VREM-FL: Mobility-Aware Computation-Scheduling Co-Design for Vehicular Federated Learning

Luca Ballotta, Nicolò Dal Fabbro, Giovanni Perin, *Member, IEEE*, Luca Schenato, *Fellow, IEEE*, Michele Rossi, *Senior Member, IEEE*, and Giuseppe Piro, *Member, IEEE*

***Abstract*—Assisted and autonomous driving are rapidly gaining momentum and will soon become a reality. Artificial intelligence and machine learning are regarded as key enablers thanks to the massive amount of data that smart vehicles will collect from onboard sensors. Federated learning is one of the most promising techniques for training global machine learning models while preserving data privacy of vehicles and optimizing communications resource usage. In this article, we propose vehicular radio environment map federated learning (VREM-FL), a computation-scheduling co-design for vehicular federated learning that combines mobility of vehicles with 5G radio environment maps. VREM-FL jointly optimizes learning performance of the global model and wisely allocates communication and computation resources. This is achieved by orchestrating local computations at the vehicles in conjunction with transmission of their local models in an adaptive and predictive fashion, by exploiting radio channel maps. The proposed algorithm can be tuned to trade training time for radio resource usage. Experimental results demonstrate that VREM-FL outperforms literature benchmarks for both a linear regression model (learning time reduced by 28%) and a deep neural network for semantic image segmentation (doubling the number of model updates within the same time window).**

***Index Terms*—5G, Federated Learning, optimization, REM, resource management, scheduling, vehicular networks.**

I. Introduction

VEHICULAR communications are expected to grow dramatically in the next few years according to the 5G Automotive Association, with applications requiring data rates of several tens of Gbit/s [1]. Siemens estimated that the data generated by autonomous vehicles will range from 3 to 40 Gbit/s depending on the autonomy level [2], which amounts to up to 19 TB every hour.

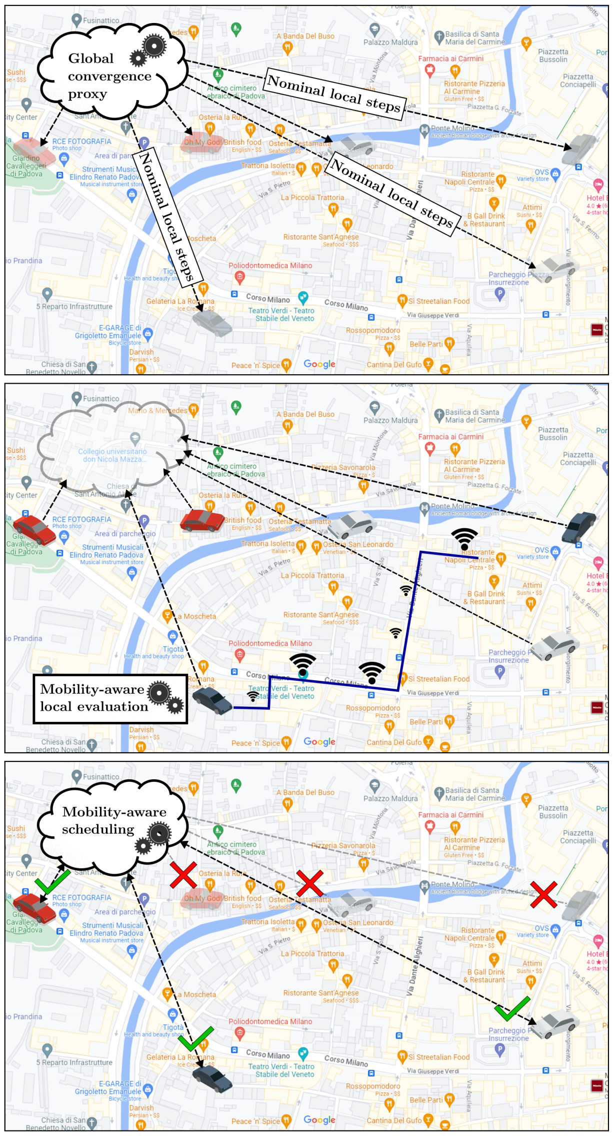


Fig. 1: Illustration of the main phases of VREM-FL, a mobility-aware co-design for allocation of communication and computation resources in vehicular FL.

The huge availability of data in modern vehicular networks paves the way to training, possibly at runtime, large deep learning models [3]. Nowadays, this is mostly accomplished via *federated learning* (FL), which is the leading solution to train neural networks from decentralized datasets. FL offers several advantages, such as a simple and flexible aggregation phase, and it preserves privacy because end-user data are never

This work was supported in part by the Italian Ministry of Education, University and Research (MIUR) through the PRIN Project n. 2017NS9FEY “Realtime Control of 5G Wireless Networks”, by the European Union under the Italian National Recovery and Resilience Plan of NextGenerationEU, partnership on “Telecommunications of the Future” (PE0000001 – program “RESTART”) and by the EU project ROBUST-6G (GA no. 101139068). Views and opinions expressed in this work are of the authors and may not reflect those of the funding institutions. *(Luca Ballotta, Nicolò Dal Fabbro, and Giovanni Perin contributed equally to this work).*

Luca Ballotta is with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: l.ballotta@tudelft.nl).

Nicolò Dal Fabbro is with the department of Electrical and Systems Engineering, University of Pennsylvania, 19104, USA (e-mail: ndf96@seas.upenn.edu).

Giovanni Perin, Luca Schenato, and Michele Rossi are with the Department of Information Engineering, University of Padova, 35131 Padova, Italy (e-mail: {giovanni.perin.1; l.schenato; michele.rossi}@unipd.it).

Michele Rossi is also with the Department of Mathematics “Tullio Levi-Civita,” University of Padova, 35121 Padova, Italy.

Giuseppe Piro is with the Department of Electrical and Information Engineering, Politecnico di Bari, 70125 Bari, Italy, and with the Consorzio Nazionale Interuniversitario per le Telecomunicazioni, 43124 Pisa, Italy (e-mail: giuseppe.piro@poliba.it).

transmitted, but only the model weights are sent from the end-users to the central aggregator.

Remarkably, despite the extensive amount of work available on communication-constrained [4], [5] and channel-aware [6], [7] FL algorithms, scheduling policies for vehicular networks that jointly consider mobility, communication/channel resources, and learning aspects are currently lacking.

In the present article, these aspects are *jointly* tackled for the first time, proposing vehicular radio environment map (REM) federated learning (VREM-FL), an FL scheduler that orchestrates the computation and the transmission of local models from the end users (the vehicles) to the central model aggregator (at the roadside network). VREM-FL exploits the availability of REMs to pick the best instants for the vehicles' local model transmission. This is possible because REMs are stable over time [8], [9] as they depend on static obstacles, such as buildings in urban environments. Hence, their knowledge can be combined with information on the planned user routes to improve FL transmission schedules. The optimization criteria correspond to minimizing the channel resources that are wasted due to transmitting when the channel conditions are poor, and to meeting a deadline for the global model update.

VREM-FL organizes resource allocation into the three stages shown in Fig. 1: (top box) the central orchestrator, based on channel and computation resources, decides how many vehicles participate in the current round, proposes an initial number of steps to update their local models, and sets a maximal round latency (deadline); (middle box) all vehicles, based on their private planned routes, learning metrics, and available REMs, independently estimate the channel quality they will experience in the near future, adjust the number of local steps, and decide when to transmit their own models. Then, they communicate to the aggregator an estimated *participation cost* that encompasses the usage of communication and training resources; (bottom box) the orchestrator combines the received vehicles' costs with centrally available information about fairness of participation across vehicles, scheduling those with the highest priority. The scheduled vehicles send their updated local models to the orchestrator, which finally refines the global model.

Contributions: Our contributions are summarized as follows.

- We propose a novel computation-scheduling co-design tailored to vehicular FL, VREM-FL. The knowledge of vehicles' routes and REMs is leveraged to optimize the communication resources used by vehicles, ensuring high learning performance while balancing training time and network resources. To the best of our knowledge, this is the first co-design of resource allocation in vehicular federated learning that exploits mobility and channel information supplied by REMs to account for time-varying channel quality, as typically experienced by traveling vehicles.
- We simulate a realistic environment using the street map of the city of Padova, Italy, and the popular simulator of urban environment (SUMO) [10]. We also evaluate the performance of VREM-FL on a real-world mobility dataset of taxi cabs in Rome, Italy [11]. On these maps, base stations (BSs) are deployed according to typical parameters of fifth generation (5G) cellular systems [12], to provide communications services to the vehicles. To capture realistic settings where channel quality measures are available at a limited number of locations, REMs are obtained via Gaussian process regression [9], [13].
- VREM-FL is evaluated via an extensive simulation campaign comprising (1) controlled experiments with a least squares (LS) toy example and (2) a realistic scenario on deep learning for a real-world semantic segmentation task with the popular dataset `ApolloScape` [14]. Also, we compare VREM-FL with literature benchmarks such as *federated averaging* (FedAvg) [15] and the algorithms in [16], where clients are selected based on a fairness metric, and in [17], where client selection is based on channel gain estimates at the beginning of learning rounds.

Organization of the Article: In Section II, the state-of-the-art is presented. Section III provides a general overview of VREM-FL. In Section IV, the system model is presented, including the FL setup and the radio environment settings. The problem is defined in Section V, while VREM-FL is detailed in Section VI. Numerical simulation results demonstrating the effectiveness of VREM-FL are presented in Section VII. Finally, conclusions are drawn in Section VIII along with future research directions.

II. Related Work

In this section, we review the state-of-the-art on FL over wireless networks, focusing on existing computation-communication co-design methods and vehicular FL. We also discuss relevant approaches leveraging REMs for wireless resource management, underlining the novelty of our work.

A. User Scheduling for FL in Wireless Networks

In the last few years, many research works have investigated scheduling techniques for FL over wireless networks [18], [19], [20], [21], [22]. In [23], scheduling policies were experimented with by simulating users connected to different access points together with different signal-to-interference-plus-noise ratio (SINR) thresholds. Recent work [24] proposed PALORA, a scheduling method jointly considering fairness with respect to local models, signal-to-noise ratio (SNR) levels, and resource blocks utilization. Along the same lines, paper [16] considered asynchronous updates and a computation-communication tradeoff, and proposed scheduling strategies based on *age-of-information (AoI)* and a *fairness* metric in wireless networks. In [25], scheduling was performed based on data quality metrics to jointly solve a user selection and bandwidth allocation problem. A scheduling approach based on data selection was proposed in [26], where the gradient norm was used to compute a metric that relates learning efficiency with data selected by users for local training. The authors of [27] proposed a scheduling policy for FL over an orthogonal frequency-division multiple access (OFDMA) scheme where scheduling decisions are based on learning accuracy and channel quality. Reference [28] addressed scheduling for over-the-air FL based on local gradient, channel conditions, and energy consumption to improve learning accuracy and convergence.

B. FL in Vehicular Networks

There is a growing interest in machine learning solutions for problems related to vehicular networks, given the increasing need for data-driven algorithms in intelligent transportation systems [29]. Applications examples include wireless resource management [30], traffic flow prediction [31], vehicle-based perception for autonomous driving [32], [33], to name a few.

With respect to the federated training setup, some works have focused on *static* vehicular federated learning [34], [35], with emphasis on scenarios involving parking lots [36] and related issues, like parking space estimation.

Conversely, cooperative training of machine learning models in scenarios involving user mobility, such as vehicular networks, is a key open research topic [37], [38]. In this context, relevant works have considered vehicular FL for, e.g., image classification [39], proactive caching [40], and object detection [41].

Other recent works have investigated the allocation and optimization of network resources in vehicular FL [42], [43], [44], [45], [46]. Specifically, [44] analyzed the impact of vehicular mobility on wireless transmissions, proposing wireless network optimization techniques. The authors of [46] considered cache queue optimization at the edge server, while other related scheduling techniques were proposed in [42], [43]. Reference [45] focused on tuning the number of local iterations based on mobility awareness in relation to short-lived connections with the base stations. The authors of [47] studied the adoption of FL in an IoV scenario where vehicles communicate with 5G base stations, exploiting context information such as cell association and mobility prediction together with the related channel quality for users' scheduling. The authors of [48] jointly scheduled participating vehicles and optimized computational resources. While this approach addresses both learning accuracy and latency/energy restrictions, the decision-making is centralized and may violate the privacy of vehicles. A similar centralized approach was proposed in [17], where the authors select vehicles based on a minimum latency criterion by considering the CPU frequency and the latest known/estimated channel gain. In this work, model compression was also used to further reduce latency and energy consumption.

Despite these efforts, the recent survey [49] remarks that open challenges such as efficient allocation of communication and computational resources, learning-based selection of participating vehicles to enhance training, privacy and security issues, and robustness to noisy training samples are still to be addressed. In fact, none of the works mentioned above combines *mobility patterns* and *radio environmental awareness* to optimize the learning performance of FL, while at the same time optimizing wireless network resources. Our work is the first to jointly consider these aspects under private vehicular mobility which, as we shall see, leads to sizeable advantages over previous solutions and tackles the challenges mentioned in [49] on resource optimization, vehicle selection, and privacy.

C. REMs for Wireless Resource Management

A radio environment map (REM) is a geographic database of average communication quality metrics. In recent years, REMs have been proposed as an effective tool to manage wireless resources [8]. REMs had been advocated for predictive resource allocation in [50] and for handover management in 5G networks in [51]. Recently, in 5G systems with massive multiple-input multiple-output (mMIMO) transmission, REMs have been adopted for energy efficient design [52], inter-cell interference coordination [53], beam management [54], and cell-edge users throughput improvement via dynamic point blanking [55]. Although the use of REMs has been investigated for several resource management applications in wireless communications, the present work is the first to exploit them for network resources optimization in FL and, specifically, in vehicular FL.

III. VREM-FL in a Nutshell

In this article, we are concerned with the optimal resource allocation for FL tasks executed by vehicles that travel within an urban environment and by an edge server that acts as the centralized aggregator of their models. In what follows, we interchangeably use the terms "vehicle" and "client" depending on the role that we would like to emphasize.

A. The Problem: Resource Allocation for Vehicular FL

We aim to minimize the objective cost associated with FL

$$\text{cost} = \text{cost}_{\text{loss}} + \text{cost}_{\text{latency}} + \text{cost}_{\text{channel}} \tag{OBJ}$$

where $\text{cost}_{\text{loss}}$ measures performance of the global model, $\text{cost}_{\text{latency}}$ is the training time, and $\text{cost}_{\text{channel}}$ refers to network resources used by the clients to upload their local models.

We consider a threefold space of intervention. The first aspect is *scheduling clients* during training. Due to limited communication resources, all clients cannot transmit their local models at every learning round. Hence, at each round, a subset of clients is selected to update the global model, and a *scheduling strategy* is utilized to choose which ones.

The second aspect is the *local computation* performed by the scheduled clients. To ensure convergence of an FL algorithm to an accurate global model, the clients must carefully choose the number of descent steps to update their local models.

The third aspect of our design is the *transmission of local models* from the vehicles to the edge server. Sending the local models as soon as the local updates are done (*greedy transmission behavior*) yields the fastest training. However, this strategy ignores the channel status, which depends on mobility and radio channel. It descends that a greedy behavior need not make the best use of the available channel resources. The proposed policy is mobility and channel aware and allows for a more profitable usage of channel resources. It leads to reduced transmission energy and channe usage, thus releasing resources for other users that also need to exploit wireless transmissions.

B. The Solution: VREM-FL

To minimize the cost (OBJ) in a vehicular scenario, we propose Vehicular REM-based Federated Learning (VREM-FL), a co-design that jointly optimizes the threefold decision-making introduced above. An intuitive description of our co-design algorithm is illustrated in Fig. 1. In this section, we provide a high-level overview of how VREM-FL works and defer the detailed explanation to Section VI.

VREM-FL runs before each learning round and consists of three phases, which correspond to the boxes in Fig. 1.

During the *centralized optimization* phase (top box), the edge server (*i.e.,* the orchestrator) computes and broadcasts the number of local steps that all the vehicles should perform to achieve the fastest training convergence. This computation uses a proxy for global convergence assuming that all the scheduled vehicles run the same number of local steps [56]. For this, the server does not need to know the vehicles' local cost functions. This phase is formalized in Section VI-A.

In the *local customization* phase (middle box), each vehicle adjusts the number of local steps based on a local convergence criterion, to trade local training speed for global accuracy. Hence, the vehicle optimizes the communication of the local model by opportunistically delaying its transmission, seeking the best trade-off between the components $\text{cost}_{\text{latency}}$ and $\text{cost}_{\text{channel}}$ in the cost (OBJ). To perform this optimization, the vehicle leverages knowledge of (1) the channel quality via the availability of an estimated REM and (2) its planned trajectory, as explained in Section VI-B. No training is performed at this stage. Instead, the choice of computation and communication is used in the following phase to discern which vehicles are the best candidates to be scheduled at the current round.

In the *centralized scheduling* phase (bottom box), the edge server receives the estimated costs for participating in the round from all the vehicles. These costs depend on the local decision-making performed during the previous phase and are combined with global information available at the server that measures fairness of updates among the vehicles. This, in the form of a combination of AoI and scheduling frequency metrics, ensures appropriate participation of all the vehicles throughout the training. The algorithm executed at the server to make scheduling decisions is described in Section VI-C.

IV. System Model

In this section, we present the setup considered throughout the article. In Sections IV-A and IV-B, we introduce an FL task solved by vehicles that move within an urban environment served by BSs. In Section IV-C, we describe the channel model that vehicles use to assess whether they should join a round and optimize their next transmissions to the server.

A. Vehicular Federated Learning

We consider a set $\mathcal{V} \doteq \{1, \dots, N\}$ of N vehicles that move within an urban environment. The mobility area is served by N_{bs} 5G BSs. As vehicles travel, they collect data to improve tasks of interest for assisted or autonomous driving, such as semantic segmentation for local navigation, pedestrian detection to enhance safety of road users, route optimization based on real-time traffic information, or allocation of wireless resources used by other tasks or vehicles that share the network. Each vehicle $v \in \mathcal{V}$ collects a local, private dataset denoted as $\mathcal{D}_v$.

To efficiently learn complex tasks from locally gathered data, vehicles in $\mathcal{V}$ are connected to an edge server running an FL algorithm. This allows the vehicles to cooperatively learn a common machine learning (ML) model without uploading their collected data to the server, which may be impractical through high data volume or undesirable because of privacy concerns.

In principle, the vehicles may learn a model while they are *static*, *e.g.,* parked. This would ease resource management for FL, such as scheduling of updates that could be performed under constant channel conditions. However, recent work [2], [57] remarked that the sensing capabilities of autonomous vehicles may generate GBs of sensory data per second. The cost for storing such a massive amount of data is high in terms of both energy consumption and storage capacity, and it would be impractical and expensive for the vehicles to run FL tasks after one or multiple trips. This issue urges to run FL tasks *on traveling vehicles* [38], [41], which could destroy training data right after using them and save on storage and energy consumption. However, including mobility into resource management is nontrivial. The aggregator may not know routes of vehicles, which affect the experienced channel quality. On the other hand, the vehicles may not know channel conditions across the whole environment (*e.g.,* a city) and share limited computation resources with other driving-related jobs.

B. Preliminaries on Federated Learning

An FL task is described by the optimization problem

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \mathcal{L}(\theta) \doteq \ell\left(\theta; \{\mathcal{D}_v\}_{v\in\mathcal{V}}\right) + \lambda \left\|\theta\right\|^2 \tag{FL}$$

where θ is the model parameter, ℓ is the loss function that depends on the dataset, $\left\|\theta\right\|^2$ is a regularization term that, in words, penalizes "complex" models, and λ is the regularization weight. In the following, we refer to the total cost $\mathcal{L}(\theta)$ as (regularized) loss. The size of the parameter θ amounts to B [bit]. Problem (FL) is tackled in an iterative fashion that alternates between (1) vehicles updating their local models (*local update*) and (2) the server aggregating all or some local models and sending the updated global model to all vehicles.

We highlight up-front that our proposed co-design algorithms can be tailored to any choice of algorithm used to solve (FL). For instance, VREM-FL can accommodate both synchronous and asynchronous aggregations as explained in Remark 3. This is possible because vehicles can upload their local models within a given time window, and the global model is updated only after all models are received or the deadline expires. Nonetheless, for the sake of exposition and to ground the discussion, in the rest of this article we will assume that the vehicles train their local models via gradient descent (GD) or stochastic gradient descent (SGD) and that the edge server runs FedAvg [15]. This choice is motivated by the simplicity and the popularity of this FL algorithm. Other learning and aggregation schemes, such as FedDrop [58] or FedLin [5], require to minimally adjust our algorithms as described in Remark 4. A high-level snippet of vanilla FedAvg (without client scheduling) is provided in Algorithm 1, where all clients perform H SGD steps for each local update. We refer to a cycle composed of local updates and global aggregation (Lines 2 to 6) as *(learning) iteration* or *(learning) round*, whose duration coincides with the time interval between two consecutive updates of the global model, and the edge server runs FedAvg for T rounds. We denote by θ^t the global parameter at iteration t, and by $\theta^t_{v,t+H}$ the locally updated parameter of vehicle v during round t (before transmission to the aggregator).

As commonly assumed in the literature [23], [24], each learning round has a deadline after which global aggregation is executed, regardless of the status of local updates, and a new round begins afterward. Time is slotted into slots of duration τ [s], which we consider the finest granularity to allocate resources in a time-varying fashion. Among the steps of Algorithm 1, our present work addresses local training (Line 4) and transmission of local models from vehicles to the server (Line 5). Hence, we further assume that at every round the vehicles have a deadline of K_{max} [s] to update their local models and to upload them to the server, corresponding to $\lfloor K_{\text{max}}/\tau \rfloor$ time slots. The set of time slots available during iteration t, when we allocate resources for training and transmission of each vehicle, is denoted by $\mathcal{K}_t$. In the following, we will refer to the local training performed by the vehicles as *computation* to highlight the allocation of computational resources.

Algorithm 1: Vanilla FedAvg [15]

```
Input: Loss $\mathcal{L}$, rounds $T$, parameter $\theta^0$, local steps $H$.
Output: Learned parameter $\theta^t$.
1 for $t = 0, 1, \ldots, T$ do
2     server broadcasts global parameter $\theta^t$ to vehicles;
3     foreach vehicle $v \in \mathcal{V}$ do
4         train local model: $\theta^t_{v,t+H} \leftarrow \text{SGD}(\theta^t, H; \mathcal{D}_v)$;
5         transmit local parameter to server;
6     server aggregates local models:
        $\theta^{t+1} \leftarrow \sum_{v\in\mathcal{V}} w_v \theta^t_{v,t+H}$;
```

C. Radio Environment and Bitrate

The area served by the BSs features a heterogeneous channel quality depending on the network coverage at different geographical locations. To assess channel quality across the urban environment, we use an REM. This is a database that links a (quantized) geographical location to the (estimated) value of some channel quality metric. In this work, we are interested in the average bitrate associated with a location $x \in \mathbb{R}^2$. As so, we consider an REM that contains information about the SINR experienced on average at each location. Given a wireless transmission setting and a geographical location x, the value of the SINR associated with x contained in the REM can be used to infer the expected average bitrate experienced by a user located at x. This information is crucially used for resource orchestration by VREM-FL. In fact, vehicles may experience different channel quality depending on their real-time location. Formally, we express the location-to-bitrate map as follows. Given a vehicle v located at x^v_t at time t, the corresponding estimated average bitrate h^v_t is

$$h^v_t = \beta(\gamma(x^v_t), \eta^v_t), \tag{1}$$

where $\gamma(\cdot)$ is the location-to-SINR map encoded by the REM, $\beta(\cdot)$ is the SINR-to-bitrate map, and η^v_t is the bandwidth used by vehicle v at time t.

In practice, we assume that an REM γ of the environment is computed *a priori* via some estimation technique [59], [60], [61] and is stored at the BS, which broadcasts this information to the vehicles taking part in FL. Because REMs are approximately constant across time [8], [9], they are sent to the vehicles only once during FL training. When a handover occurs and a vehicle enters coverage area of a new BS, the latter sends its corresponding REM to the vehicle, updating the channel information that the vehicle uses to allocate computation and transmission resources. We assume that the routes traversed by the vehicles are planned in advance (at least partially) so that they know their respective trajectories in the near future. Specifically, at time t, vehicle v knows the locations it is about to traverse over the next D time instants, denoted by $x^v_t, \ldots, x^v_{t+D}$, for some time horizon D. This information is converted to estimated bitrate values $h^v_t, \ldots, h^v_{t+D}$ for a vehicle's trajectory, where $h^v_k = \beta\left(\gamma(x^v_k), \eta^v_t\right)$ as per (1). This information is used in our resource-allocation algorithms as detailed in Section V.

V. Problem Formulation

We now formalize the computation-scheduling co-design problem at the core of our contribution. We first define the design parameters for the considered decision-making (Section V-A), and then formally write the objective function (OBJ) along with the overall optimization problem (Section V-B).

A. Design Parameters

Given problem (FL) and an algorithm to solve it (FedAvg), our space of intervention is the threefold decision-making associated with the algorithm workflow discussed in Section III-A.

1) Scheduling: First, we design a *scheduling strategy* to select the vehicles that participate in each learning iteration. Scheduling is needed because the total amount of vehicles involved in an FL task is typically large and cannot be handled at once due to the limited communication resources available at the BSs. It descends that only a (small) fraction of the vehicles can be simultaneously served through the available bandwidth to ensure an acceptable quality of service. We denote the subset of vehicles that participate in round t by $\mathcal{V}^S_t \subset \mathcal{V}$ and the maximum number of vehicles that can be scheduled in round t by $M^S_t < N$, with $|\mathcal{V}^S_t| \leq M^S_t$.

2) Computation: For each vehicle scheduled in a learning round, we consider two aspects for co-design. First, we allocate the amount of *computation at the vehicle*, that is, the number of descent steps (of GD or SGD) that the vehicle performs during that round to train the local model. Through a careful choice of the number of local steps, we can effectively trade convergence speed of FedAvg for the quality of the final global model. For each time slot $k \in \mathcal{K}_t$ available during learning iteration t, we denote by $a^v_k \in \{0, 1\}$ the *computation decision* of vehicle v for slot k: if $a^v_k = 1$, it means that v performs a batch of local steps during slot k, otherwise no computation is carried out in that slot. The total number of time slots used by vehicle v for local model update during learning round t is denoted by $T^v_{\text{cpu},t} \doteq \sum_{k\in\mathcal{K}_t} a^v_k$. To make the training meaningful, we allocate at least $T^{\text{min}}_{\text{cpu}} \geq 1$ slots for computation at every round.

3) Communication: After a vehicle has updated its local model, we optimize the *transmission from vehicle to server*. Although the training time for FL is trivially minimized if local models are immediately transmitted, in this work we are additionally interested in an efficient allocation of network resources. This allows us to reduce both channel bandwidth occupancy and transmission energy needed for FL. We assume that all vehicles have constant transmission power, so that optimizing for communication resources is equivalent to transmitting where the SINR is high. We denote by $b^v_k \in \{0, 1\}$

the *transmission decision* of vehicle v in slot k. If $b_k^v = 1$, then v uses time slot k to transmit its local model to the server; otherwise, no transmission occurs during slot k. The total time vehicle v takes to upload its local model during round t is denoted by $T_{\text{tx},t}^v \doteq \sum_{k\in\mathcal{K}_t} b_k^v$. Further, we denote by K_t^v the total time elapsed from the beginning of local training to reception of the updated local model at the server, which we name *round latency* of vehicle v in learning round t.

B. Optimization Problem

We aim to jointly optimize the three costs in (OBJ). As discussed above, optimizing computation and scheduling resources reduces training loss $\text{cost}_{\text{loss}}$ and latency $\text{cost}_{\text{latency}}$ of the FL task, while optimizing communication resources translates into efficient channel usage $\text{cost}_{\text{channel}}$ during training.

With a slight abuse of notation, we express $\text{cost}_{\text{loss}}$ as a function of both computation and scheduling (note that it also depends on the final parameters learned by FedAvg, θ^T):

$$\text{cost}_{\text{loss}} = \mathcal{L}\left(\left\{\mathcal{V}_t^S, \{a_t^v\}_{v\in\mathcal{V}_t^S}\right\}_{t\in\mathcal{T}}; \theta^T\right), \tag{TL}$$

where $a_t^v \doteq \{a_k^v\}_{k\in\mathcal{K}_t}$ denotes all computation decisions of vehicle v throughout round t and $\mathcal{T} \doteq \{1,\dots,T\}$ gathers all learning rounds. The term $\text{cost}_{\text{latency}}$ is upper bounded by TK_{max} but varies depending on 1) allocation of computation and communication resources, and 2) the channel quality experienced by the scheduled vehicles. We formalize this as

$$\text{cost}_{\text{latency}} = K\left(\left\{\{a_t^v, b_t^v; h_t^v\}_{v\in\mathcal{V}_t^S}\right\}_{t\in\mathcal{T}}\right), \tag{OL}$$

where $b_t^v \doteq \{b_k^v\}_{k\in\mathcal{K}_t}$ and $h_t^v \doteq \{h_k^v\}_{k\in\mathcal{K}_t}$ denote all transmission decisions of vehicle v and bitrate values experienced by v during round t, respectively. Finally, we quantify the channel usage as the total time (number of time slots) the vehicles reserve channel bandwidth to upload their local models to the server. This quantity amounts to summing transmission times $T_{\text{tx},t}^v$ across all scheduled vehicles and all learning rounds. Because the transmission time $T_{\text{tx},t}^v$ is defined by transmission decisions b_t^t, which in turn depend on the bitrate h_t^v experienced by vehicle v at round t, we express $\text{cost}_{\text{channel}}$ as

$$\text{cost}_{\text{channel}} = \sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}_t^S} T_{\text{tx},t}^v\left(b_t^v; h_t^v\right). \tag{CU}$$

The total cost (OBJ) addressed in our co-design amounts to

$$\begin{aligned}\text{cost}\left(\mathcal{V}^S, a, b; h, \theta^T\right) = &\ \mathcal{L}\left(\mathcal{V}^S, a; \theta^T\right)\\ &+ (1-w_{\text{tx}})K\left(a, b; h\right)\\ &+ w_{\text{tx}}\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}_t^S} T_{\text{tx},t}^v\left(b_t^v; h_t^v\right)\end{aligned} \tag{2}$$

where $\mathcal{V}^S \doteq \{\mathcal{V}_t^S\}_{t\in\mathcal{T}}$ denotes the full client schedule, $a \doteq \{a_t^v\}_{v\in\mathcal{V}_t^S, t\in\mathcal{T}}$, $b \doteq \{b_t^v\}_{v\in\mathcal{V}_t^S.t\in\mathcal{T}}$, and $h \doteq \{h_k^v\}_{k\in\mathcal{K}_t, t\in\mathcal{T}}$ gather all computation and transmission decisions, and bitrate values associated with scheduled vehicles across rounds. The weight $w_{\text{tx}} \in [0,1]$ trades $\text{cost}_{\text{latency}}$ (OL) for $\text{cost}_{\text{channel}}$ (CU). If $w_{\text{tx}} = 0$, only the training time is penalized in (2); if $w_{\text{tx}} = 1$, latency is neglected and usage of network resources is discouraged. The learning cost (TL) and latency cost (OL) jointly depend on all scheduled clients, while the resource cost (CU) decomposes linearly across those.

TABLE I: List of symbols used in this article.

$\mathcal{V}$	set of vehicles
B [bit]	size of model parameter θ
τ [s]	duration of one time slot
K_{max} [s]	maximum latency of every learning iteration
$T_{\text{cpu}}^{\text{min}}$	minimum number of computation slots at every iteration
$\mathcal{K}_t$	set of time slots available for iteration t
s_v	local steps that vehicle v runs in one time slot
a_k^v	computation decision of vehicle v for slot k
b_k^v	transmission decision of vehicle v for slot k
$T_{\text{cpu},t}^v$	number of computation slots of vehicle v in iteration t
$T_{\text{tx},t}^v$	number of transmission slots of vehicle v in iteration t
h_k^v [bit/s]	bitrate experienced by vehicle v in slot k
B_t^v [bit]	bits that vehicle v transmits during iteration t
K_t^v [s]	latency of vehicle v in iteration t
$\mathcal{V}_t^S$	set of vehicles scheduled for iteration t
M_t^S	maximum number of vehicles scheduled for iteration t

Equipped with the mathematical definition (2) of (OBJ), we are now ready to formalize the computation-scheduling co-design problem tackled in the rest of this work.

Problem 1 (Optimal computation-scheduling co-design for vehicular FL)**.** Given (i) a set of vehicles $\mathcal{V}$, (ii) an REM of the environment γ, (iii) an FL algorithm, (iv) model parameters $B, K_{\text{max}}, T_{\text{cpu}}^{\text{min}}$, find (1) a vehicle schedule $\mathcal{V}^S$, (2) computation decisions a, (3) transmission decisions b, so as to optimize FL training and transmission resources:

$$\begin{aligned}\text{(P)}\quad &\underset{\mathcal{V}^S, a, b}{\arg\min} && \text{cost}\left(\mathcal{V}^S, a, b; h, \theta^T\right) && && \text{(Pa)}\\ &\text{subject to} && K_t^v \le K_{\text{max}} && \forall v\in\mathcal{V}_t^S, \forall t\in\mathcal{T}, && \text{(Pb)}\\ & && T_{\text{cpu},t}^v \ge T_{\text{cpu}}^{\text{min}} && \forall v\in\mathcal{V}_t^S, \forall t\in\mathcal{T}. && \text{(Pc)}\end{aligned}$$

In words, constraint (Pb) ensures that each round ends within the pre-assigned deadline, and constraint (Pc) requires the scheduled vehicles to perform a minimal number of descent steps. The meaning of all symbols is provided in Table I. In the next section, we propose co-design algorithms to solve Problem 1 that crucially rely on vehicular mobility and the REM to estimate the channel quality experienced by vehicles.

VI. Algorithms for Co-Design

The computation-scheduling co-design problem (P) requires designing both local operations performed by vehicles (computation and transmission) and global scheduling decisions the edge server makes. To efficiently tackle it, we propose a cascade procedure executed at the beginning of each round t, involving three phases (split between edge server and vehicles).

Centralized optimization: the edge server computes the optimal number of local steps H_t^* to be performed by the vehicles in round t and broadcasts them this value.

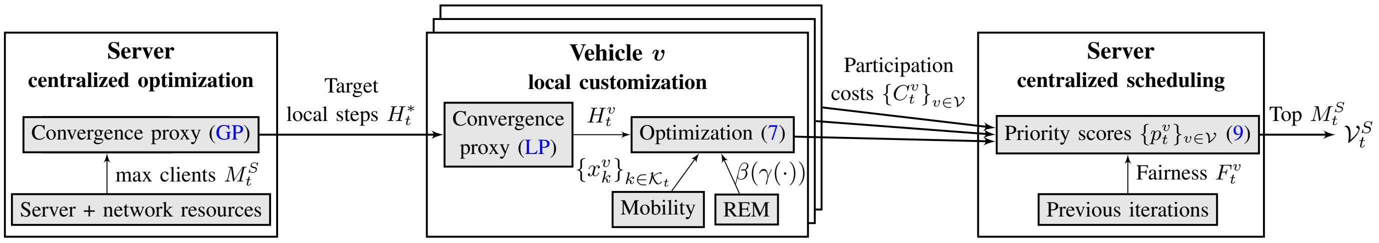


Fig. 2: **Workflow of VREM-FL.** At the beginning of learning iteration t, the server runs *centralized optimization* and transmits the globally optimal local steps to the clients. Then, each vehicle locally runs *local customization* to fine-tune computation and communication resources to be used in the round. Finally, the server performs *centralized scheduling* taking into account both feedback from the vehicles and global participation information.

Local customization: each vehicle refines its local steps and transmission (time slots to send the local model update), and sends its cost C_t^v to participate in round t to the server.

Centralized scheduling: the server receives all the costs C_t^v and uses both such local information and global knowledge about the fairness of updates to select the subset of vehicles that will actually take part in the model update at round t.

Figure 2 provides a schematic representation of the workflow of VREM-FL with the three phases summarized above. In the following, the operations of VREM-FL are described in detail.

A. Centralized Optimization

In this phase, the edge server first sets the maximum number M_t^S of vehicles that are allowed to participate in round t. Then, the server computes an approximate number of local steps to be performed by the scheduled vehicles in order to speed up the training. To evaluate at runtime the relation between the number of local steps at the vehicles and the training time, we use the proxy for *global FL convergence* proposed in [56]. This proxy assumes that M clients are scheduled at every round, and that each of them performs H local steps at every local update. The proxy is expressed in [56, Eq. (1)] as

$$T\epsilon \approxeq \Gamma(H, M) \doteq \frac{C}{H} + \left(1 + \frac{1}{M}\right) H \tag{GP}$$

where ϵ is the estimated accuracy in T rounds, *i.e.*, $\mathcal{L}(\theta^T) \leq \epsilon$, and C is a constant that depends on the data distribution.

The server homogeneously chooses H_t^* as the minimizer of the proxy (GP) with respect to H, setting $M = M_t^S$ and assuming that all vehicles run H_t^* local steps at all iterations:

$$H_t^* = \underset{H}{\arg\min} \quad \Gamma(M_t^S, H). \tag{4}$$

This first unconstrained subproblem is convex because the objective cost (GP) is convex in H. Its solution is

$$H_t^* = \sqrt{\frac{C}{\left(1 + \frac{1}{M_t^S}\right)}}. \tag{5}$$

This reveals how the optimal number of local steps H_t^* depends on the scheduled clients. It is a strictly increasing and concave function of M ($M \geq 1$) that saturates to $\sqrt{C}$ for $M \to +\infty$ (horizontal asymptote).

After computing H_t^*, the server broadcasts this value to all the vehicles for the second phase.

B. Local Customization

In this phase, each vehicle independently executes a local subroutine to allocate slots for computation and communication. This allocation attempts to optimize (i) convergence of local training and (ii) channel utilization. The resulting number of local steps is temporarily stored by the vehicles, which use it later to perform the local model update in case they are actually scheduled. If a vehicle has a means to (efficiently) estimate the loss gradient, it first refines the number of local steps H_t^* communicated by the server (*computation refinement*) based on a proxy for local convergence, obtaining a new number of local steps H_t^v, and then it optimizes for transmission (*communication optimization*). Instead, if evaluating the gradient is expensive, the vehicle skips the computation refinement at this stage and sets the number of local steps as $H_t^v = \max\left\{H_t^*, s_v T_{\text{cpu}}^{\min}\right\}$.

1) Computation refinement: To optimize the local updates of the vehicle, we consider a local proxy that jointly keeps into account the individual client convergence properties and the global recommendation H_t^* indicated by the server. As such, the proxy is obtained as the sum of multiple terms. First, we consider a "convergence" proxy $\Theta_t^v(H_t^v)$ related to the local optimality gap, which bounds the client deterministic gradient norm after H_t^v local steps of gradient descent starting from the global parameter θ^t:

$$\left\|\nabla\ell^v\left(\theta_{v,t+H_t^v}^t\right)\right\| \leq \Theta_t^v(H_t^v) \doteq \left\|\nabla\ell^v(\theta^t)\right\|\left(1 - \kappa_v^{-1}\right)^{H_t^v - 1}. \tag{LP}$$

The constant $\kappa_v > 0$ is the condition number associated with the local loss ℓ^v. The proxy (LP) is based on quadratic cost functions and we derive it explicitly in Appendix A. The proxy (LP) is motivated by the fact that the optimality gap $\|\theta - \theta^*\|$ is in general proportional to the gradient norm, but we have only access to the gradient, while θ^* is unknown. It can be computed by each client based on their local cost only and does not take into account the distributed nature of the FL problem. Hence, during this step, the vehicle v refines its computation by solving the optimization problem

$$H_t^v = \underset{H \in \mathbb{N}}{\arg\min} \quad \Theta_t^v(H) + \frac{\rho_1 H}{\|\nabla\ell^v(\theta^t))\|} + \rho_2\left(H - H_t^*\right)^2 \tag{6a}$$

$$\text{subject to} \quad H \geq s_v T_{\text{cpu}}^{\min}. \tag{6b}$$

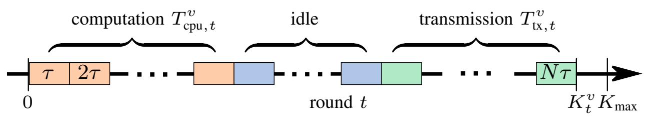

computation $T^v_{\text{cpu},t}$
idle
transmission $T^v_{\text{tx},t}$
τ
2τ
$N\tau$
0
round t
K^v_t
$K_{\max}$

Fig. 3: **Behavior of a selected client at each round.** If a client v is selected by the server, it goes through the following steps. Each slot has duration τ [s]. At first, the client performs some local (S)GD steps (*computation*) according to its computation allocation. Eventually, it transmits its updated local model (*transmission*) after time K^v_t and within the maximum allowed latency $K_{\max}$, possibly waiting for some time slots (*idle*) to enjoy a better channel quality.

The second addend in (6a) accounts for how close the vehicle is to a local minimum, forcing a few steps if the vehicle has (locally) almost converged and many steps if it is still far from convergence. The third addend encourages the chosen local steps H^v_t to be close to the target H^*_t computed by the server. The cost (6a) is convex and grows unbounded as $H \to +\infty$, so that problem (6) can be efficiently solved by a linear search. The number of computation slots is then set as $T^v_{\text{cpu},t} = \lceil H^v_t / s_v \rceil$. For the sake of simplicity, we require that the vehicle allocates all slots for computation at the beginning of the round. Formally, this means $a^v_k = 1$ for $k = 1, \dots, T^v_{\text{cpu},t}$ and $a^v_k = 0$ for $k > T^v_{\text{cpu},t}$ for all time slots $k \in \mathcal{K}_t$. The slots with $a^v_k = 1$ are the leftmost "computation" slots in Fig. 3.

2) Communication optimization: In this step, the vehicle chooses the time slots to transmit its local model to the server, adjusting the computation slots if needed. For simplicity, we require the vehicle to allocate a batch of consecutive slots also for transmission. Formally, the communication decisions for round t are $b^v_k = 0$ for $k = 1, \dots, \bar{k}_1$, and $b^v_k = 1$ for $k = \bar{k}_1 + 1, \dots, \bar{k}_2$, for some $\bar{k}_1 \geq T^v_{\text{cpu},t}$ and $\bar{k}_2 > \bar{k}_1$. We denote the set of such transmission patterns by $\mathcal{B}^v_t(T^v_{\text{cpu},t})$. The allocation of slots for computation and communication is depicted in Fig. 3, where the slots with $b^v_k = 1$ are the rightmost "transmission" slots. The vehicle considers an allocation feasible if it estimates that its local model (B [bit]) can be uploaded before the deadline $K_{\max}$ using the transmission slots. To choose the communication pattern, given $T^v_{\text{cpu},t}$ slots of computation, the vehicle attempts to solve the following optimization problem:

$$b^v_t = \underset{b \in \mathcal{B}^v_t(T^v_{\text{cpu},t})}{\arg\min} \quad C^v_t \doteq (1 - w_{\text{tx}}) K^v_t + w_{\text{tx}} T^v_{\text{tx},t} \tag{7a}$$

$$\text{subject to} \quad K^v_t \leq K_{\max}, \tag{7b}$$

$$B^v_t \geq B. \tag{7c}$$

The cost C^v_t in (7a) is designed to trade round latency K^v_t for channel occupancy $T^v_{\text{tx},t}$, according to the overall objective cost (2). To solve (7), vehicular mobility in conjunction with the REM plays a crucial role. Each vehicle v inspects the REM to estimate the available bitrate h^v_k in each slot $k \in \mathcal{K}_t$ of round t according to (1) and, in turn, the number of time slots needed to upload the model in round t. For example, suppose the vehicle is about to travel close to a well-served area (*e.g.,* a main urban road). In that case, it will likely experience a high channel quality and thus take a short time for the upload, whereas, if it approaches a weakly served area (*e.g.,* a tunnel), it will predict poor channel conditions. It might even declare (7) infeasible, giving up on joining the learning round. Figure 4 pictorially represents the mobility-aware REM-based evaluation of the cost (7a). In this figure, for case 1, transmission slots are allocated in a greedy fashion, whereas for case 2, the vehicle defers the transmission of the local model, waiting for the channel quality to improve. This trades some extra delay (higher round latency K^v_t) for a better usage of channel resources (shorter upload time $T^v_{\text{tx},t}$).

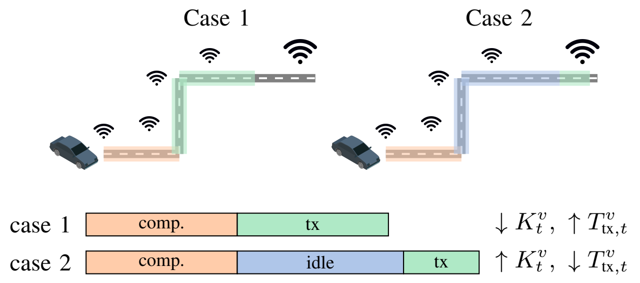

Case 1
Case 2
case 1
comp.
tx
$\downarrow K^v_t$, $\uparrow T^v_{\text{tx},t}$
case 2
comp.
idle
tx
$\uparrow K^v_t$, $\downarrow T^v_{\text{tx},t}$

Fig. 4: **REM-based solution for communication optimization.** Two evaluations of the objective cost (7a). In Case 1, transmission occurs under poor channel conditions (low bitrate) and takes long time. On the contrary, in Case 2, idle slots delay communication until when the vehicle travels through a well served area (high bitrate), so that transmission time is shorter.

Computation and communication decisions for the round are set by first solving (6) – if possible – to fine-tune the number of computation slots, and then (7) until a feasible computation-communication pattern is found or the whole allocation is declared infeasible. Algorithm 2 summarizes the workflow of this phase. Eventually, each vehicle v transmits its estimated cost C^v_t for participation in round t to the server, to inform the latter on the expected benefit of scheduling v for the present round. We use the convention that, if vehicle v cannot find a feasible allocation for round t (*i.e.,* problem (7) turns out to be infeasible), it will communicate an infinite cost C^v_t, see Line 6.

Algorithm 2: Subroutine `customize_local`

```
Input: Local proxy $\Theta^v$, minimum computation $T^{\min}_{\text{cpu}}$,
       maximum round latency $K_{\max}$, target steps $H^*_t$.
Output: Cost $C^v_t$, computation and communication
        decisions $a^v_t$ and $b^v_t$ for round $t$.
1  if can efficiently estimate $\nabla \ell^v_t$ then
2      compute $H^v_t$ as the solution to (6);
3  else
4      set $H^v_t \leftarrow \max\{H^*_t, s_v T^{\min}_{\text{cpu}}\}$;
5  set $T^v_{\text{cpu},t} \leftarrow \lceil H^v_t / s_v \rceil$;
6  set $C^v_t \leftarrow +\infty$;
7  repeat
8      compute $b^v_t$ as the solution to (7);
9      if problem (7) is feasible then
10         set $C^v_t \leftarrow C^v_t(b^v_t)$;
11         store computation $T^v_{\text{cpu},t}$ and communication $b^v_t$;
12         break;
13     else
14         set $T^v_{\text{cpu},t} \leftarrow T^v_{\text{cpu},t} - 1$;
15 until $T^v_{cpu,t} < T^{min}_{cpu}$;
```

Remark 1 (Computation refinement for scheduled vehicle). If a vehicle cannot evaluate the cost (6a) in reasonable time and skips the refinement (6), scheduled vehicles may refine their local steps *after* they have been scheduled. To this aim, they can compute or approximate the gradient $\nabla \ell^v(\theta^t)$ at the beginning of the round, *e.g.,* by running a few descent steps, and then solve for H_t^v according to (6) with the (approximate) gradient just computed. Then, they complete the local update by running SGD until they reach H_t^v total steps.

Remark 2 (Reducing latency *vs.* network resources). Optimization (7) makes both the vehicles reduce their respective latency K_t^v for round t concerning the deadline K_{max} and the server aware of their expected latency. This favors those vehicles that are likely to send their local models in short time and in turn speeds up the whole training. We experimentally demonstrate this via ablation studies and comparisons against scheduling benchmarks in Section VII. In particular, by tuning the weight w_{tx} in (7a), a system designer can encourage a short training (small w_{tx}) or a frugal usage of network resources (large w_{tx}).

C. Centralized Scheduling

After the server receives information from all vehicles about their (predicted) cost for the round, the vehicles are scheduled based on both this cost, that measures the training performance, and on fairness metrics such as the AoI and scheduling frequency, accounting for the learning accuracy of the global model. In particular, drawing inspiration from [16], we define the fairness F_t^v for vehicle v at round t as

$$F_t^v \doteq \frac{1}{\phi_t^v} + A_t^v \tag{8}$$

where ϕ_t^v is the scheduling frequency of vehicle v before round t and A_t^v is its AoI at the server. To schedule the participating vehicles, the server assigns a *priority score* p_t^v to each vehicle v for round t:

$$p_t^v \doteq \begin{cases} \dfrac{1}{C_t^v} + w_A F_t^v & \text{if } C_t^v < +\infty \\ -1 & \text{if } C_t^v = +\infty. \end{cases} \tag{9}$$

The weight w_A in (9) should be chosen so as to strike a balance between high-performing vehicles, which may significantly reduce the objective cost (2) in the short run, and overall training in the long run that needs to gather information from all vehicles to eventually learn an accurate global model.

Formally, the vehicles with the highest priority scores are scheduled, according to the following optimization problem:

$$\mathcal{V}_t^S = \underset{\mathcal{V}^S \subseteq \mathcal{V}}{\arg\max} \quad \sum_{v \in \mathcal{V}^S} p_t^v \tag{10a}$$

$$\text{subject to} \quad |\mathcal{V}^S| \leq M_t^S. \tag{10b}$$

According to our convention described in Section VI-B, the clients that communicate infeasible participation are assigned a negative priority score as per (9), which automatically excludes them from the round according to maximization of (10a).

The set $\mathcal{V}_t^S$ contains the vehicles that the server schedules for transmitting their local updates in the current round t. The workflow of VREM-FL is provided in Algorithm 3.

Remark 3 (VREM-FL supports asynchronous aggregation). VREM-FL assumes that the aggregator updates the global model after all scheduled vehicles have transmitted their updates or the deadline K_{max} expires. This means that the aggregation scheme can be *asynchronous*. In fact, if vehicles train and upload local models with the settings output by `customize_local` (respectively a_t^v and b_t^v), the timing of training and transmission will be in general different for each vehicle, yielding asynchronous reception of local models at the aggregator. For this, using knowledge of the REM and mobility pattern at the vehicle, as we propose, is essential; broadly speaking, transmitting when the channel is good is advantageous regardless of aggregation schemes. This behavior is observed in our experiments, where moreover not all scheduled vehicles transmit their local models in time. However, while this happens rarely when using VREM-FL, which leverages knowledge of the channel quality experienced by vehicles, other scheduling strategies incur several missed updates that contribute to degrading learning and wasting resources. See Fig. 8c for our experimental comparison.

Algorithm 3: VREM-FL

```
Input: Global proxy $\Gamma$, local proxy $\Theta^v$, maximal round
       latency $K_{\text{max}}$, minimal computation slots $T_{\text{cpu}}^{\text{min}}$,
       maximal number of scheduled vehicles $M_t^S$ .
Output: Scheduled vehicles $\mathcal{V}_t^S$.
centralized optimization (at the edge server):
1 compute $H_t^*$ as the solution to (4);
2 broadcast $H_t^*$ and maximal latency $K_{\text{max}}$ to vehicles;
local customization (at the vehicles):
3 foreach vehicle $v \in \mathcal{V}$ do
4     $C_t^v \leftarrow$ customize_local($\Theta^v, T_{\text{cpu}}^{\text{min}}, K_{\text{max}}, H_t^*$);
5     send $C_t^v$ to server;
centralized scheduling (at the edge server):
6 foreach vehicle $v \in \mathcal{V}$ do
7     compute $p_t^v$ as per (9);
8 populate $\mathcal{V}_t^S$ according to (10);
9 return $\mathcal{V}_t^S$.
```

Remark 4 (Extension to other FL algorithms). The convergence proxies (GP) and (LP) are the only elements that depend on the algorithms used to solve (FL) and they can be adjusted to use VREM-FL with other local training or aggregation schemes.

D. Complexity Analysis of VREM-FL

A strength of VREM-FL is its light computational and communication requirements, which can accommodate a large number of vehicles with modest computational power onboard.

1) Centralized optimization: During this phase, the edge server computes the target global local steps H_t^* as (5), which has complexity $O(1)$. The required communication corresponds to broadcasting this value once to all vehicles.

2) Local customization:

a) Computation refinement: The first part of the second phase, if executed, requires each vehicle to solve problem (6), which is the minimization of a scalar submodular function on $\mathbb{N}$ and can be solved via linear search.

b) Communication optimization: In the second part of the second phase, each vehicle solves problem (7) that requires the evaluation of the cost function (7a) at least $|\mathcal{K}_t| - T_{\text{cpu},t}^v$ times, where $T_{\text{cpu},t}^v$ is the number of slots allocated for local training from either problem (6) or the received H_t^*. In the worst case, when (7) is infeasible, the vehicle performs about

$(T^v_{\text{cpu},t}+1)|\mathcal{K}_t|-T^v_{\text{cpu},t}/2$ evaluations of (7a), which is linear with $|\mathcal{K}_t|$. In our realistic experiments, the vehicles almost always solved (7) in a few attempts, resulting in a low computational requirement. Each vehicle transmits to the server its cost C^v_t.

3) Centralized scheduling: The edge server computes one priority score per vehicle and selects the participating ones via problem (10). Both operations are linear with the number of vehicles. A fast implementation computes the priority (9) of each vehicle v as soon as its cost C^v_t is received, and keeps the vehicles ordered by priority with `InsertionSort`. In this case, the computational complexity to solve (10) is $O(1)$.

VII. Numerical Results

We perform FL experiments with both synthetic and real-world data. We address vehicular mobility by both generating trajectories of vehicles with a realistic simulator and using real-world mobility data. Without loss of generality, we set a constant bandwidth $\eta^v_t \equiv \eta$, identical for all vehicles and learning iterations. In Section VII-A, we describe the urban environment, mobility data, and REM generation. In Section VII-B, we present the benchmarks compared with VREM-FL. In Section VII-C, we showcase results for a linear regression model on a least-squares problem with synthetic data. This allows us to conduct ablation studies that isolate the effects of several features of VREM-FL and highlight their benefits. In Section VII-D, we use VREM-FL to train a deep neural network model for semantic segmentation, a task of interest for assisted and autonomous driving. For this experiment, we use the real-world dataset `ApolloScape` [14] and both simulated and real-world mobility data.[1]

A. Mobility and Urban Radio Environment Generation

We implement the simulations in Python. We use the map of Padova, Italy, from `OpenStreetMap` [64] and use SUMO [10] to simulate $1,000$ vehicles that move across the city for one hour, discarding the first ten minutes of simulation to let the road map populate with a sufficiently large number of vehicles. For the second experiment, we use a real-world mobility dataset that we describe in Section VII-D4.

On top of the city map, BSs are deployed with an inter-site distance of 600 m, according to typical 5G deployment criteria. Average SINR values and the corresponding bitrates have been obtained through the Matlab 5G NR link-level simulation tool [65]. SINR values are calculated considering (i) the transmission power of vehicles, (ii) physical settings of the 5G NR, (iii) propagation models, and (iv) interference and noise power. We set the transmission power of vehicles to 23 dBm. Without loss of generality, we assume to allocate (in the frequency domain) a fixed number of 10 resource blocks for data transmission. The carrier frequency is set to 3.5 GHz. The sub-carrier space and the resulting size (in the frequency domain) of each single radio resource block are respectively set to 30 kHz and 360 kHz. Hence, the per-client bandwidth for 10 resource blocks is $\eta = 3.6$ MHz (see (1)). The antenna height of BSs and vehicles is set to 25 m and 1.5 m, respectively. Path loss parameters are set according to the urban microcell scenario, as defined in the TR 38.901 specification of 3GPP.

[1]Code for simulations available at https://github.com/lucaballotta/vrem-fl.

TABLE II: Parameters used in the two experiments on linear regression (Section VII-C) and deep learning (Section VII-D). Curly brackets indicate multiple trials. Bold font indicates the best choice w.r.t. (2).

	Linear regression	Deep learning	
		SUMO	roma/taxi
B	400 B	42.3 MB	
learning horizon	30 rounds	1 h	
τ	1 s	1 s	
$K_{\max}$	100 s	2 min	
$T^{\min}_{\text{cpu}}$	1	1	
s_v	1 (GD)	3 (SGD with bz $= 32$)	
$\lvert\mathcal{V}\rvert$	1000	1000	92
M^S_t	30	$\{5, 10, \mathbf{15}, 20, 25\}$	10
C in (GP)	200	1000	
(ρ_1, ρ_2) in (6a)	$(0.001, 1)$	$(1, 0.02)$	
w_{tx} in (7a)	$\{0, \mathbf{0.5}, 1\}$	0.9	
w_A in (9)	0	0.01	{0.01, **0**}

The slow-fading (shadowing) is added to the coverage area of each BS following the 3GPP guidelines [66], with a de-correlation distance of 25 m and standard deviation of 6 dB, which are common values for urban environments [67]. Finally, the noise figure, used to derive the noise power, is set to 6 dB. The bitrate corresponding to a given SINR average value – *i.e.,* the map $\beta(\cdot)$ in (1) – is obtained according to the physical uplink shared channel (namely 5G NR PUSCH) throughput experienced in a 5G New Radio link [68], [69], [70] for that value of SINR. The average throughput is calculated through the Matlab 5G NR link-level simulation tool implementing the 3GPP NR standard [65], assuming that vehicles experience a given average SINR value during the time slot τ of 1 s.

B. Scheduling Benchmarks

We compare VREM-FL against four scheduling benchmarks:

Round robin: Vehicles are chosen by the scheduler based on a round robin policy, *i.e.,* in a cyclic order.

FedAvg [15]: Vehicles are chosen by sampling them randomly with equal probability (uniform random variable).

Fairness [16]: This scheme optimizes the metric proposed in [16] and is equivalent to computing scheduling priority as $p^v_t = F^v_t$. A crucial difference between this method and VREM-FL is that the cost C^v_t is not used and thus the two cases in (9) are indistinguishable. That is, even if a vehicle could communicate that its participation is infeasible, "Fairness" neglects this information and uses only fairness-related information F^v_t known to the scheduler.

Centr-SNR [17]: This algorithm is adapted from [17], where clients are scheduled in a centralized way based on the uplink channel gain (SNR) reported to the server. To draw a fair comparison, we select the vehicles with the best estimated bitrate at the beginning of each learning round. We use this scheme for the second experiment in Section VII-D.

C. First Experiment: Linear Regression on Synthetic Data

We first address a least squares problem with a linear regression model to illustrate the functioning of our proposed

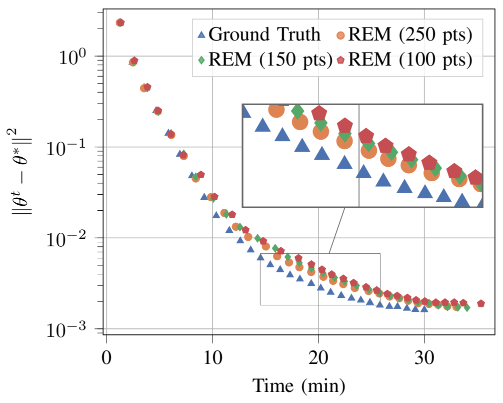


Fig. 5: Model convergence with VREM-FL and varying REM quality for linear regression experiment. The curves correspond to different estimates of the REM obtained by varying granularity of available measures.

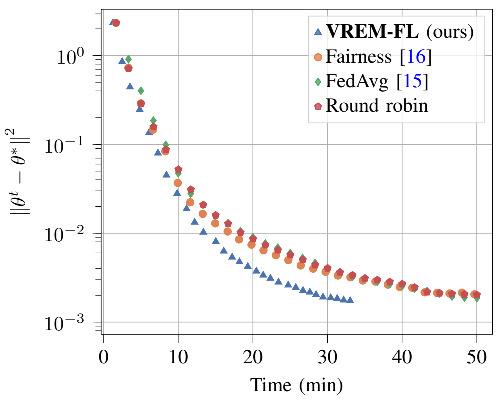


Fig. 6: Model convergence with VREM-FL *vs.* scheduling benchmarks for linear regression experiment. All algorithms achieve comparable accuracy, but VREM-FL significantly reduces training latency (33 min *vs.* 50 min).

algorithm. In Section VII-C1, we describe the synthetic dataset used for this experiment. In Section VII-C2, we study how performance of VREM-FL varies with the precision of the REM, considering both perfect channel knowledge and estimated REMs that may differ from actual channel conditions. In Section VII-C3, we compare VREM-FL against literature benchmarks and demonstrate its superiority in reducing training time while more sparingly allocating channel resources. In Section VII-C4, we perform several ablations to inspect how the metrics included in (2) are affected by different design choices of VREM-FL parameters.

1) Synthetic dataset generation: We generate S synthetic data samples $x_1, ..., x_S$, each obtained as a random linear combination of the form $x = \sum_{j=1}^{n} z_j \tilde{u}_j$, where $z_1, ..., z_n$ are i.i.d. Gaussian random variables, and $\{\tilde{u}_1, ..., \tilde{u}_n\} = \{\sigma_1 u_1, ..., \sigma_n u_n\}$, with $\sigma_1, ..., \sigma_n$ constants between 10^{-2} and 1, while $\{u_1, ..., u_n\}$ are n linearly independent basis vectors. Response values $y_1, ..., y_S$ are generated from the data samples as $y_i = x_i^T \theta^*$, $i = 1, ..., S$, where the generating parameter θ^* is also obtained as a random linear combination of $\{\tilde{u}_1, ..., \tilde{u}_n\}$.

The least squares loss in (FL) is $\ell(\theta) = \sum_{i=1}^{S} (\theta^\top x_i - y_i)^2$. We show here the results obtained with regularization parameter $\lambda = 10^{-4}$. We obtained similar results with $\lambda \in \{10^{-3}, 10^{-5}\}$ that are illustrated in Appendix B. For the purpose of the ablation study with synthetic data, we set a small parameter dimension $n = 25$. Accordingly, to get meaningful results with respect to the scheduling design, we rescale the bitrate values by a factor 2×10^{-5}.

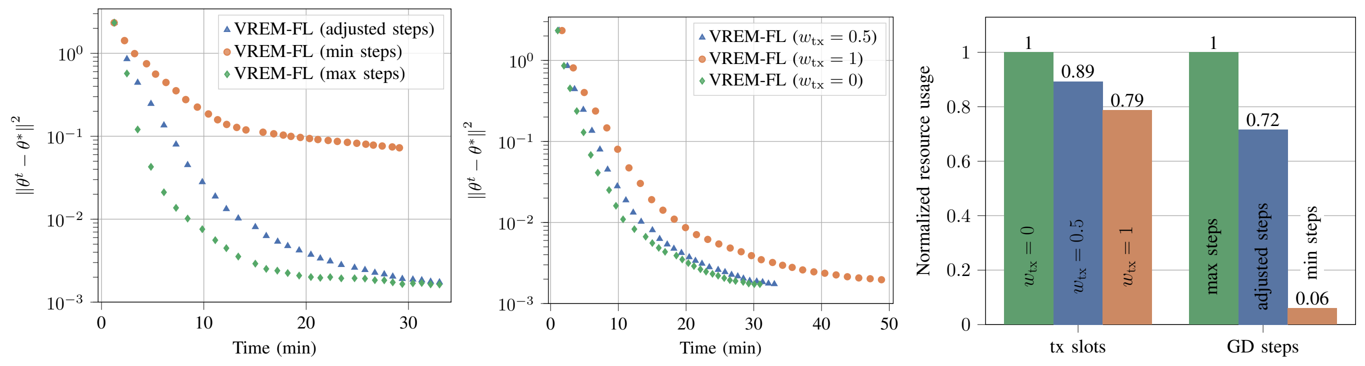
2) Performance using estimated REMs: We show the performance of VREM-FL on the synthetic dataset described in Section VII-C1 when the REM is estimated via Gaussian Process Regression (GPR) [9], [13] performed on a limited number of measurements. We consider the cases where 100, 150, or 250 measurements are available in each BS cellular sector. For each sector, we select the measuring locations uniformly at random. To perform GPR, we assume that the standard deviation and the de-correlation distance of the shadowing process are known *a priori*.[2] The parameters for FL and VREM-FL are reported in the first column of Table II.

In Fig. 5, we show the distance from the optimal parameter θ^* of the least squares problem as a function of the simulation time, where different lines denote a different number of measures available to generate the REM maps through GPR. Markers denote the simulation slot where the server has received all the updates from the clients and computes the average, *i.e.,* the end of a learning round. Triangular blue markers correspond to the ground truth, namely, the real REM is available at the scheduler. We can see that the more points are available to generate the REM estimation, the more likely it is to avoid stragglers. Stragglers are those nodes that slow down the learning process because they do not have sufficient computational or communication resources. The presence of scheduled nodes that had an estimated channel quality superior to the real one makes the learning round last longer, hence the total latency increases. The estimated map obtained using 100 points produces a total learning duration approximately 5 minutes longer than the ground truth and 2 minutes longer than the estimated map with 250 points. For the rest of the experiments, we used the REM estimated with 250 points.

3) Comparison with other scheduling policies: We show the comparison of VREM-FL against scheduling benchmarks in Fig. 6. "FedAvg" and "Round robin" perform similarly. The learning accuracy converges, but it takes longer than the other two scheduling algorithms. The target of 30 rounds is not even reached after the simulation horizon of 50 minutes. Also, "Fairness" uses the whole simulation horizon. Nonetheless, it yields a slight advantage in learning performance at early stages because it integrates information in a smarter way, ensuring that all vehicles contribute evenly. Under the given settings, VREM-FL reduces the total latency by at least 28% while providing the same model accuracy. This is achieved by wisely using the network resources through estimated channel conditions.

4) Ablation study: We propose an ablation study to isolate the effects of the proposed local steps adaptation strategy (Fig. 7a) and transmission policy (Fig. 7b). Figure 7a shows that performing the minimum number of GD steps (label "min steps", *i.e.,* only one step) locally at each round makes convergence steady but slow. The policy labeled "max steps" is obtained by filling all idle slots between computation and transmission (see Fig. 3) with additional GD steps. In this way, the total latency is the same of our optimized solution and local models are transmitted during the same time slots, but the total number of local iterations is higher than the proposed

[2]If this is not the case, they can be estimated via standard techniques [9].



(a) Model convergence varying computation policy. (b) Model convergence varying transmission policy. (c) Resource usage under different policies.

Fig. 7: Ablation study on VREM-FL with linear regression experiment. For computation policies, we vary the number of local GD steps with fixed transmission weight $w_{tx} = 0.5$. For transmission policies, we vary the number of slots when vehicles occupy the channel while using the adjusted steps computation policy.

VREM-FL (label "adjusted steps"). Indeed, the latter version may limit the number of local steps based on convergence proxies (LP) and (GP). Although "max steps" is initially faster than "adjusted steps", both solutions converge after 40 minutes (*i.e.,* approximately 25 rounds). Figure 7c shows the GD steps normalized with respect to the strategy "max steps" on the right. Strategy "min steps" uses only 6% of the total steps and its convergence is too slow. Noteworthy, "adjusted steps" reduces the GD steps by 28%, which directly translates into higher energy efficiency and better usage of computation resources while reaching the same accuracy in a comparable time.

In Fig. 7b, the results relative to varying the weight w_{tx} in the optimization problem (7) are shown. Specifically, tuning w_{tx} between 0 and 1 makes the solution move along the Pareto front between the two extremes $w_{tx} = 1$ (orange circles), which corresponds to minimizing the number of slots where the channel is filled with communication, and $w_{tx} = 0$ (green diamonds), which corresponds to minimizing the total latency. Setting $w_{tx} = 0$ corresponds to transmitting the local models as soon as the GD steps are completed, while the policy $w_{tx} = 1$ uses the REM and waits for the available transmission window with the highest bitrate. Any coefficients in between correspond to a weighted solution between these two criteria. For example, we show the results for $w_{tx} = 0.5$ (blue triangles). As $w_{tx} \to 0$, convergence is faster because this solution minimizes the total latency. The scenario with $w_{tx} = 0.5$ is reasonably close and it takes 2 minute longer to complete the given 30 rounds. Conversely, setting $w_{tx} = 1$ penalizes only the usage of transmission resources and uses all the available 50 minutes. By looking at the resource usage (Fig. 7c, leftmost bars), we see that, in this context, it is possible to reduce the number of slots used for communications by at most 21% (*i.e.,* from $w_{tx} = 0$ to $w_{tx} = 1$). Interestingly, by setting $w_{tx} = 0.5$, we reduce the resource usage by 11%, hence significantly improving the efficiency while performing very close to $w_{tx} = 0$ in terms of overall training latency.

D. Second Experiment: Deep Learning for Real-World Semantic Segmentation

In this section, we address a vision-based semantic segmentation task by training a deep neural network model. This experiment allows us to validate the effectiveness of VREM-FL for real-world applications. In Section VII-D1, we describe the real-world semantic segmentation dataset Apollo, while in Section VII-D2 we report details on the learning model and VREM-FL parameters. Then, we perform two sets of experiments with different mobility patterns. For both experiments, we use the bitrate obtained through the estimation method of Section VII-C2 with 250 samples. In Section VII-D3, we use the vehicular mobility simulated with SUMO for the first experiment. In Section VII-D4, we use a real-world dataset with taxi trips in Rome, Italy.

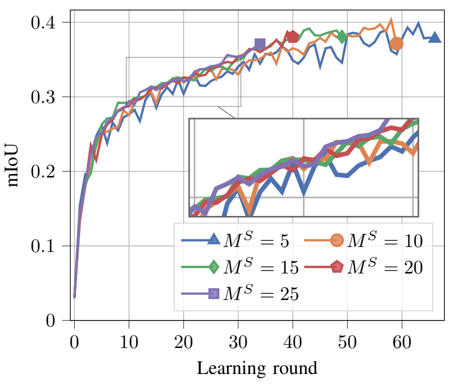
1) Real-world dataset ApolloScape and learning performance: We use the real-world `ApolloScape` lane segmentation dataset [14] to show the performance of VREM-FL on a realistic task. It consists of more than 110 thousand annotated frames from 73 street scene videos recorded in China with various weather conditions. We realistically split the dataset by assigning one record to each available vehicle, so that data are correlated in time and space within the same client and are non-iid across clients. In words, we simulate video streams independently acquired by vehicles with onboard cameras while traveling. We downsampled the frames to make the training compatible with limited computing resources of vehicles.

To measure the learning performance, we use the mean intersection over union (mIoU) score, a popular evaluation metric for segmentation tasks. The mIoU is obtained by first computing the ratio of the area of the intersection between predicted and ground truth regions to the area of their union, and then taking their average. Formally, let $\mathcal{R}$ denote the set of semantic regions and $|\mathcal{R}|$ denote its cardinality, then it holds

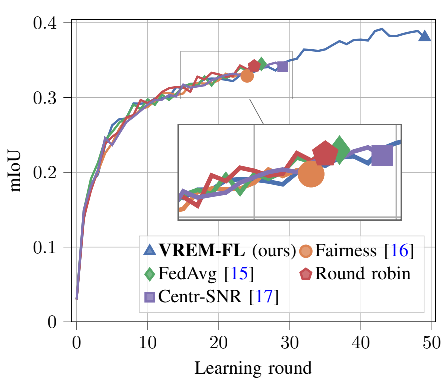
$$\text{mIoU} \doteq \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{r^{\text{pred}} \cap r^{\text{true}}}{r^{\text{pred}} \cup r^{\text{true}}}. \tag{11}$$

2) Training and VREM-FL settings: We train the deep network `deeplabv3` [71], a state-of-the-art model for semantic image segmentation, choosing `mobilenetv3-large` [72] as a backbone. The initial local learning rate is set to $\text{lr} = 2.5 \times 10^{-4}$ with a cosine annealing scheduler and the local optimizer is SGD. We train the model with minibatches of size 32, and the vehicles run $s_v = 3$ local steps of SGD per time slot (measured on an NVIDIA GeForce RTX 2080 GPU).

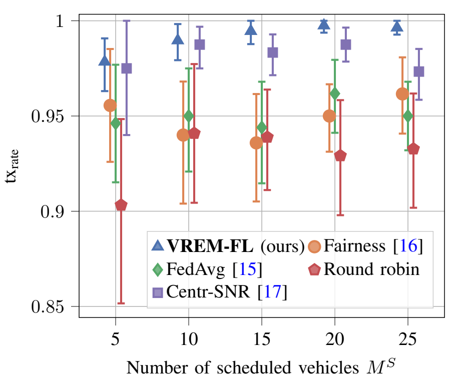
We modify the VREM-FL parameters to reflect the different



(a) Learning performance (mIoU score) with VREM-FL as the number of scheduled vehicles varies. Few scheduled vehicles make learning unstable; many vehicles increase the total latency.



(b) Learning performance with VREM-FL *vs.* scheduling benchmarks ($M_t^S = 15$). VREM-FL doubles the learning rounds as compared to the benchmarks given the same time horizon.



(c) Successful transmissions of scheduled clients. By smartly leveraging mobility patterns and REM, VREM-FL achieves the fewest missed updates and the most efficient usage of channel resources.

Fig. 8: Performance of VREM-FL on the real-world semantic segmentation experiment with simulated vehicular mobility.

TABLE III: Performance of VREM-FL *vs.* the benchmarks in the real-world semantic segmentation experiment with simulated mobility w.r.t. the performance metrics addressed through the cost function (2). For the transmission slots we report the mean value with a 95% confidence interval.

	mIoU	Learning rounds	Tx slots
VREM-FL	**0.392**	**49**	$\mathbf{6.74 \pm 0.05}$
Fairness [16]	0.332	24	15.0 ± 1.5
FedAvg [15]	0.346	26	15.2 ± 1.8
Round robin	0.343	25	17.5 ± 2.0
Centr-SNR [17]	0.347	29	9.04 ± 0.65

nature of the problem; see the second column of Table II. These choices increase the local SGD steps and induce vehicles to wait for the best transmission window in the given time.

3) Simulated mobility: In Fig. 8a, the mIoU score of VREM-FL is plotted against the learning round as the number of scheduled clients M^S varies. Selecting a few clients, *e.g.*, $M^S = 5$, the system performs many rounds within the simulation horizon (one hour) because each round is completed in short time. However, the learning quality is poor as witnessed by frequent spikes because a low number of clients is not representative of the full dataset, which may vary a lot across rounds. On the other hand, increasing the number of clients results in smoother learning curves. However, for *e.g.*, $M^S = 25$, we trade enhanced learning stability for a longer round duration because scheduling more clients means increasing the chance that at least one of them has a poor channel and takes a long time to transmit the parameters. The zoomed area in Fig. 8a reveals that the learning curves stabilize for $M^S \geq 15$. For the following results, we set $M^S = 15$ as the best tradeoff between learning quality and latency.

Figure 8b compares VREM-FL with the benchmark scheduling algorithms. There is no significant difference w.r.t. the learning quality and even "Fairness" does not provide advantages concerning "FedAvg". However, using VREM-FL, the server aggregates the parameters at a double rate than the benchmarks. Leveraging the REM, the scheduler chooses the fastest clients to transmit their local model weights, and the system performs 49 learning rounds instead of the 24-26 performed by the benchmarks within the same time horizon. This directly translates to a higher mIoU for the same training time – about 9.3% higher than the best benchmark strategy. The detailed comparison with respect to the three performance metrics addressed in Problem 1 is provided in Table III.

Clients that overestimate the channel quality may fail to send their model parameters update within the learning round deadline, wasting computation and communication resources. To assess how VREM-FL prevents resources from being wasted, we evaluate the fraction of scheduled clients uploading the model to the server within the deadline. Formally, we compute

$$\text{tx}_{\text{rate}} \doteq \frac{1}{T} \sum_{t \in \mathcal{T}} \frac{\left|\{v \in \mathcal{V}_t^S : v \text{ uploads update in round } t\}\right|}{M_t^S}. \quad (12)$$

Figure 8c shows tx_{rate} as the number of scheduled clients M^S varies. Compared to other algorithms, the channel quality maps allow VREM-FL and "Centr-SNR" to effectively select only those clients who will send their weights in time. This result is consistent since the variance is relatively small. VREM-FL performs slightly better than "Centr-SNR" as the channel quality is evaluated locally at the vehicles estimating the location at transmission time. On the other hand, a drop in the share of clients transmitting their weights is observed without REM availability, with a worst-case scenario of 85% in the simulated settings. The average value of the benchmark algorithms is around 94%, while for VREM-FL is close to 100%. This demonstrates that not only VREM-FL outperforms the benchmark strategies in terms of learning performance, but it also does so with a more economical and efficient usage of computing resources at vehicles and communication resources at the network edge.

4) Real-world mobility: We now apply VREM-FL to the dataset roma/taxi [11] that gathers traces of taxi trips recorded in Rome, Italy, between February and March 2014. For training, we selected the recordings from 7:30 pm to 8:30 pm on February 15, comprising 92 different taxis in total. In the original recordings, each taxi independently and asynchronously transmitted its real-time location every 15 s. We linearly interpolate the recorded traces and obtain synchronous trajectories at 1 Hz to allocate resources in a granular fashion.

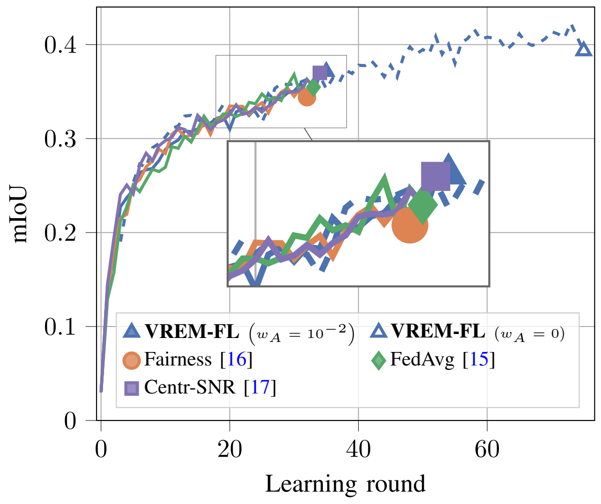


Fig. 9: Learning performance (mIoU score) with VREM-FL *vs.* scheduling benchmarks on the real-world semantic segmentation experiment with real-world vehicular mobility. VREM-FL ($w_A = 0$) outperforms the benchmarks as it uses the REM information to effectively schedule fastest transmitting vehicles. In this way, it performs more learning rounds in the same time horizon.

The REM varies a little at a few meter distance, hence we expect the bitrate at interpolated locations to be very similar to the one the taxis would experience along the true routes.

In Fig. 9, the mIoU of VREM-FL with two values of the fairness weight w_A is compared with the benchmarks. Setting $w_A = 10^{-2}$ allows us to combine the REM with fairness information; cf. (9). However, since at least 11% of the vehicles are scheduled in each round, fairness has a marginal impact because each vehicle is frequently selected in any case. The effect on learning is thus negligible, and VREM-FL with $w_A = 10^{-2}$ performs only 1 to 3 learning rounds more than the benchmarks within the given time horizon. On the other hand, setting $w_A = 0$ means scheduling the vehicles based on the sole REM and mobility information, and VREM-FL performs about 40 learning rounds more than the benchmarks, reaching a mIoU score of 0.43 as opposed to 0.37 achieved by "Centr-SNR" and VREM-FL with $w_A = 10^{-2}$. Figure 10 shows the number of communications slots used by the vehicles to upload the local models (left) and the fraction of successful uploads tx_{rate} (right). VREM-FL with $w_A = 0$ significantly outperforms the benchmarks, reducing the channel usage by 45% concerning the best benchmark "Centr-SNR". This highlights the need for online, distributed, and predictive scheduling in vehicular contexts as the channel quality experienced both across learning rounds and during each round can vary a lot depending on the vehicle speed and the environment geometry. VREM-FL with $w_A = 10^{-2}$ is still better than "FedAvg" and "Fairness" because it incorporates channel quality information to schedule vehicles. The successful update rate tx_{rate} confirms the trends just discussed. VREM-FL with $w_A = 0$ ensures almost total participation of the scheduled clients (99.9%), although the centralized approximation "Centr-SNR" performs close (99%), suggesting that in this case the bitrate at the beginning of a round is a good proxy for that experienced during the round.

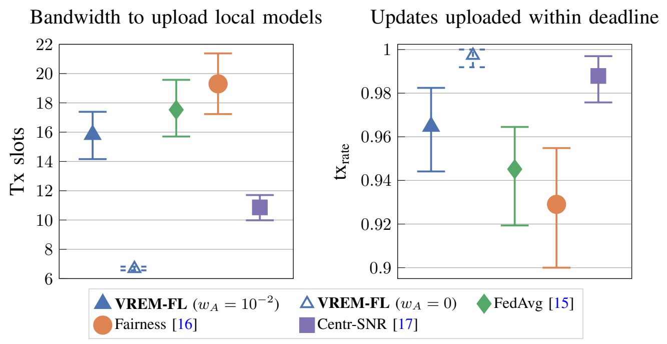


Fig. 10: Resource usage comparison between VREM-FL and scheduling benchmarks on the semantic segmentation experiment with real-world mobility. VREM-FL ($w_A = 0$) reduces the bandwidth for transmissions of updates (left) and increases the fraction of scheduled vehicles that upload the updates within the deadline (right), significantly improving resource efficiency.

VIII. Conclusion and Future Work

Motivated by the need of efficient solutions for FL tasks in vehicular networks, we have proposed VREM-FL, a computation-scheduling co-design algorithm that jointly optimizes a learning-related performance metric and network-related communication resources. Specifically, VREM-FL orchestrates local computations at the vehicles, transmission of their local models to the edge server, and schedules clients at each learning iteration to strike a good balance between learning accuracy, training time, and wireless channel usage. Experimental results on a synthetic LS problem and on a real-world semantic segmentation task demonstrate that VREM-FL provides superior learning performance as compared to common scheduling strategies by promoting a frugal use of computation and communication resources.

The present study focuses on the FL algorithm, client mobility is assumed to be given and non-controllable. However, future smart and autonomous vehicles may be in the position of changing their planned route to favor ancillary tasks, such as the execution of an FL algorithm or the transmission of data to roadside servers. Hence, we foresee scenarios where decision-making can be augmented via trajectory steering of (some of) the participating vehicles, to optimize even further the vehicle learning performance and their resource utilization.

Appendix A
Analytical Derivation of Local Proxy (LP)

We choose the proxy $\Theta_t^v(H_t^v)$ based on the convergence behavior of deterministic gradient descent when applied to least squares problems. For completeness, we now illustrate the explicit derivation of this proxy. Let $\ell(\theta) = \sum_{i=1}^{S}(\theta^\top x_i - y_i)^2 + \lambda\|\theta\|^2$ be a regularized quadratic cost function with $\lambda > 0$ (see Section VII-C1). We denote by $g(\theta) = \nabla\ell(\theta) \in \mathbb{R}^n$ and $\Lambda = \nabla^2\ell(\theta) \in \mathbb{R}^{n\times n}$ the gradient and the (constant) Hessian matrix of the cost function, respectively. Let θ^* be the unique minimizer of $\ell(\theta)$. In least squares, the gradient can be written as $g(\theta) = \Lambda(\theta - \theta^*)$ [63]. Recall that, for a constant step size $\alpha > 0$, the gradient descent update is $\theta^{t+1} = \theta^t - \alpha g(\theta^t)$ starting from the initial parameter θ^0.

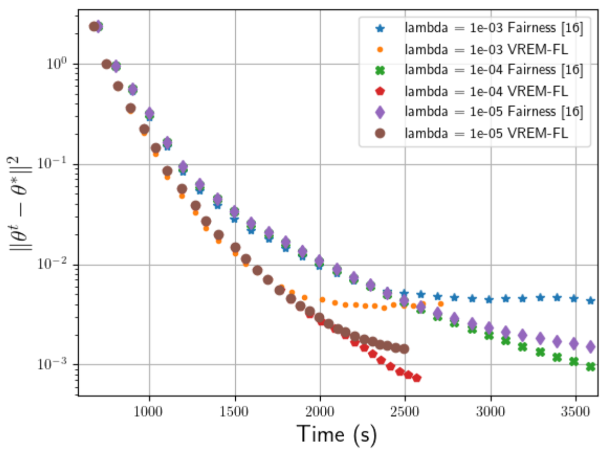


Fig. 11: Comparison of VREM-FL against Fairness [16], for different values of the regularization parameter λ.

Therefore, we can write

$$\begin{aligned} g(\theta^{t+1}) &= \Lambda\left(\theta^{t+1}-\theta^*\right) = \Lambda\left(\theta^t-\theta^*-\alpha g\left(\theta^t\right)\right) \\ &= g\left(\theta^t\right)-\alpha\Lambda g\left(\theta^t\right) = \left(I-\alpha\Lambda\right)g\left(\theta^t\right). \end{aligned} \tag{13}$$

Hence, the rate of convergence of $g(\theta^t)$ to zero is dictated by the largest eigenvalue (in magnitude) of $(I-\alpha\Lambda)$ that, denoting the eigenvalues of Λ by $\lambda_1 \geq \cdots \geq \lambda_n > 0$, is minimized by choosing $\alpha = \frac{2}{\lambda_1+\lambda_n}$ (see, *e.g.,* the proof in [63, Theorem 1]). This yields a convergence factor of $\frac{\lambda_1-\lambda_n}{\lambda_1+\lambda_n} \leq 1-\frac{\lambda_n}{\lambda_1} = 1-\frac{1}{\kappa}$, where $\kappa = \frac{\lambda_1}{\lambda_n}$ is the condition number of the problem. Consequently, we get

$$\left\|g\left(\theta^{t+1}\right)\right\| \leq \left(1-\frac{1}{\kappa}\right)^{t+1}\left\|g\left(\theta^0\right)\right\|, \tag{14}$$

from which we derive the proxy (LP).

Appendix B
Additional Experiments on Linear Regression

Here, for completeness, we include some results illustrating the performance of VREM-FL under different values of the regularization parameter λ in (FL). In particular, we show results for $\lambda = 10^{-3}, 10^{-4}, 10^{-5}$. The results consistently show the superior performance of VREM-FL under all the considered configurations of the regularization parameter.

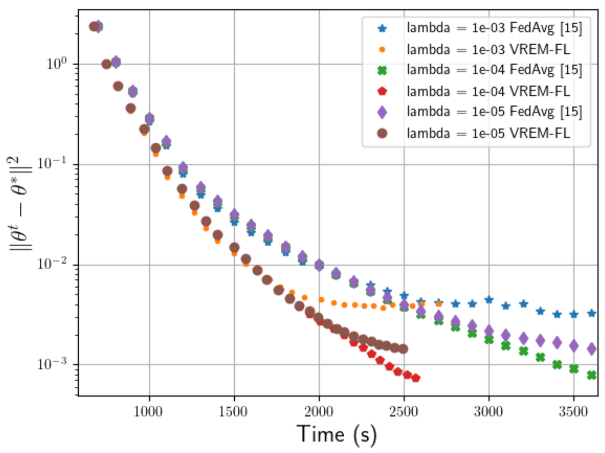


Fig. 12: Comparison of VREM-FL against FedAvg [15], for different values of the regularization parameter λ.

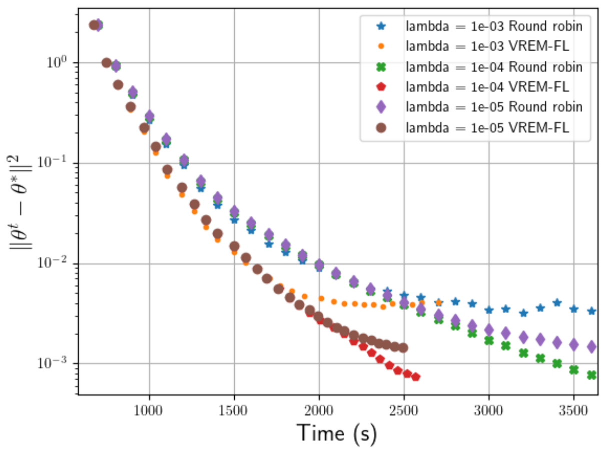


Fig. 13: Comparison of VREM-FL against the round robin benchmark, for different values of the regularization parameter λ.

References

[1] 5GAA. (2023) Evolution of vehicular communication systems beyond 5g. Accessed on: May 30, 2024. [Online]. Available: https://5gaa.org/evolution-of-vehicular-communication-systems-beyond-5g/
[2] Siemens. (2021, Jan. 22) *The Data Deluge: What do we do with the data generated by AVs?* Accessed on: Nov. 15, 2023. [Online]. Available: https://blogs.sw.siemens.com/polarion/the-data-deluge-what-do-we-do-with-the-data-generated-by-avs/
[3] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6g wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 122–134, 2020.
[4] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv e-prints*, p. arXiv:1610.05492, 2016.
[5] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani, "Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients," *Proc. NeurIPS*, vol. 34, pp. 14 606–14 619, 2021.
[6] 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, 2020.
[7] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457–2471, 2020.
[8] S. Bi, J. Lyu, Z. Ding, and R. Zhang, "Engineering radio maps for wireless resource management," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 133–141, 2019.
[9] N. Dal Fabbro, M. Rossi, G. Pillonetto, L. Schenato, and G. Piro, "Model-free radio map estimation in massive MIMO systems via semi-parametric Gaussian regression," *IEEE Wireless Commun. Lett.*, vol. 11, no. 3, pp. 473–477, 2022.
[10] Eclipse. (2023) *Simulator of Urban MObility*. Accessed on: Nov. 15, 2023. [Online]. Available: https://eclipse.dev/sumo/
[11] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "Crawdad roma/taxi," 2022, accessed on: Mar. 6, 2024. [Online]. Available: https://dx.doi.org/10.15783/C7QC7M
[12] A. Grassi, G. Piro, G. Boggia, M. Kurras, W. Zirwas, R. SivaSiva Ganesan, K. Pedersen, and L. Thiele, "Massive MIMO interference coordination for 5G broadband access: Integration and system level study," *Computer Networks*, vol. 147, pp. 191–203, 2018.
[13] L. S. Muppirisetty, T. Svensson, and H. Wymeersch, "Spatial wireless channel prediction under location uncertainty," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1031–1044, 2015.
[14] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and

R. Yang, "The apolloscape dataset for autonomous driving," in *Proc. IEEE CVPR Workshops*, 2018, pp. 954–960.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[16] M. E. Ozfatura, J. Zhao, and D. Gündüz, "Fast federated edge learning with overlapped communication and computation and channel-aware fair client scheduling," in *Proc. IEEE SPAWC*, 2021, pp. 311–315.

[17] C. Chen, B. Jiang, S. Liu, C. Li, C. Wu, and R. Yin, "Efficient federated learning in resource-constrained edge intelligence networks using model compression," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 2643–2655, 2024.

[18] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3643–3658, 2021.

[19] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3606–3621, 2021.

[20] L. Ye and V. Gupta, "Client scheduling for federated learning over wireless networks: A submodular optimization approach," in *Proc. IEEE CDC*, 2021, pp. 63–68.

[21] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *Proc. IEEE ICC*, 2020, pp. 1–6.

[22] I. Mohammed, S. Tabatabai, A. Al-Fuqaha, F. E. Bouanani, J. Qadir, B. Qolomany, and M. Guizani, "Budgeted online selection of candidate IoT clients to participate in federated learning," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5938–5952, 2021.

[23] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, 2020.

[24] J. Zhang, S. Chen, X. Zhou, X. Wang, and Y.-B. Lin, "Joint scheduling of participants, local iterations, and radio resources for fair federated learning over mobile edge networks," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2022.

[25] A. Taïk, H. Moudoud, and S. Cherkaoui, "Data-quality based scheduling for federated edge learning," in *Proc. IEEE LCN*, 2021, pp. 17–23.

[26] Y. He, J. Ren, G. Yu, and J. Yuan, "Importance-aware data selection and resource allocation in federated edge learning system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13 593–13 605, 2020.

[27] J. Leng, Z. Lin, M. Ding, P. Wang, D. Smith, and B. Vucetic, "Client scheduling in wireless federated learning based on channel and learning qualities," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 732–735, 2022.

[28] J. Du, B. Jiang, C. Jiang, Y. Shi, and Z. Han, "Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1035–1050, 2023.

[29] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 124–135, 2019.

[30] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7857–7867, 2016.

[31] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2015.

[32] 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. Computer Society*, vol. 1, pp. 45–61, 2020.

[33] D. Jallepalli, N. C. Ravikumar, P. V. Badarinath, S. Uchil, and M. A. Suresh, "Federated learning for object detection in autonomous vehicles," in *Proc. IEEE BigDataService*, 2021, pp. 107–114.

[34] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. 35, no. 2, pp. 152–159, 2021.

[35] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1616–1629, 2022.

[36] X. Huang, P. Li, R. Yu, Y. Wu, K. Xie, and S. Xie, "Fedparking: A federated learning based parking space estimation with parked vehicle assisted edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9355–9368, 2021.

[37] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 2027–2057, 2021.

[38] A. M. Elbir, B. Soner, S. Çöleri, D. Gündüz, and M. Bennis, "Federated learning in vehicular networks," in *Proc. IEEE MeditCom*, 2022, pp. 72–77.

[39] 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. 23 920–23 935, 2020.

[40] Z. Yu, J. Hu, G. Min, H. Xu, and J. Mills, "Proactive content caching for Internet-of-Vehicles based on peer-to-peer federated learning," in *Proc. IEEE ICPADS*, 2020, pp. 601–608.

[41] X. Zhou, W. Liang, J. She, Z. Yan, and K. I.-K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6G supported internet of vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5308–5317, 2021.

[42] P. K. Sangdeh, C. Li, H. Pirayesh, S. Zhang, H. Zeng, and Y. T. Hou, "CF4FL: A communication framework for federated learning in transportation systems," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, pp. 3821–3836, 2023.

[43] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, and W. Shi, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11 073–11 087, 2022.

[44] M. F. Pervej, R. Jin, and H. Dai, "Resource constrained vehicular edge federated learning with highly mobile connected vehicles," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1825–1844, 2023.

[45] B. Xie, Y. Sun, S. Zhou, Z. Niu, Y. Xu, J. Chen, and D. Gunduz, "MOB-FL: Mobility-aware federated learning for intelligent connected vehicles," in *Proc. IEEE ICC*, 2023, pp. 3951–3957.

[46] Q. Wu, X. Wang, Q. Fan, P. Fan, C. Zhang, and Z. Li, "High stable and accurate vehicle selection scheme based on federated edge learning in vehicular networks," *China Commun.*, vol. 20, no. 3, pp. 1–17, 2023.

[47] G. Wang, F. Xu, H. Zhang, and C. Zhao, "Joint resource management for mobility supported federated learning in internet of vehicles," *Future Gener. Comput. Syst.*, vol. 129, pp. 199–211, 2022.

[48] X. Zhang, Z. Chang, T. Hu, W. Chen, X. Zhang, and G. Min, "Vehicle selection and resource allocation for federated learning-assisted vehicular network," *IEEE Trans. Mob. Comput.*, pp. 1–12, 2023.

[49] V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, "Federated learning for connected and automated vehicles: A survey of existing approaches and challenges," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 119–137, 2024.

[50] H. Abou-zeid, H. S. Hassanein, and S. Valentin, "Optimal predictive resource allocation: Exploiting mobility patterns and radio maps," in *Proc. IEEE Global Commun. Conf.*, 2013, pp. 4877–4882.

[51] A. C. S. Rodriguez, N. Haider, Y. He, and E. Dutkiewicz, "Network optimisation in 5G networks: A radio environment map approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12 043–12 057, 2020.

[52] M. Hoffmann, P. Kryszkiewicz, and A. Kliks, "Increasing energy efficiency of massive-MIMO network via base stations switching using reinforcement learning and radio environment maps," *Computer Commun.*, vol. 169, pp. 232–242, 2021.

[53] W. B. Chikha, M. Masson, Z. Altman, and S. B. Jemaa, "Radio environment map based inter-cell interference coordination for massive-MIMO systems," *IEEE Trans. Mobile Comput.*, 2022.

[54] M. Hoffmann and P. Kryszkiewicz, "Beam management driven by radio environment maps in O-RAN architecture," *arXiv e-prints*, p. arXiv:2303.11742, 2023.

[55] ——, "Radio environment map and deep Q-learning for 5G dynamic point blanking," in *Proc. SoftCOM*, 2022, pp. 1–3.

[56] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, 2019.

[57] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous V2X communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, 2021.

[58] D. Wen, K.-J. Jeon, and K. Huang, "Federated dropout—a simple approach for enabling federated learning on resource constrained devices," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 923–927, 2022.

[59] K. Sato and T. Fujii, "Kriging-based interference power constraint: Integrated design of the radio environment map and transmission power," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 1, pp. 13–25, 2017.

[60] K. Sato, K. Suto, K. Inage, K. Adachi, and T. Fujii, "Space-frequency-interpolated radio map," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 714–725, 2021.

[61] V.-P. Chowdappa, C. Botella, J. J. Samper-Zapater, and R. J. Martinez, "Distributed radio map reconstruction for 5G automotive," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 2, pp. 36–49, 2018.

[62] L. Ballotta, N. Dal Fabbro, G. Perin, L. Schenato, M. Rossi, and G. Piro, "VREM-FL: Mobility-aware computation-scheduling co-design for vehicular federated learning," *arXiv e-prints*, p. arXiv:2311.18741, 2024, (pdf).

[63] N. Dal Fabbro, S. Dey, M. Rossi, and L. Schenato, "SHED: A Newton-type algorithm for federated learning based on incremental Hessian eigenvector sharing," *Automatica*, vol. 160, p. 111460, 2024.

[64] OSM. (2023) *OpenStreetMap*. Accessed on: Nov. 15, 2023. [Online]. Available: https://www.openstreetmap.org/#map=6/42.088/12.564

[65] Mathworks. (2023) *5G Toolbox*. Accessed on: Nov. 15, 2023. [Online]. Available: https://www.mathworks.com/help/5g/index.html?s_tid=CRUX_lftnav

[66] 3GPP, "Study 3D Channel Model for LTE," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.873, 01 2018, version 12.7.0.

[67] ITU-R, "Guidelines for Evaluation of Radio Interface Technologies for IMT-Advanced," Technical Report M.2135, 2008.

[68] 3GPP, "NR; Physical channels and modulation." 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.211, 2020.

[69] ——, "NR; Multiplexing and channel coding." 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.212, 2020.

[70] ——, "NR; Physical layer procedures for data." 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, 2020.

[71] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv e-prints*, p. arXiv:1706.05587, 2017.

[72] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proc. IEEE/CVF ICCV*, 2019, pp. 1314–1324.

Luca Ballotta received the Master's degree in Automation Engineering and the Ph.D. degree in Information Engineering from the University of Padova, Italy, in 2019 and 2023, respectively. He is currently a postdoctoral researcher at the Delft University of Technology, Delft Center for Systems and Control (DCSC). He was Visiting Student at the Massachusetts Institute of Technology in 2020 and 2022. He was awarded with the Young Author Prize at the 2020 IFAC World Congress and was finalist of the 2024 EECI PhD Award. His research interests include control over networks under resource constraints, resilient consensus and distributed control, and safe distributed control.

Nicolò Dal Fabbro obtained his Master's degree in Telecommunications Engineering and his PhD degree in Information Engineering from the University of Padova, Italy, in 2020 and 2023, respectively. He is currently a postdoctoral researcher at the University of Pennsylvania, USA, where he is with the department of Electrical and Systems Engineering. His research interests are in federated learning, reinforcement learning and wireless sensing.

Giovanni Perin (Member, IEEE) received the M.Sc. degree (summa cum laude) in ICT for Internet and Multimedia and the Ph.D. degree in Information Engineering from the University of Padova, Italy, in 2019 and 2023, respectively, where he is currently a postdoc research fellow. In 2019, he spent six months as a visiting student with the Deutsche Telekom Chair of Communication Networks, Technical University of Dresden, Germany, working on broadcast routing, while in 2022, he was a Visiting Scholar with the University of California at Irvine, Irvine, USA, conducting research on vehicular communications and edge computing. His research focuses on sustainable edge computing, distributed optimization and processing, and federated learning.

Luca Schenato (Fellow, IEEE) received the Dr. Eng. degree in electrical engineering from the University of Padova in 1999 and the Ph.D. degree in Electrical Engineering and Computer Sciences from the UC Berkeley, in 2003. He held a post-doctoral position in 2004 and a visiting professor position in 2013-2014 at U.C. Berkeley. Currently he is Full Professor at the Information Engineering Department at the University of Padova. His interests include networked control systems, multi-agent systems, wireless sensor networks, smart grids and cooperative robotics. Luca Schenato has been awarded the 2004 Researchers Mobility Fellowship by the Italian Ministry of Education, University and Research (MIUR), the 2006 Eli Jury Award in U.C. Berkeley and the EUCA European Control Award in 2014, and IEEE Fellow in 2017. He served as Associate Editor for IEEE Trans. on Automatic Control from 2010 to 2014 and he is he is currently Senior Editor for IEEE Trans. on Control of Network Systems and Associate Editor for Automatica.

Michele Rossi (Senior Member, IEEE) is a Professor of Wireless Networks at the Department of Information Engineering (DEI) and of Machine Learning for Human Data Analysis at the Department of Mathematics, both at the University of Padova, Italy. He is the coordinator of the Master's Degree in ICT for Internet and Multimedia at DEI. His research interests lie broadly in wireless sensing systems (joint communication and sensing for next generation wireless systems), green mobile networks (energy efficient operation and resource allocation), energy efficient machine and deep learning (lightweight architectures including spiking neural networks) for edge computing. Over the years, he has been involved in numerous European projects on wireless sensing and Internet of Things and has collaborated with major companies such as Ericsson, DOCOMO, Nokia, Samsung and INTEL. His research is currently supported by the European Commission through the projects GREENEDGE (GA no. 953775) on "green edge computing for mobile networks" (project coordinator) and ROBUST-6G on "smart, automated and reliable security service platforms for 6G systems" (GA no. 101139068). Prof. Rossi has been the recipient of six best paper awards and one best dataset award from the IEEE, and currently serves on the Editorial Boards of the IEEE Transactions on Mobile Computing.

Giuseppe Piro (Member, IEEE) is an Associate Professor at "Politecnico di Bari", Italy. He received the Ph.D. degree in Electronic Engineering, a first, and a second level degree (both with Hons.) in Telecommunications Engineering from "Politecnico di Bari", Italy, in 2006 and 2008, respectively. His main research interests include wireless networks, network simulation tools, 5G and beyond, network security, nano-scale communications, Internet of Things, and Software-Defined Networking. He serves as Associate Editor for Internet Technology Letter (Wiley), Wireless Communications and Mobile Computing (Hindawi), and Sensors (MDPI).