[13] W. Gao, Z. Zhao, G. Min, Q. Ni, and Y. Jiang, "Resource allocation for latency-aware federated learning in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 17, no. 12, pp. 8505–8513, Dec. 2021.
[14] J. Qiu, D. Grace, G. Ding, M. D. Zakaria, and Q. Wu, "Air-ground heterogeneous networks for 5G and beyond via integrating high and low altitude platforms," *IEEE Trans. Wireless Commun.*, vol. 26, no. 6, pp. 140–148, Dec. 2019.
[15] X. Cao, P. Yang, M. Alzenad, X. Xi, D. Wu, and H. Yanikomeroglu, "Airborne communication networks: A survey," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1907–1926, Sep. 2018.
[16] Y. Shibata, N. Kanazawa, M. Konishi, K. Hoshino, Y. Ohta, and A. Nagate, "System design of gigabit HAPS mobile communications," *IEEE Access*, vol. 8, pp. 157 995–158 007, Aug. 2020.
[17] G. K. Kurt *et al.*, "A vision and framework for the high altitude platform station (HAPS) networks of the future," *IEEE Commun. Surv. Tuts.*, vol. 23, no. 2, pp. 729–779, Apr.–Jun. 2021.
[18] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.
[19] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Netw. Appl.*, vol. 26, pp. 1145–1168, 2021.
[20] R. A. Dziyauddin *et al.*, "Computation offloading and content caching and delivery in vehicular edge network: A survey," *Comput. Netw.*, vol. 197, 2021, Art. no. 108228.
[21] X. Li, Y. Dang, M. Aazam, X. Peng, T. Chen, and C. Chen, "Energy-efficient computation offloading in vehicular edge cloud computing," *IEEE Access*, vol. 8, pp. 37 632–37 644, Feb. 2020.
[22] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.
[23] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. Salinas, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2021.
[24] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14 198–14 211, Dec. 2020.
[25] Q. Ren, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and J. Chen, "Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems," 2021, *arXiv:2106.14928*. [Online]. Available: https://arxiv.org/abs/2106.14928
[26] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "EC-SAGINs: Edge computing-enhanced space-air-ground integrated networks for internet of vehicles," *IEEE Internet Things J.*, 2021, to be published, doi: 10.1109/JIOT.2021.3052542.
[27] C. Sun, W. Ni, and X. Wang, "Joint computation offloading and trajectory planning for UAV-assisted edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5343–5358, Aug. 2021.
[28] J. Hu, C. Chen, L. Cai, M. R. Khosravi, Q. Pei, and S. Wan, "UAV-assisted vehicular edge computing for the 6G internet of vehicles: Architecture, intelligence, and challenges," *IEEE Commun. Standards Mag.*, vol. 5, no. 2, pp. 12–18, Jun. 2021.
[29] A. M. Elbir, B. Soner, and S. Coleri, "Federated learning in vehicular networks," 2020, *arXiv:2006.01412*. [Online]. Available: https://arxiv.org/abs/2006.01412
[30] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5341–5351, Aug. 2021.
[31] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular Internet of Things: Recent advances and open issues," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 45–61, May 2020.
[32] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.
[33] S. Wang *et al.*, "Federated learning for task and resource allocation in wireless high altitude balloon networks," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17460–17475, Dec. 2021.
[34] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Update aware device scheduling for federated learning at the wireless edge," in *Proc. 2020 IEEE Int. Symp. Inf. Theory*, Los Angeles, CA, USA, 2020, pp. 2598–2603.
[35] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, Aug. 2020.
[36] W. Bao, C. Wu, S. Guleng, J. Zhang, K.-L. A. Yau, and Y. Ji, "Edge computing-based joint client selection and networking scheme for federated learning in vehicular IoT," *China Commun.*, vol. 18, no. 6, pp. 39–52, Jun. 2021.
[37] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3384–3398, Mar. 2020.
[38] M. Chen *et al.*, "Distributed learning in wireless networks: Recent progress and future challenges," 2021, *arXiv:2104.02151*. [Online]. Available: https://arxiv.org/abs/2104.02151
[39] M. Isaksson and K. Norrman, "Secure federated learning in 5 G mobile networks," in *Proc. GLOBECOM IEEE Glob. Commun. Conf.*, Taipei, Taiwan, 2020, pp. 1–6.
[40] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.
[41] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.
[42] X. Mo and J. Xu, "Energy-efficient federated edge learning with joint communication and computation design," *J. Commun. Inf. Netw.*, vol. 6, no. 2, pp. 110–124, Jun. 2021.
[43] N. Sagias, G. Tombras, and G. Karagiannidis, "New results for the shannon channel capacity in generalized fading channels," *IEEE Commun. Lett.*, vol. 9, no. 2, pp. 97–99, Feb. 2005.
[44] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Berlin, Germany: Springer, 2015.

Swapnil Sadashiv Shinde (Student Member, IEEE) received the M.S. degree in telecommunication engineering in 2020 from the University of Bologna, Bologna, Italy, where he is currently working toward the Ph.D. degree. From 2015 to 2017, he was a Project Engineer with the Indian Institute of Technology Kanpur, Kanpur, India. His research focuses on the connected vehicles for beyond 5G scenarios.

Arash Bozorgchenani (Member, IEEE) received the Ph.D. degree in telecommunications and IT from the University of Bologna, Bologna, Italy. Since 2020, he has been a Research Associate with Lancaster University, Lancaster, U.K. He spent one year as a Postdoctoral Researcher with the University of Bologna. He is involved in both national Gaucho (PRIN 2015, Italy) and European (H2020 SANCUS) projects. His research interests include resource allocation, machine learning, and optimization techniques in wireless communications.

Daniele Tarchi (Senior Member, IEEE) received the Ph.D. degree in informatics and telecommunications engineering from the University of Florence, Florence, Italy, in 2004. He is currently an Associate Professor with the University of Bologna, Bologna, Italy. He is the author of more than 130 published articles in international journals and conference proceedings. His research interests mainly include wireless communications and networks, satellite communications and networks, edge computing, fog computing, smart cities, and optimization techniques.

Qiang Ni (Senior Member, IEEE) is currently a Professor with the School of Computing and Communications, Lancaster University, Lancaster, U.K. He has authored or coauthored more than 300 papers in his research filed, which include future generation communications and networking, including green communications/networking, millimeter-wave wireless, cognitive radio systems, 5G/6G, SDN, cloud networks, edge computing, dispersed computing, IoT, cyber physical systems, AI/machine learning, and vehicular networks. He was an IEEE 802.11 Wireless Standard Working Group Voting Member and a contributor to various IEEE wireless standards.