Mobility and Deadline-aware Task Scheduling Mechanism for Vehicular Edge Computing

Joahannes B. D. da Costa, Allan M. de Souza, Rodolfo I. Meneguette, Eduardo Cerqueira, Denis Rosario, Christoph Sommer, Leandro Villas

***Abstract*—Vehicular Edge Computing (VEC) is a promising paradigm that provides cloud computing services closer to vehicular users. In VEC, vehicles and communication infrastructures can form pools with computational resources to meet vehicular services with low-latency constraints. These resource pools are known as Vehicular Cloud (VC). The usage of VC resources requires a task scheduling process. In this case, depending on its complexity, a vehicular service can be divided into different tasks. An efficient task scheduling needs to orchestrate where and for how long such tasks will run, considering the available pools, the mobility of nodes, and the tasks' deadline constraints. Thus, this article proposes an efficient VC task scheduler based on an approximation heuristic and resources prediction to select the best VC for each task, called MARINA. MARINA aims to analyze the behavior of vehicles that share their computational resources with the VC and make scheduling decisions based on the mobility (VC availability) of these vehicles. Simulation results under a realistic scenario demonstrate the efficiency of MARINA compared to existing state-of-the-art mechanisms in terms of the number of tasks scheduled, monetary cost, system latency, and Central Processing Unit (CPU) utilization.**

***Index Terms*—Vehicular Edge Computing, Task Scheduling, Resource Prediction, Recurrent Neural Network**

I. Introduction

VEHICLES are becoming more intelligent, connected, and autonomous to allow safer, greener, and efficient transportation systems [1]–[3]. This new connected and autonomous vehicle era will support several applications that require high bandwidth and low latency [4]. For instance, self-driving vehicles, real-time learning, and artificial intelligence-oriented applications are expected to be largely deployed, producing lots of data with low latency requirements [5], [6].

Vehicular Edge Computing (VEC) emerged as a promising paradigm to move communication, computing, and storage resources closer to vehicular users [7], [8]. In this sense, VEC aims to reduce latency while decreasing the heavy traffic from the network core [9]. VEC considers a set of Vehicular Cloud (VC) to allow users to request resources to meet application/service demands that require computational power above the locally supported ones. A VC group shares computational resources, such as processing units and storage capacity, available in a set of vehicles (*i.e.*, either moving or parked vehicles) and infrastructure (*i.e.*, Roadside Unit (RSU), 5G Base Stations (BSs), network controllers, and Remote Server (RS) in the Internet cloud) nodes to provide cloud services, such as the Infrastructure-as-a-Service (IaaS) [10].

In this VEC scenario, each network controller, called VEC controller, is responsible for coordinating a predefined city region, such as a neighborhood, for forming the VCs based on the regional knowledge about the vehicle and infrastructure nodes through the Vehicle-to-Infrastructure (V2I) communication. Hence, the VEC controller can be responsible for two main processes, namely, VC formation and task scheduling [11]–[13]. The VC formation process concerns an efficient grouping and management of computational resources available by the network entities to form a VC [4], [9]. On the other hand, the task scheduling process concerns the efficient use VC resources, *i.e.*, the tasks will be scheduled to be processed in a given VC. Therefore, these two processes are crucial to group computational resources and provide cloud services closer to vehicular users. Advance the state-of-the-art in the task scheduling process is essential to the success of connected autonomous vehicle ecosystems. New approaches to efficiently select and decide where and when tasks will be scheduled in dynamic and mobile vehicular environments are required. At least three key issues need to be addressed for developing an efficient task scheduling mechanism to meet users' demands in VEC scenarios, namely *(1) Vehicular Mobility*; *(2) Deadline Constraints*; and *(3) Monetary Costs*.

The *Vehicular Mobility* is one of the main challenging factors for the efficient performance of vehicular scenarios with support to VEC applications since vehicle mobility causes several changes in network topology and (including intermittent connections) [14]. For instance, vehicular mobility causes an unexpected disconnection between VC members and the VEC controller, leading to a fluctuation in terms of the number of available resources in the VC and/or causing the task processing results loss, impacting the system efficiency. Hence, the mobility of vehicles has a high impact in the

Manuscript received November 22, 2022; revised January 23, 2023; accepted May 3, 2023. This work is supported by the grants #2015/24494-8, #2018/16703-4, and #2021/13780-0 of the São Paulo Research Foundation (FAPESP). Rodolfo I. Meneguette would like to acknowledge the financial support granted by FAPESP, process #2020/07162-0, in his research.

J. B. D. da Costa is with the Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil, and also with the TU Dresden, Dresden, Germany. (web: ic.unicamp.br/~joahannes.costa)

A. M. de Souza and L. Villas are with the Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil. (email: allan.souza@ic.unicamp.br; leandro@ic.unicamp.br)

R. I. Meneguette is with the University of São Paulo (USP), São Carlos, Brazil. (email: meneguette@icmc.usp.br)

E. Cerqueira and D. Rosário are with the Federal University of Pará (UFPA), Belém, Brazil. (email: cerqueira@ufpa.br; denis@ufpa.br)

C. Sommer is with TU Dresden, Faculty of Computer Science, Dresden, Germany. (web: cms-labs.org/people/sommer)

IEEE Transactions on Intelligent Transportation Systems Digital library entry → https://doi.org/10.1109/TITS.2023.3276823

number of tasks attended/scheduled [15]. In this sense, it is essential to consider vehicular mobility information to estimate vehicles' location, which can be used to estimate the future resource availability in each VC. This aims in selecting stable VCs to run the tasks, where VC's stability refers to its low rate of resource variability over time. However, how to predict these vehicular dynamics accurately and explore their spatiotemporal correlation is still an open issue [16]. In this sense, Recurrent Neural Network (RNN) can use in vehicular mobility estimates due to their high accuracy rates in general form predictions [17]–[19].

The *Deadline Constraints* represent the time that tasks can wait to be served. In other words, it is important that the obtained result must be returned to the requesting entity before a hard deadline. Otherwise, this result becomes useless [20], [21]. Therefore, the tasks' deadline constraints can be better observed and considered from a resource availability estimation in the VCs, making it possible to infer the processing time and the service probability to increase the scheduled and completed tasks rate. Also, the system latency is minimized by prioritizing tasks with less processing time. The system latency represents the waiting time and the time that a task will remain running in a given processing configuration.

Finally, *Monetary Cost* refers to the price of using computational resources, such as the pay-as-you-go basis commonly used in cloud computing IaaS architectures [1]. For instance, the monetary cost can be minimized by knowing the task processing time, making using computational resources less costly for the end users. In addition to task deadline constraints, the tasks often have high processing requirements. Scheduling tasks with high computational power requirements, such as traffic image processing or real-time learning, is a challenging issue in a highly dynamic vehicular environment [22]. In other words, the task scheduling mechanisms must make accurate decisions considering all the tasks' computational requirements, regardless of the variability of these requirements in different classes of applications. Besides, scheduling mechanisms must always minimize costs for end-users.

To the best of our knowledge, a unique solution that considers mobility-awareness, deadline constraints, and monetary costs for decision-making in a task scheduling process is still an open issue and remains a challenge. Thus, this article proposes MARINA (Mobility and deAdline-awaRe task schedulIng mechaNism for vehiculAr edge computing) to maximize the number of tasks scheduled while minimizing the monetary costs of utilizing VC's resources. The MARINA runs on VEC controllers to coordinate the task scheduling in multiple VCs. In MARINA's operation, a vehicle without enough computational resources to run a specific task can forward this task to be processed somewhere in the vehicle ecosystem. In this sense, MARINA selects a set of tasks to be scheduled in real-time in each available VC based on the Pareto Optimality and Bin Covering Problem (BCP). Pareto optimality allows the joint minimization between the deadline and estimated processing time. Besides, the BCP makes it possible to find the best fit for the minimization provided by Pareto, always seeking to maximize the number of scheduled tasks. MARINA also considers a Long Short-Term Memory (LSTM) to predict resource availability in each VC based on vehicular mobility information. Hence, MARINA prioritizes scheduling tasks in VCs with more available resources to maximize the fulfillment of demands in as few rounds as possible. Simulation results show that, compared to state-of-the-art solutions, MARINA can schedule more tasks while minimizing monetary cost and system latency.

The main contributions of this article are the following:

- The use of vehicular mobility information and an LSTM architecture to predict the available resources in each VC with high accuracy.
- An efficient task scheduling mechanism that considers the task deadline, mobility-awareness, and monetary costs in its decision-making process. Thus, it schedules a high number of tasks without increasing the monetary cost of using VC's computational resources.
- A combined approach that leverages low-computational complexity techniques to minimize the monetary cost and maximize the number of scheduled tasks.
- A detailed performance evaluation with a realistic mobility trace, showing the benefits of vehicular mobility information for the task scheduling process compared to other state-of-the-art approaches.

The remainder of this article is structured as follows. Section II describes key related works. Section III introduces the system model, problem definition, and mobility prediction model. Section IV presents the MARINA's operation. Section V discusses the performance evaluation and results obtained. Finally, Section VI presents the conclusions and future works direction.

II. Related Work

This section presents state-of-the-art research regarding task scheduling in VCs. In general, scheduling approaches can be classified into three perspectives. *(1)* Centralized, where an entity has a scenario holistic view and makes more accurate decisions. This approach has the advantage of an environment global view, but maintaining this knowledge is a challenge for system scalability; *(2)* Decentralized, which has different agents that make decisions based on local knowledge. This approach does not need to maintain global knowledge, but the accuracy of decisions is compromised in some cases; and *(3)* Hybrid, which combines the advantages of two previous techniques to increase system efficiency [31]. Observing the advantages of each scheduling approach, we consider only the Centralized and Hybrid approaches due to the essential role the 5G network can play in these specific scenarios.

Pereira et al. [24] proposed FORESAM, a policy for scheduling tasks in VCs based on the fog computing paradigm for urban environments. Specifically, vehicles cooperate with the set of BSs to create a pool of resources for vehicular services. FORESAM decides whether resources are available based on Analytic Hierarchy Process (AHP) mathematical method. However, FORESAM employs greedy decision-making, where in case a task does not fit into the VC, it is discarded, impacting the overall system efficiency.

TABLE I
SUMMARY OF RELATED WORKS HIGHLIGHTING THEIR CHARACTERISTICS AND LIMITATIONS.

Related work	Characteristics			
	System design	VC's entities	Scheduling strategy	Mobility information
Hattab et al. [23]	Centralized	Vehicles	Queueing theory	
Da Costa et al. [13]	Centralized	Vehicles	Optimization	
Pereira et al. [24]	Hybrid	Vehicles/BS	Analytic Hierarchy Process (AHP)	
Dai et al. [25]	Hybrid	BS	Optimization	
Chen and Xu [21]	Hybrid	Vehicles	Reinforcement Learning (RL)	
Liu et al. [26]	Hybrid	Vehicles/BS	Deep Reinforcement Learning (DRL)	
Luo et al. [7]	Hybrid	Vehicles/BS	Particle Swarm Optimization (PSO)	
Gao et al. [27]	Hybrid	Vehicles/BS	Deep Q-Network (DQN)	
Misra and Bera [28]	Centralized	Vehicles	Integer Linear Program (ILP)	✓
Wu et al. [29]	Centralized	Vehicles/BS	Optimization	✓
Kazmi et al. [30]	Hybrid	Vehicles/BS	DRL	✓
MARINA	Hybrid	Vehicles/BS	Heuristic	✓

Some works consider optimization algorithms for decision-making. Da Costa et al. [13] introduced CRATOS, a combinatorial optimization-based mechanism for task scheduling in VCs. CRATOS considers the arrival of tasks in real-time and regardless of each other following a Poisson process. The VEC controller located at a higher level in the network receives the resource requests (which are the tasks) and schedules them in the available VCs. 0/1 Knapsack Problem is used to schedule tasks optimally given the contextual configuration (tasks' input size and VC's available computational resources). However, CRATOS does not consider tasks with deadlines higher than 1s in its decision process. Also, the user defines how much it will cost to process his task. Hence, CRATOS would not fit into an actual application, as the monetary cost model used by cloud providers is different.

Dai et al. [25] presented a probabilistic algorithm for cooperative task scheduling in VCs. The authors formulate Cooperative Computing Offloading (CCO) problem by modeling the procedure for uploading, migrating, and scheduling tasks based on queuing theory to minimize the delay in the system. The BS makes online scheduling based on the calculated probability provided by a probabilistic offloading algorithm. Three aspects are considered for computing the offloading probability: the time that vehicles will stay in BS's coverage range; the density of vehicles; and resources available in the BS. However, this computation is offline, which means it is executed on a high-performance server in the Internet cloud. The probability that the BS will meet demand is computed before the process of task arrival, which can degrade the system's performance in a highly dynamic environment. Finally, VCs considers only the resources available in the BSs.

Luo et al. [7] presented a detailed analysis of the delay and cost of task offloading for VCs. The authors first establish an offloading framework with communication and computation for VC, considering tasks with different requirements. In this sense, a multi-objective optimization problem is formulated to joint minimization the delay and the monetary cost. A Particle Swarm Optimization (PSO)-based computation offloading (PSOCO) algorithm is proposed to obtain the Pareto-optimal solutions. However, due to its bio-inspired approach, its convergence time can impact the algorithm's performance.

Some works use queuing theory to model the vehicular scenario. Hattab et al. [23] proposed a polynomial time algorithm for task scheduling in VCs with different resources. First, the algorithm classifies tasks according to the completion and wait times. Afterward, it selects a subset of tasks with the lowest proportion and then solves a sequence of Linear Programs. They formulate the bottleneck assignment problem, where the goal is to minimize the completion time of the scheduled tasks in the available VCs. However, this work does not consider the vehicle mobility for VC formation, *i.e.*, VCs are stationary, and the proposed algorithm considers only one VC.

Some approaches use Machine Learning (ML) techniques for the scheduling decision process. Kazmi et al. [30] proposed a task scheduling mechanism for mobile vehicular networks using a DRL-based approach. The approach considers electric vehicles in task processing. The focus is to make energy consumption more efficient. Mobility information considers the relative mobility and kinematic equations to estimate the communication phase between two vehicles. However, an evaluation comparing the solution with state-of-the-art is still necessary. Furthermore, the BS does not cooperate in task processing, decreasing system efficiency. Gao et al. [27] present a solution for task scheduling that aims to minimize service delay and energy consumption. The authors use an DQN approach for scheduling decision-making and a Gradient Descent (GD) method for Central Processing Unit (CPU) frequency allocation. However, the work needs to discuss the solution convergence time, which depending on the scenario, can make it unfeasible for more dynamic environments.

Chen and Xu [21] presented an RL-based algorithm for task scheduling in VC, called DATE-V, which provides resources for scheduling tasks with deadline constraints in the VCs. DATE-V is based on a contextual and combinatorial Multi-Armed Bandit (MAB) learning framework. The algorithm uses vehicle contextual data (*i.e.*, speed, location, and computational resources available) to infer the probability of completing a task replication under random vehicles. DATE-V also replicates the task received in the BS and sends such replications to different vehicles to ensure service attendance and increase its success rate. However, these replications potentially lead to a costly solution, given that defining the number of replications is challenging. Liu et al. [26] proposed a vehicle-assisted task scheduling approach for mobile

users, considering delay constraints. The authors formulate an optimization problem to maximize the long-term usefulness of VC's resources. Considering stochastic vehicle traffic, dynamic computation requests, and time-varying communication conditions, the problem is later formulated as a Semi-Markov Decision Process (SMDP). Two reinforcement learning methods are proposed to obtain the optimal computation offloading and VC's resource allocation, namely *(i)* Q-learning based and *(ii)* Deep Reinforcement Learning (DRL) methods. However, given the nature of DRL approaches, it is necessary to train the model in advance. Thus, the solution depends on the offline phase for its execution, which can negatively impact highly dynamic scenarios.

Some works consider mobility prediction to estimate vehicle positions and ensure reliability in the task scheduling process. For example, Misra and Bera [28] proposed a mobility-aware task scheduling scheme named Soft-VAN, which aims to minimize task computation delay in a software-defined vehicular network. Soft-VAN consists of two phases, fog node selection and task scheduling. An Integer Linear Program (ILP) is solved in the first phase to get the optimal number of fog nodes required for a given network. In the second phase, the authors formulate an optimization problem to minimize the delay in task computation and consider the vehicle's mobility and the parameters associated with the scheduling decisions. However, the algorithm depends on an offline phase to identify fog nodes to assist the delivery process of the tasks performed. Also, they do not consider the vehicles' dynamics to make the processing decisions for the task, only to send the final result.

Wu et al. [29] investigate the task scheduling and resource allocation optimization problem by considering the vehicular mobility effect in the VEC environment. Specifically, the authors formulate the joint optimization problem from a Min-Max perspective to reduce the overall task latency. Then they decompose the non-convex problem into two sub-problems, one-to-one matching and bandwidth resource allocation. Also, considering a vehicle's relatively stable moving patterns in a short period, the authors further introduce mobility prediction to design a mobility prediction-based scheme to obtain a better solution. However, the mobility model is unrealistic since vehicles need constant acceleration during the task scheduling.

Table I summarizes the main characteristics of reviewed studies considering system design, the entities forming the VCs, the approach used in the task scheduling process, and vehicular mobility information. Based on the state-of-the-art analysis, we conclude that centralized approaches allow the constructed knowledge to be global and more accurate. However, the network's overload levels can be high and overload the centralized entities, raising the monetary costs of using computational resources [13], [23], [28], [29]. In another direction, some works consider hybrid architectures, enabling to build on regional knowledge without burdening a central entity and the network's core with high message exchange [7], [21], [24]–[26]. A VEC architecture allows the cooperation of computational resources among vehicles and BSs for VCs creation, allowing resources to be always available. For instance, considering only vehicles, the approaches are limited to regions with high vehicle flows to maintain the high resource availability. On the other hand, considering BS as an entity that composes the VCs enables more effective management both in forming VCs and using resources, making the processes more stable and reliable [7], [24], [26], [29].

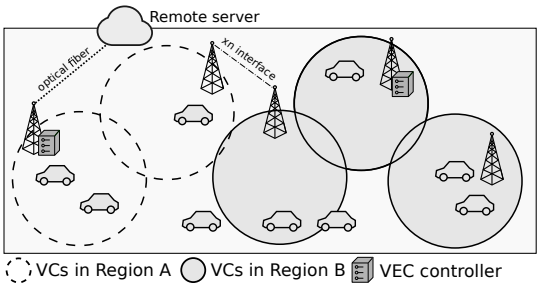


Fig. 1. The architecture employed by MARINA, presenting its main components; in particular, Vehicular Clouds (VCs) and Vehicular Edge Computing (VEC) controllers.

Regarding the scheduling approach, it is essential to consider decision-making techniques with low complexity to deal with the real-time requirements of different classes of applications explicitly defined by their deadlines. In this sense, in computational complexity order, some studies consider simple computing algorithms [23], [24], others consider optimization techniques [7], [13], [25], [28], [29], and some studies consider machine learning algorithms [21], [26], [27], [30].

In this sense, the techniques used should be less computationally complex, and thus their decision time does not negatively impact the system's overall efficiency. Furthermore, creating knowledge about the dynamics of computational resources through vehicular mobility is essential to estimate future resource availability. In this sense, approaches can make better decisions, relating task requirements, such as deadlines, and computational resources available in the future, reducing the task scheduling interruptions and the monetary cost employed by this process. However, only a few studies consider vehicular mobility information in their decision-making processes [28]–[30]. To the best of our knowledge, only MARINA considers every critical characteristic previously mentioned not provided by existing task scheduling mechanisms in a VEC environment. In other words, MARINA considers a hybrid architecture to avoid network overload, VCs formed by vehicles and BSs to increase resource availability, a polynomial-time heuristic for decision making, and vehicle mobility information to estimate future computational resources in each VC.

III. System Model

This section introduces the MARINA task scheduling in VEC scenarios, which considers available resources, tasks' deadline constraints, and vehicular mobility for decision-making. We first overview the designed system and then illustrate the VCs formation process, mobility prediction approach, and monetary cost model. Finally, we formulate the problem. Table II summarizes the key notations used in this work.

TABLE II
SUMMARY OF KEY NOTATIONS.

Symbol	Description	Symbol	Description
U	set of vehicles	u_i	a vehicle $\in U$
B	set of BSs	b_y	a BS $\in B$
V	set of VCs	v_j	a VC $\in V$
T	set of tasks	t_l	a task $\in T$
C	set of controllers	ω_{u_i}	vehicle's CPU-cycle
n	# of tasks	ϕ_{u_i}	vehicle's storage
x	# of vehicles	ω_{b_y}	BS's CPU-cycle
m	# of BSs/VCs	ϕ_{b_y}	BS's storage
e	# of controllers	Ω_j	VC's total CPU-cycle
l	index of task	Ψ_j	VC's shared CPU-cycle
i	index of vehicle	Φ_j	VC's total storage
j	index of VC	D_l^t	task deadline
o	index of controller	$d_l^t j$	task proc. time in VC j
y	index of BS	C_l	task monetary cost
$\mathcal{Q}$	waiting queue	w_l^t	task CPU-cycle required
$\mathcal{R}$	run queue	$Price(t_l)$	resource price used by t_l
R	number of regions	$\mathcal{P}$	pareto set

A. Overview

Figure 1 shows the system architecture composed of vehicles, BSs, VEC controllers, and an RS in the Internet cloud. We consider a scenario composed of x moving vehicles, denoted as $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Also, we consider a set of BSs deployed in the city, denoted as $b_y \in B = \{b_1, b_2, \ldots, b_m\}$, where m is the total number of BSs. Each BS has wired communication (*e.g.*, optical fiber) with the RS and could provide processing power and storage capacity on the network edge, decreasing response time for some applications [26]. Also, each BS on the 5G network has an *Xn* interface, which allows the information exchange between neighboring BSs and assists in the handover process [32]. Each vehicle has an On-Board Unit (OBU) that allows communication with the neighbor devices through Vehicle-to-Everything (V2X) communication. For instance, vehicles can associate with a BS and communicate with an RS to access the Internet, request resources for data processing, and share their resources.

We used a BS-UE (Base Station-User Equipment) method based on the maximum signal to interference plus noise ratio (SINR) for this association process. In this case, a vehicle will associate with a BS which provides the Max-SINR [33]. The Max-SINR ensures that the vehicle is associated with only one BS [34]. After the association between vehicle and BS, the BS sends the vehicle' information to RS.

In this scenario, we consider the city divided into R regions (neighborhoods), and each region has at least one BS. We also consider that each VEC controllers cover a given region R, and it is randomly deployed in a given BS of such region. It is important to mention that VEC controllers are randomly deployed in the scenario, since this is not our research focus. We denote the set of VEC controllers as $c_o \in C = \{c_1, c_2, \ldots, c_e\}$, where $e = |R|$ is the total number of controllers, and directly related to the number of regions R. Each controller is responsible for managing the BSs in its city region, where this management includes the VC formation and task scheduling processes.

In the VC formation process, the VEC controller requests

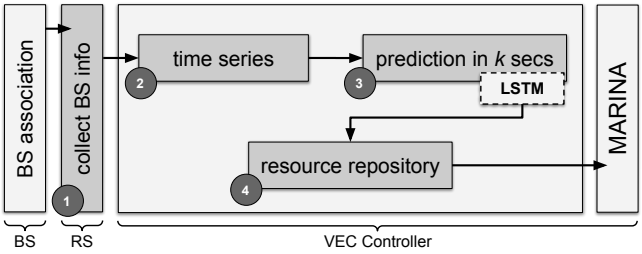


Fig. 2. Vehicular Cloud (VC) formation process – from Base Station (BS) association to scheduling – based on a prediction approach using Long Short-Term Memory (LSTM).

to the RS the information about BSs and vehicles to build their regional knowledge. In this case, the Publish/Subscribe paradigm is considered to obtain the relevant information without inserting unwanted traffic on the network. The VEC controller subscribes to BSs' updates in its region. Thus, the RS plays the role of Publisher, and the regional VEC controllers play the role of Subscribers. Based on the information about the BSs, the VEC controllers could starts the VCs formation process. In this sense, VCs can be classified in different regions according to BSs' positions in the city. Considering that the number of VCs is the same number of BSs in the scenario, we can denote a set of VCs by $v_j \in V = \{v_1, v_2, \ldots, v_m\}$, where m is the total number of VCs. A VC consists of a set of nodes (*i.e.*, vehicles and BS) that can share two types of computational resources ω and ϕ, namely CPU-cycle frequency (processing power) and storage capacity, respectively.

In this context, ω_{u_i} denotes the vehicle's CPU-cycle frequency, ϕ_{u_i} denotes the vehicle's storage capacity, ω_{b_y} denotes the BS's CPU-cycle frequency, and ϕ_{b_y} denotes the BS's storage capacity. Therefore, each VC is represented by a tuple $\{id, resources_{vehicles}, resources_{bs}, resources_{total}\}$, where *id* is the VC's unique identification, $resources_{vehicles}$ is the total resources of vehicles (ω_{u_i} and ϕ_{u_i}), $resources_{bs}$ is the total resources of BS (ω_{b_y} and ϕ_{b_y}), and $resources_{total}$ is the sum of resources in the VC (Ω_j and Φ_j), calculated as

$$\Omega_j = \sum_{i=1}^{x} \omega_{u_i} + \sum_{y=1}^{m} \omega_{b_y}, \quad \text{if } u_i, b_y \in v_j, \tag{1}$$

$$\Phi_j = \sum_{i=1}^{x} \phi_{u_i} + \sum_{y=1}^{m} \phi_{b_y}, \quad \text{if } u_i, b_y \in v_j. \tag{2}$$

In short, the total amount of processing power resources Ω_j and Φ_j of each VC v_j is the sum of the shared resources from each vehicle u_i and BS b_y belonging to a given VC v_j.

Figure 2 depicts the VCs formation process based on a spatial layer to add temporal information about vehicle mobility and their computational resources. We assume that BS is aware of vehicle mobility to add the temporal layer in the VC formation, which is possible to take advantage of beacons already exchanged by vehicles to obtain the vehicles' information, avoiding extra overhead. Specifically, each vehicle transmits periodic beacons containing its id, computational resources, speed, positioning, and route. In this sense, the BS

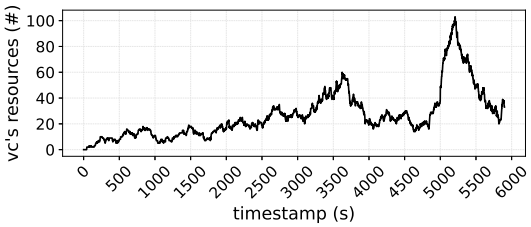


Fig. 3. Time series created by the system for a Vehicular Cloud (VC).

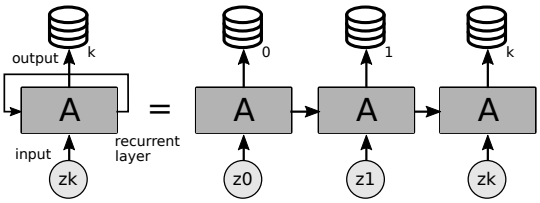


Fig. 4. Recurrent Neural Network (RNN) employed by MARINA.

receives such information and forwards it to the RS. Based on the information requested by the VEC controller to the RS, the VC formation process begins with mapping which BS holds which vehicles in its coverage (Label 1 in Figure 2). Given the history of associations in the BSs, it is possible to decompose this information into time series representing the time variation of vehicles in each VC (Label 2 in Figure 2). Spatiotemporal information on the dynamics of resources in each VC is created and stored in the regional VEC controller. Considering the importance of estimating the resources available in a VC to make task scheduling decisions with high precision, an LSTM-based prediction model is used in this step. With this, we can define a time window and estimate the resources available in each VC (Label 3 in Figure 2). Finally, a resource repository stores the predicted information, and the MARINA consults such spatiotemporal information for decision-making (Label 4 in Figure 2). Therefore, it is possible to estimate resources available in each VC in k time slots by considering the spatiotemporal information. In this sense, we can represent the resources available at VC in each $k \in K$ by Ω_{jk} and Φ_{jk}.

B. Mobility prediction model

We consider a mobility prediction process to add a temporal layer to the spatial information about vehicle mobility and their computational resources, which is essential in task scheduling scenarios for VCs environments. In this sense, we need to estimate the vehicles' dwell time in each VC, enabling to obtain information on the VCs' computational capacity in a predefined time window.

Several works consider Markov models, Extended Kalman Filter (EKF), Support Vector Regression (SVR), and Autoregressive Integrated Moving Average (ARIMA) models to predict vehicular mobility [35], [36]. However, neural networks have gained increasing attention from academia and industrial groups for the accurate predictions offered by their various models. In this scenario, an RNN is a deep learning approach that extends the traditional feed-forward networks with internal cycles [37]. These internal cycles allow tracking information sequences to create spatiotemporal knowledge through current and past inputs. LSTM is an advanced version of the RNN architecture developed to model chronological sequences and their long-term dependencies with greater precision [38].

In this sense, to employ an RNN in MARINA, we need to build a dataset containing resources available information in each VC. The model learns from past dynamics and accurately estimates the available resources in a given time window. To this end, the information about user association in a given BS is stored in the RS. In this way, when the VEC controller starts the VC formation process, it is easier to aggregate and create each VC's resources time series. Let $Z = \{z_0, z_1, \ldots, z_k\}$ be a vector that represents the dataset, in which each element consist of a tuple $z_k = \{timestamp, resources\}$ representing a simulation step and resources available in this step, for each VC, as shown in Figure 3.

In summary, the predictor model is defined as $f = \Upsilon \circ \Theta$, where $\circ$ indicates applying function Υ on function Θ's output. The feature learning machine $\Theta(\cdot)$, which converts inputs into features, is utilized to process the input data first. After that, the next step involves the representation function $\Upsilon(\cdot)$, which maps features into a prediction [17]. In this sense, the prediction process of the next available resources for each VC is given according to

$$z_{\Omega_j}^{k+1} = \Upsilon_k \circ \Theta_k(z_{\Omega_j}^k). \tag{3}$$

where $z_{\Omega_j}^{k+1}$ represents the next available resources of the VC j considering its previous observations $z_{\Omega_j}^k$.

Figure 4 shows how the RNN employed by MARINA works. In summary, the RNN receives an input z_k representing a set of tuples, and for each VC, it employs an LSTM as a recurrent layer. The dynamic employed by LSTM is to store past information in long-term memory to explore the internal relationships between each prediction. The information persists across the network and is used in comparisons to improve prediction estimates [39]. Finally, the output represents the future resources available in each VC given a time window k.

C. Problem Definition

Each task $t_l \in T = \{t_1, t_2, \ldots, t_n\}$ is denoted by a tuple $\{id_l^t, s_l^t, w_l^t, D_l^t\}$ where id_l^t means the unique task identification, s_l^t denotes the size of task input data, w_l^t is the numbers of CPU cycles required to complete the task, and D_l^t is the deadline constraint. In this way, the scheduling mechanism aims to optimize scheduling a set of tasks $T' \subset T$ to be processed in the available VCs without increasing monetary costs and scheduling as many tasks as possible.

The computational resources of the same VC are shared among different tasks scheduled in that cloud. Thus, Ω_j for calculating the computation delay for a given task must be updated according to the degree of sharing of that resource within the VC, represented by Ψ_j. The value of Ω_j is divided by the number of tasks $|T_j'|$ that have been scheduled for this VC, according to

$$\Psi_j = \frac{\Omega_j}{|T_j'|}. \tag{4}$$

According to the literature, all processes that add delay must be considered to compute the computing delay for a task scheduling in VC, [20], [21], [40]. For example, the transmission delay of the resource request, the scheduling delay between BS and VC, the task computation delay in the VC, and the entire reverse path until the requesting obtains the result of its task. However, this article only considers the task computation delay in the VC. This is because this metric simplifies the understanding of the efficiency of task scheduling solutions, given that the entire network infrastructure is the same for all approaches. The computation delay d_{lj}^t can be obtained based on the required CPU cycle w_l^t divided by the CPU-cycle Ψ_j of the server (VC), according to

$$d_{lj}^t = \frac{w_l^t}{\Psi_j}, \quad \forall\ t_l \in v_j. \tag{5}$$

As noted, each task has deadline constraints in its configuration, represented by D_l^t. Therefore, scheduling solutions must consider this metric to ensure that these restrictions are respected and tasks are successfully processed in the VCs. In this scenario, if $d_{lj}^t \leq D_l^t$, the task was successfully scheduled and executed in the v_j VC. Also, when a task is scheduled and starts to be processed, there is a cost associated with this execution. The monetary cost is modeled as

$$\mathcal{C}_l = d_{lj}^t \times \left(w_l^t \times Price(t_l)\right), \tag{6}$$

where d_{lj}^t is the t_l processing time in $v_j \in V$ and w_l^t is its CPU cycles required. $Price(t_l)$ is the resource price, calculated as

$$Price(t_l) = \begin{cases} 11.444 & \text{if } t_l \text{ uses } b_y\text{'s proc. resources,} \\ 5.016 & \text{if } t_l \text{ uses } u_i\text{'s proc. resources.} \end{cases} \tag{7}$$

These costs are based on instances with GPU capacity available on Amazon EC2[1], such as *g4ad* and *g3* for BS and vehicle, respectively.

When a task arrives in the system, the VEC controller must select the best VC to process this task. This selection must consider the monetary cost of using VC's resources. In this case, we must first apply a Pareto optimization for a joint minimization between processing time and task deadline. These metrics are directly related to monetary costs. Thus, the Pareto set allows us to find, among the queued tasks in the system, a set $\mathcal{P}$ that jointly minimizes the processing time and the deadline. With the Pareto set $\mathcal{P}$ defined, which has the tasks that imply a lower monetary cost, we can select from this set the tasks that will be processed in the VCs, represented by T'.

To coordinate this selection, we can reduce this problem as a Bin Covering Problem (BCP). BCP solves the items' packaging problem with different weights in a finite set of bins. Given a set of items, the BCP decides how many items can be stored in the same bin. The algorithm aims to maximize the number of items stored. However, given that BCP is a combinatorial NP-hard problem, we use an approximation heuristic First-Fit Decreasing (FFD) to find a solution to our problem instances in polynomial time [41]. With the FFD heuristic, they are sorted in non-increasing order of sizes before placing the items in bins. Each item attempts to be placed in the first bin that can accommodate this item. If no bin is found, a new bin is observed, and the item is put in this new bin. FFD can be implemented to have a running time of at most $\mathcal{O}(n \log n)$, where n is the number of items (tasks).

In this context, we formulate a task scheduling problem by maximizing the number of tasks served while minimizing the cost monetary of VC's resources used in this process. It should be noted that minimizing the monetary cost is provided by the step that finds the Pareto set $\mathcal{P}$. Furthermore, maximization is performed by selecting tasks in the given Pareto set. In this way, we can define the problem as follows:

$$\text{P1}: \text{maximize} \sum_{l=1}^{|\mathcal{P}|} t_l, \qquad l \in \mathcal{P}, \tag{8}$$

$$\text{subject to } d_{lj}^t \leq D_l^t, \qquad j \in V,\ l \in \mathcal{P}, \tag{9}$$

$$\sum_{l=1}^{|\mathcal{P}|} s_l^t \leq \Phi_{jk}, \qquad k \in K,\ j \in V,\ l \in \mathcal{P}, \tag{10}$$

$$\sum_{l=1}^{|\mathcal{P}|} w_l^t \leq \Omega_{jk}, \qquad k \in K,\ j \in V,\ l \in \mathcal{P}, \tag{11}$$

The constraint (9) guarantees that the task deadline is respected, avoiding rescheduling. The constraints (10) and (11) ensures that VCs' storage and processing limits are respected during the k required processing time intervals.

IV. Efficient Task Scheduling Mechanism for VEC Environments

This subsection describes an efficient task scheduling mechanism for VEC, called MARINA. We consider a scenario composed of multiple VCs coordinated by regional VEC controllers, which runs MARINA for task scheduling. MARINA finds the Pareto set to ensure monetary cost minimization and considers the BCP to resolve the problem in P1 of equations (8 et seq.).

A. MARINA's operations

In short, MARINA first searches for the Pareto set $\mathcal{P}$ using the joint minimization of task processing times and task deadlines as a criterion, creating a vector for each criterion (processing time and deadline). Therefore, the vectors are arranged in a 2-dimensional plane, and the Pareto set is found. Figure 5 presents an example of searching for the Pareto set in a queue with 24 tasks. We can obtain a Pareto solution set in 2-dimensional in polynomial time $\mathcal{O}(n \log n)$ [42]. This step ensures that the tasks selected to be verified with BCP already have joint minimization of estimated processing times and deadline constraints. In this case, this directly impacts the overall monetary cost.

As discussed earlier, the VEC environment is highly dynamic due to vehicular mobility. Therefore, many works model VEC environments as an M/M/1 queue to bring the system's dynamics closer to the real world [20], [43]. However, in realistic scenarios, several entities can process existing requests. Also, it is essential to consider that the service time of the

[1]https://aws.amazon.com/ec2/dedicated-hosts/pricing/

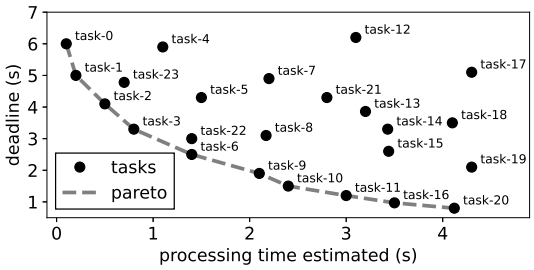


Fig. 5. Pareto set example with 24 tasks.

queued tasks is defined by the task requirements, such as size, required number of CPU cycles, and deadline constraints [20]. Thus, we consider that our VEC environment works as an M/G/z queue. The tasks arrive in the system following a Poisson process (M). The service time of the tasks is defined by their characteristics (that is, it follows a general distribution G). Finally, the system has more than one server (z) to attend to requests, which are the various VCs formed in the scenario.

When tasks arrive in the system, they are queued. Traditional approaches schedule tasks based on the organization in that queue. However, as the processing time and deadline of the tasks are crucial factors, MARINA prioritizes the minimization of these aspects in its decision-making process. That is, if a scheduling choice returns a lower processing time than another, the use of resources will also be minimized. In the same way, the monetary cost associated with the lower processing time will also be minimized. However, to perform this selection, MARINA needs to know the processing time of all queued tasks. In this step, Equation (5) estimates the processing time since it is unknown how many tasks will be scheduled in this VC.

Algorithm 1 shows how our VEC system works over time. Initially, the data structures that store the tasks in the system, the tasks in execution, and the VCs of the scenario are created (Lines 1 and 2). In a dynamic environment, tasks are generated independently, following an arrival rate that can be defined after observing the system's behavior. Thus, each time slot $k \in K$ and following a Poisson process, a set of tasks arrives in the system to be executed (Line 3). The task set is queued and goes on standby to be scheduled and executed (Line 4). For the task scheduling process to occur, it is necessary to know the VCs available in the scenario. Thus, if k is equal to the VC formation interval, the formation process occurs, and the VC information is maintained until the next interval (Lines 5 and 6). If the $\mathcal{Q}$ queue is not empty (Line 7), the queued tasks and the set of VCs are passed to the task scheduling mechanism (Lines 8 and 9) presented in Algorithm 2. After the scheduling process, verification is performed at each time slot k if the tasks in the $\mathcal{R}$ queue (run queue) reached their processing time in each VC (Lines 10 and 11). If the task has completed its execution, the monetary cost is calculated, and the task is removed from $\mathcal{Q}$ and $\mathcal{R}$ queues (Lines 12 to 15). Otherwise, the task execution estimate is updated, as other tasks may have left the system, and more resources may be available at the corresponding VC (Lines 16 and 17). We

Algorithm 1: Vehicular Edge Computing (VEC) environment

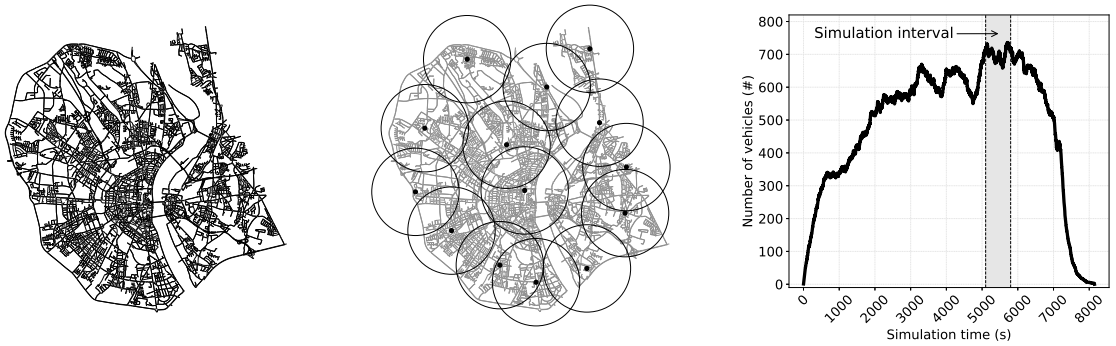
```
1  Q, R ← ∅                                  ▷ Waiting and Run queues
2  V, T ← ∅                                  ▷ VC and Task sets
3  foreach time slot k ∈ K do
4      Q.enqueue(T_k)
5      if k is update interval then
6          V ← update VCs with model in Section III-A
7      if Q ≠ ∅ then
8          T ← Q
9          R ← MARINA(T, V)                  ▷ Algorithm 2
10     if R ≠ ∅ then
11         foreach r ∈ R do
12             if r has completed its execution then
13                 compute cost with Equation (6)
14                 Q.dequeue(r)
15                 R.dequeue(r)
16             else
17                 Update r computation estimation
```

emphasize that Algorithm 1 is executed in the VEC controllers.

After a VC formation process, task scheduling is triggered, and information such as available VCs and the set of tasks is provided to the MARINA. In summary, Algorithm 2 describes the MARINA's operations in a VEC controller. In this sense, the controller gets the VC set V and task set T, which gives the T' task scheduling set as an output. The set V is sorted non-increasing, so VCs with more available resources are prioritized (Line 1). Two vectors are created based on T, the first PT for the estimated processing time, and the second D for deadlines (Lines 3 and 4). MARINA calls the procedure PARETOSET with configuration for joint minimization of vectors PT and D (Line 5). The procedure returns a set $\mathcal{P}$ containing the ID of the tasks that are part of the minimal Pareto set. With the Pareto set defined, the BCP uses this task subset to schedule tasks, considering other computational requirements of each task (Line 6). In this step, BCP returns a subset of candidate tasks S'. Also, the total number of resources needed for this returned set is calculated (Line 7). After that, for each task in the set, it is verified if the VC will have available resources until its deadline D_l^t (Lines 8 and 9). If not, that task is removed from the set S' and its required resources are removed from the total resource estimate (Lines 10 and 11). If so, its actual processing time is calculated (Line 13). If the processing time is longer than its deadline, the task is removed from S' and will be rescheduled in the next round. Otherwise, set S' is added to the scheduled tasks list T'.

B. Computational complexity

The MARINA's time complexity is analyzed as follows. MARINA has three main stages. The first stage runs in time $\mathcal{O}(n \log n)$ in the worst case, which is the time complexity to find Pareto set [42]. In the same direction, the BCP algorithm



(a) TAPAS Cologne submap (b) Base Stations (BSs) in the scenario (c) simulation interval considered

Fig. 6. Simulation scenario.

Algorithm 2: Abstraction of MARINA

Input: task set T and VC set V
Output: scheduled tasks T'

$V \leftarrow$ decreasing order of available resources
foreach $v_j \in V$ **do**
  $PT \leftarrow$ processing time (t_l, v_j) for each $t_l \in T$
  $D \leftarrow$ deadline of each $t_l \in T$
  $\mathcal{P} \leftarrow$ ParetoSet($\{PT, D\}$, sense=$[min, min]$)
  $S' \leftarrow$ BinCoveringProblem($\mathcal{P}, v_j$)
  $totalResources \leftarrow \sum_{l=1}^{|S'|} w_l^t, \ \forall t_l \in S'$
  foreach $t' \in S'$ **do**
    ▷ use predicted vehicular information
    if $totalResources < v_j$ *until* D_l^t **then**
      $S' \leftarrow S' \setminus \{t'\}$
      $totalResources \leftarrow totalResources - t'$
    else
      $d_{lj}^t \leftarrow$ Equation (5)
      if $d_{lj}^t > D_l^t$ **then**
        $S' \leftarrow S' \setminus \{t'\}$
      else
        $T' \leftarrow S'$
return T'

needs $\mathcal{O}(n \log n)$ to sort non-increasing order of items by sizes (CPU cycles) and to cycle through all the items to be checked. Also, the number of tasks decreases based on the select VC with maximum resources available, where the number of VCs is represented by m. In the worst case, the second and third stages have linear $\mathcal{O}(n)$ complexity to check the deadline and computation delay constraints for each temporarily scheduled task S'. Finally, a constant $\mathcal{F}$ is added that represents the computational complexity of the prediction step, which may vary according to the technique used. In summary, the algorithm's time complexity can be described as $\mathcal{O}(\max\{m\} + n \log n + \mathcal{F})$ in the worst case, with m being the number of VCs and n the number of tasks. In this context, MARINA is a polynomial-time algorithm.

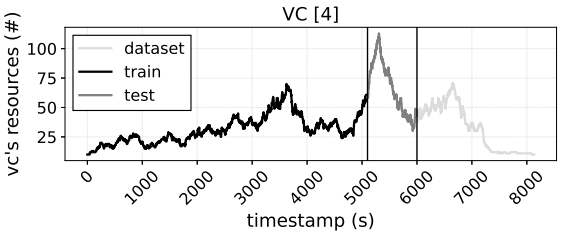


Fig. 7. Example of the data used for the training and prediction processes.

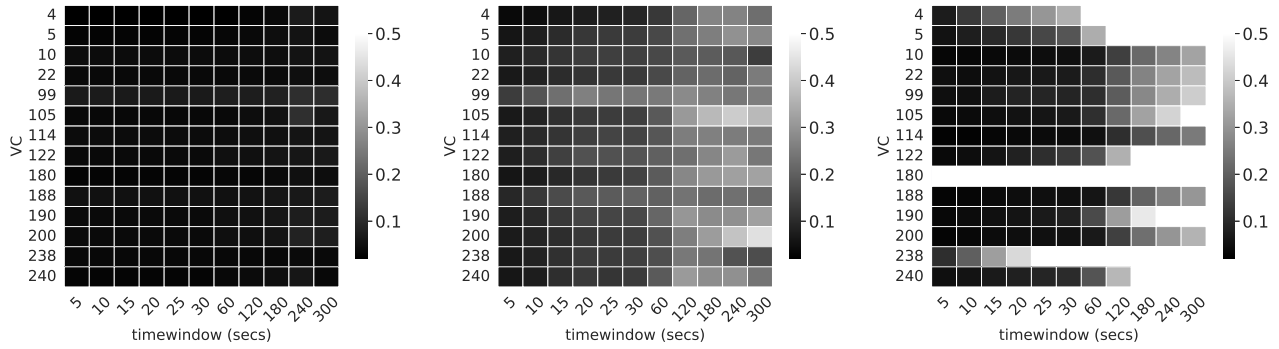
V. Evaluation

This section describes the methodology and metrics used to evaluate MARINA performance in a VEC environment. First, we show the simulation environment, including implementation, parameters, and evaluation metrics. Second, to better understand the resource prediction model used, we present LSTM results compared to other models in the literature. Finally, we present and discuss the results of task scheduling using the prediction model employed and the main insights.

A. Simulation environment

The experiments were carried out with the Simulator of Urban Mobility (SUMO), in version 1.11.0. The algorithms were implemented in Python 3.8 and connected to SUMO through the TraCI interface. We considered a deterministic realistic mobility trace from TAPAS Cologne[2] project, which reproduces vehicle traffic in the city of Cologne, Germany, as shown in Figure 6(a). The trace contains vehicular mobility from 6 to 8 AM on a typical working day and covers a region of $400\,\mathrm{km}^2$. However, only a city submap with $114\,\mathrm{km}^2$ was picked for our simulation experiments because it contains a greater variability of vehicles over time and up to 700 vehicles at peak times. The simulation time was $700\,\mathrm{s}$, with 100 initial seconds of warm-up, as shown in Figure 6(c). The simulations were run 33 times to obtain a $95\,\%$ confidence interval.

[2]http://kolntrace.project.citi-lab.fr/



(a) LSTM (b) DNN (c) SVR

Fig. 8. Results of resource predictions by Long Short-Term Memory (LSTM), Dense Neural Network (DNN), and Support Vector Regression (SVR).

TABLE III
SIMULATION PARAMETERS

Parameter description	Value
BS's wireless communication range	2000 m
Vehicle's wireless communication range	250 m
Mean size of the tasks	[1,10] MByte
Required CPU of the tasks	[1,30] MI
Task delay constraint	3, 5, and 7 s
Tasks distribution	Poisson
Tasks arrival rate λ	1, 3, and 5 tasks/second
Computational capability of each BS	15 MIPS
Computational capability of each vehicle	1 MIPS
Number of vehicles	550 $\sim$ 700
Simulation time	700 s
Scenario area	114 km^2
Number of VEC controllers	4
RS's communication latency	[1,5] ms
Recurrent layer	Bidirectional LSTM
Loss function	Mean Square Error (MSE)
Batch size	64
Sequence length	5
# of LSTM cells	128
Number of epochs	100

The Bag-of-Tasks (BoT) applications were considered since they have no dependence on each other and can be executed out of the submission order. The tasks' deadline varies between 3, 5, and 7 seconds. This is important to generalize the representation of possible application classes. The VC formation interval was defined in 5 seconds. The scenario considers different task arrival rates (*i.e.*, 1, 3, and 5 tasks/second) following a Poisson distribution [20], [25], [40]. Also, the size s_l^t of each task is $[1, 10]$ MByte. The number of CPU cycles w_l^t required to complete the task is fixed in $[1, 30]$ Million of Instructions (MI). We consider that a vehicle u_i makes available resources (CPU-cycle capacity and storage capacity) in 1 unit/vehicle, the number of CPU cores ω_{u_i} (CPU-cycle capacity in Million of Instructions Per Second (MIPS)), which without loss of generality is proportional to the storage capacity ϕ_{u_i} (1 MByte). In this way, we can get an idea of the impact that sharing lower resource units employ on the system. The communication ranges of vehicles and BSs were 250 m and 2000 m, respectively.

In addition, we consider 14 BSs, and each one is capable of sharing processing and storage resources, where such values were configured at 15 MIPS and 15 MByte, respectively. We deployed 4 VEC controllers, and each can manage up to 4 neighboring BSs. We deployed the BSs in the city following positioning information provided by the TAPAS Cologne project, as shown in Figure 6(b). Moreover, we used the TensorFlow framework version 2.8.2 to implement the RNN [44]. We also consider a Graphics Processing Unit (GPU) NVIDIA(R) Tesla V100 with 5120 CUDA cores and 32 GB of VRAM to train the ML models. Table III summarizes the main simulation parameters.

We considered three scheduling mechanisms to compare their performance with MARINA, namely: (i) *UNC* [23] task scheduling scheme is a classic queuing theory algorithm that is widely used as a policy for scheduling tasks in computational systems. It is similar to *First-Come-First-Serve* (FCFS) scheme, but the *UNC* scheme is free to select any task in the task queue for scheduling during the current time. (ii) *FORESAM* [24] considers a multi-criteria analytical method to select the most appropriate VC to receive the task. *FORESAM* considers all task requirements. (iii) *CRATOS* [13] considers a combinatorial optimization approach to schedule tasks in the available VCs. The algorithm was adapted for our scenario. The value of each task was defined as 1, given that, by default, the tasks have no value associated with them.

We consider the following evaluation metrics:

- *Root Mean Square Error (RMSE)*: quantifies the difference between ground-truth and predicted data regarding resource availability in each VC. This metric is widely used to measure the performance of predictors. In our case, it was used to evaluate and decide the best model for predicting vehicular resources for the proposed task scheduling mechanism.
- *Scheduled tasks (%)*: percentage of successfully scheduled tasks. We consider successful scheduling when a task is scheduled in a VC and can be executed respecting its deadline constraint.
- *Monetary Cost ($/time)*: represents the monetary cost of using VC's computational resources for k units of time. As defined in Section III-C, a VC comprises vehicles and BS, each of which has a different monetary cost.
- *System latency (s)*: refers to the processing time of the task in a given computational configuration plus the queue

waiting time. That is, this metric shows how efficient the decisions made by the mechanisms are.
- *CPU time (s)*: represents the sum of the execution time of all the processes involved in the mechanism. This time may vary depending on the machine's configuration used in the evaluation. For this evaluation, we used an Intel(R) Xeon(R) CPU X5650 (24×2.66GHz) with Linux architecture x86_64.

B. Results: Mobility prediction

We used $85\,\%$ of the samples of the data for training and $15\,\%$ for testing to analyze the RNN performance in our scenario. The prediction performance of the RNN was measured in terms of RMSE compared to Dense Neural Network (DNN) and Support Vector Regression (SVR), which are two approaches widely used in state-of-the-art to predict time series. Figure 7 shows an example of a time series built for VC with ID number 4. As can be seen, most of the time series ($85\,\%$) is used for model training, with the remainder ($15\,\%$) being used to test the predictions. Given the characteristics of the mobility trace, records from $6000\,\mathrm{s}$ were disregarded as they only contain the dynamics of removing vehicles from the simulation.

Figure 8 compares the RMSE obtained in the predictions with the considered models. This assessment considers the average RMSE among the 14 VCs in the scenario in different time horizons for prediction. At this stage, we performed an exploratory assessment of the size of the prediction window. This is important to show the degradation of the prediction as the prediction window increases, which is expected in this type of evaluation. Thus, for each VC, we consider the prediction windows 5, 10, 15, 20, 25, 30, 60, 120, 180, 240, and 300 seconds. In summary, the hyperparameters for training the LSTM model are: number of epochs = 100, batch size = 64, and number of LSTM cells = 128.

In this sense, we can see that the predictions provided by LSTM achieve an average RMSE of about 0.1 in most prediction windows, while the predictions provided by DNN and SVR suffer more significant degradations as the prediction window increases. LSTM is generally more efficient than other models at considering past events to provide predictions closer to dataset ground truth. Thus, this type of RNN proved ideal for helping our task scheduling mechanism. Based on the results, the LSTM proved to be more robust in resource prediction phase. Thus, we chose to use it to help our task scheduling process with the time window setting equal to $5\,\mathrm{s}$.

C. Results: Task scheduling

Figure 9 shows the behavior of the waiting queue Q over the simulation time considered if no scheduling mechanism is used. Different arrival rates significantly increase the difficulty of orchestrating the task schedule process. For example, at the highest arrival rate ($\lambda = 5$), 4500 tasks are queued at the end of the simulation run. In other words, it is a challenging scenario, considering that task sizes and deadline constraints also vary according to their arrival in the system.

Figure 10 shows the difference in packets sent on the network, considering the centralized and hybrid architectures. In this sense, in a centralized architecture, where one entity makes all the decisions and builds its global knowledge, all vehicles must maintain communication with that entity. Thus, the number of packets sent per time unit is directly related to the number of vehicles communicating with the central entity. However, in a hybrid architecture, intermediary entities aggregate the messages of vehicles in their coverage and send this information to the remote server in a single data stream. The number of data messages transiting the network core is significantly reduced. In this evaluation, we can observe an average reduction of $50\,\%$ in the number of sent packets on the network when the hybrid architecture is considered.

Figure 11 displays the percentage of successfully scheduled and executed tasks with different maximum delay constraints 3, 5, and 7 seconds, respectively. We can check the behavior of the mechanisms when the system receives a high load of requests. All mechanisms improve their performance as the deadline increases. However, we can see that they all have difficulty scheduling as the task arrival rate increases. In all configurations MARINA performs better than other mechanisms. Also, when the arrival rate is equal to 1, MARINA can schedule over $91\,\%$ of the tasks in the configurations with the largest maximum deadline constraint, as shown in Figure 11(a). Both FORESAM and UNC operate similarly in this configuration. CRATOS has the worst performance when the deadline increases. This is due to its selection strategy, which is only concerned with the task size, not considering fundamental aspects such as deadline and computation delay. The tasks scheduled with CRATOS have processing restrictions that are not considered in its decision-making process. In the configuration with a task arrival rate equal to 3, MARINA already starts to have difficulty scheduling when the deadline grows, as shown in Figure 11(b). However, the MARINA still manages to schedule up to $90\,\%$ of tasks in scenarios with the highest deadline constraints. In this configuration, we see a more significant difference between FORESAM and UNC due to UNC applying the simple greedy policy. In this scenario, MARINA is superior in all configurations considered. Finally, when the task arrival rate is 5, MARINA maintains its results closer to $88\,\%$. FORESAM has results close to MARINA in this scenario. In FORESAM's setup, the main factor taken into account in its decision process was the task deadline, so when the deadline increase, its scheduling choices cannot relate to all the problem restrictions well. Even applying a simple scheduling method, the UNC can handle many requests compared to CRATOS.

Figure 12 shows the monetary cost of using the computational resources of the VCs. As mentioned in Section III-C, the cost of the vehicle's resources is less than that of the BS. Therefore, to minimize this cost, the approaches choose to select the resources of the vehicles for the scheduling first. We can see that MARINA minimizes the monetary cost in all evaluations. In this case, it is natural for the performance of all mechanisms to fall due to the percentage of scheduled tasks that also decreases in more challenging scenarios, as shown in Figure 11. The best performance of MARINA occurs due

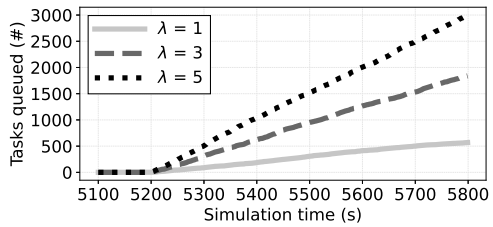


Fig. 9. Example of task queue Q over the simulation time considered.

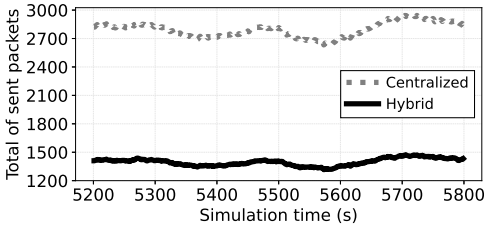
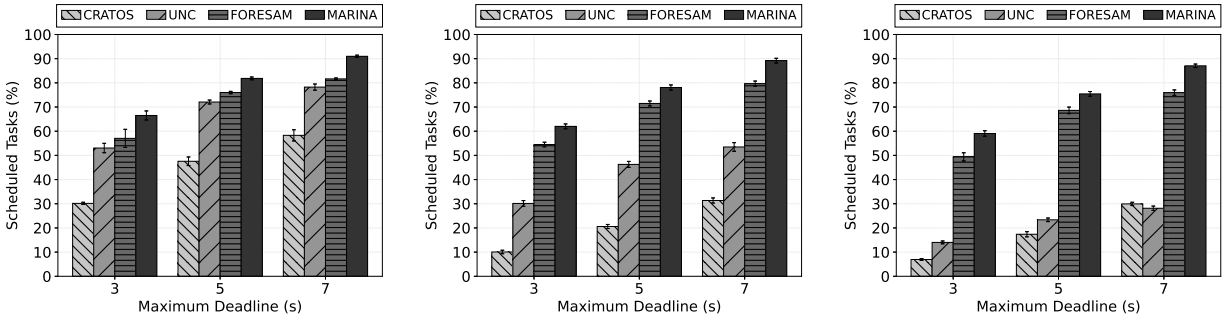
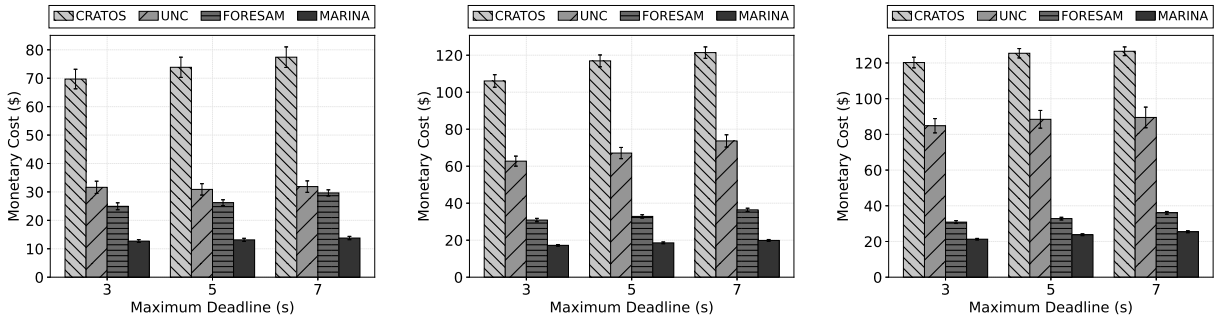


Fig. 10. Total sent packets on the network considering different system architectures.



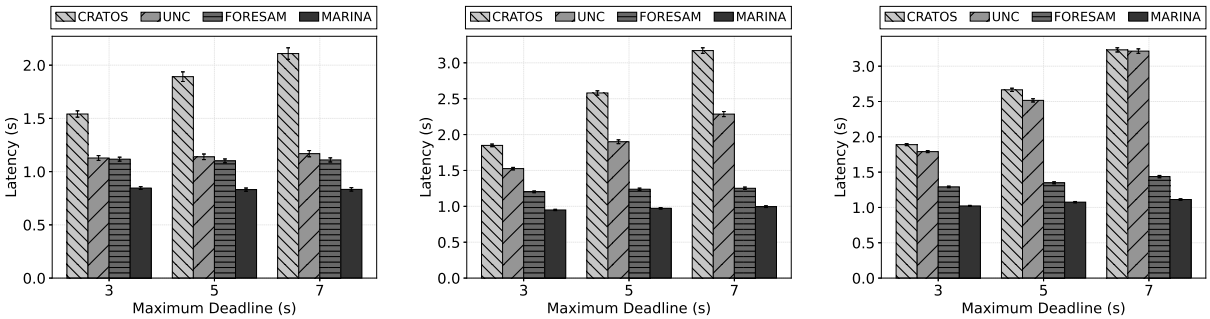
(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 11. Results of the scheduled tasks with different maximum deadline constraints and task arrival rates.



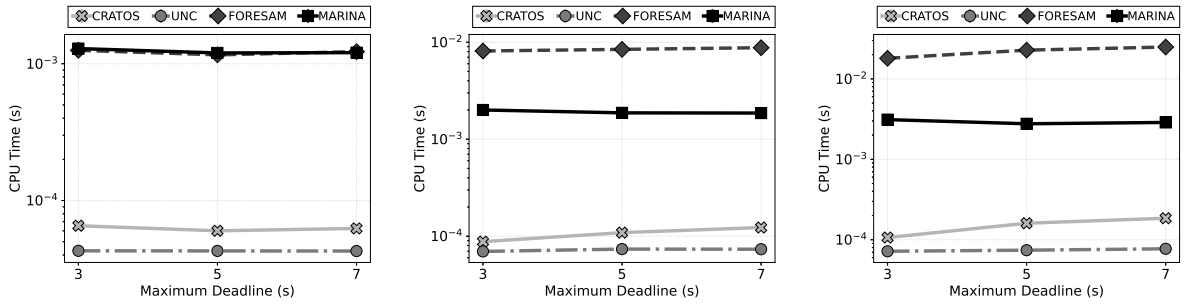
(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 12. Results of the monetary cost with different maximum deadline constraints and task arrival rates.



(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 13. Results of the system latency with different maximum deadline constraints and task arrival rates.



(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 14. Results of the CPU time with different maximum deadline constraints and task arrival rates.

to the selection of the tasks considering the VC's predicted available resources and the joint minimization provided by the Pareto set approach. Applying BCP allows a more significant number of tasks to be scheduled out in the same VC, prioritizing that such resources come from vehicles and not from BS. In addition, the more challenging the scenario (increasing the task arrival rate) becomes, the more costly the scheduling process becomes. More computational resources need to be used to run the existing tasks. MARINA reduces the monetary cost by up to $80\,\%$ in all scenarios. CRATOS proves to be more costly because its decision process involves only the amount of computational resources available. In this way, many tasks are rescheduled during the mechanism's operation, thus increasing its final monetary cost. The UNC maintains its expected behavior of increasing the monetary cost as deadline constraints increases. The application of Pareto set and BCP by MARINA allows the best task set to be scheduled in the same VC with minimum processing time and deadline constraints.

Figure 13 depicts the system latency results, including the computational delay and queue time. This metric is important because it shows the impact of scheduling tasks in a given processing configuration and is directly associated with how the scheduling approach selects VCs. We can see, in all evaluations, that MARINA reduces the computation delay of the tasks in the VCs. Hence, it is mainly due to selecting the VCs employed by MARINA, filtering based on the VC's processing rate considering the task deadline constraint. In addition to selecting the VC with greater computational capacity, selecting tasks based on the joint minimization of processing time and deadline helps reduce system latency. In addition, CRATOS and UNC employ a higher computation delay in all assessments due to their decision strategy that considers only the order in which tasks arrive in the system. Specifically, in the configuration with a tasks arrival rate equal to 3 and 5, MARINA and FORESAM stabilize their performance due to the absence of considerable variation in the number of tasks scheduled, as shown in Figures 13(b) and 13(c). In summary, MARINA has better managed VC's resources when considering aspects of both tasks and VCs in its decision-making process. In other words, an efficient task scheduling approach should maximize the number of scheduled tasks while minimizing the time that such tasks await their schedule.

Finally, Figure 14 presents the CPU time by the VEC controllers that run the scheduling approaches. This metric is directly related to the approach's computational complexity. Also, an approach that manages to schedule more tasks has more overall CPU time, even with less computational complexity. In all evaluations, UNC has lower CPU time because its selection method is simple, selecting the task to be scheduled based on its order of arrival in the system, operating with time complexity $\mathcal{O}(n^2)$, where n is the total number of tasks. CRATOS has the second-lowest CPU time, but this can be justified by the number of tasks scheduled and successfully executed, even having pseudo-polynomial complexity ($\mathcal{O}(\max\{m\} + n \times W)$, where m is the set VC, n is the number of tasks, and W is the size of the VC considered in each round). FORESAM has a high CPU time because it iteratively selects one task at a time given a VC, making a more significant number of checks for each VC considered in the round. FORESAM uses the AHP technique in its decision process, and the time complexity of AHP is $\mathcal{O}(\min\{mn^2, m^2n\})$, where m is alternatives, and n is criteria. Finally, MARINA has CPU time statistically close to FORESAM. This is also true of its iterative decision-making process. In certain rounds, the returned Pareto set may be small, requiring further rounds to fill the VC. A 2-dimensional set is constructed at each decision round, and the algorithm searches for the Pareto optimal set, taking $\mathcal{O}(n \log n)$ time.

VI. Conclusion

This article introduced MARINA, an efficient task scheduling for VEC environments. MARINA divides its scheduling process into three stages. The first step is finding the Pareto set based on a joint minimization of processing time and deadline constraints. After that, it applies BCP with the FFD heuristic to select a set of tasks for a given VC, aiming to maximize the number of tasks scheduled and minimize the monetary cost of this processing. The second step selects the tasks based on the correlation between the tasks' deadline constraint and predicted information about the computational resources available in each VC. Finally, the third step verifies the processing time of these tasks in the selected VC to reduce the computation delay and, consequently, the monetary cost of using the computational resources. MARINA employs an RNN architecture to predict vehicular resources in VCs and assist in its decision-making process.

Simulation results in a realistic vehicular scenario show that, compared to other solutions, MARINA has high performance in scheduling a higher number of tasks at lower monetary cost of using resources and lower overall system latency. It also keeps the CPU usage time around $10\,\mathrm{ms}$ in the regional VEC controllers.

During the task scheduling process, the system only has a macroscopic view of the VCs, so our objective is to have a microscopic view of the task management within each VC and increase the fault tolerance from MARINA. Besides, it is essential to highlight that the proposed solution can be extended to consider different metrics, such as energy consumption, load balancing, and reward for sharing resources. As future work, in addition to the different decision metrics, we plan to consider a more heterogeneous environment, including electric vehicles and their specific energy consumption models and mobile devices such as Unmanned Aerial Vehicles (UAVs). Also, we plan to consider load balancing in the task scheduling process to distribute available resources evenly. Additionally, we could consider migrating computational resources to increase reliability in maintaining the task scheduling process. Finally, we intend to consider a network environment where other applications use the VEC controller to verify the impact of the task scheduling in a shared resources environment.

REFERENCES

[1] A. Waheed, M. A. Shah, A. Khan, S. ul Islam, S. Khan, C. Maple, and M. K. Khan, "Volunteer Computing in Connected Vehicles: Opportunities and Challenges," *IEEE Network*, vol. 34, pp. 212–218, 2020.

[2] I. W. Damaj, D. K. Serhal, L. A. Hamandi, R. N. Zantout, and H. T. Mouftah, "Connected and Autonomous Electric Vehicles: Quality of Experience survey and taxonomy," *Vehicular Communications*, vol. 28, p. 100312, Apr. 2021.

[3] C. A. Brennand, F. D. da Cunha, G. Maia, E. Cerqueira, A. A. Loureiro, and L. A. Villas, "Fox: A traffic management system of computer-based vehicles fog," in *2016 IEEE symposium on computers and communication (ISCC)*. IEEE, 2016, pp. 982–987.

[4] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–46, Jan. 2023.

[5] S. Zhang, J. Liu, H. Guo, M. Qi, and N. Kato, "Envisioning Device-to-Device Communications in 6G," *IEEE Network*, vol. 34, no. 3, pp. 86–91, May 2020.

[6] G. P. Rocha Filho, R. I. Meneguette, J. R. T. Neto, A. Valejo, L. Weigang, J. Ueyama, G. Pessin, and L. A. Villas, "Enhancing intelligence in traffic management systems to aid in vehicle traffic congestion problems in smart cities," *Ad Hoc Networks*, vol. 107, p. 102265, 2020.

[7] Q. Luo, C. Li, T. Luan, and W. Shi, "Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing," *IEEE Transactions on Services Computing*, vol. 1374, pp. 1–12, 2021.

[8] C. A. Brennand, A. Boukerche, R. Meneguette, and L. A. Villas, "A novel urban traffic management mechanism based on fog," in *2017 IEEE symposium on computers and communications (ISCC)*. IEEE, 2017, pp. 377–382.

[9] A. Boukerche and V. Soto, "Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, Jul. 2021.

[10] A. Masood, D. S. Lakew, and S. Cho, "Security and Privacy Challenges in Connected Vehicular Cloud Computing," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2725–2764, 2020.

[11] A. Katiyar, D. Singh, and R. S. Yadav, "State-of-the-art approach to clustering protocols in VANET: a survey," *Wireless Networks*, vol. 26, no. 7, pp. 5307–5336, Jun. 2020.

[12] S. Olariu, "A Survey of Vehicular Cloud Research: Trends, Applications and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2648–2663, Jun. 2020.

[13] J. B. D. da Costa, R. I. Meneguette, D. Rosário, and L. A. Villas, "Combinatorial Optimization-based Task Allocation Mechanism for Vehicular Clouds," in *IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, May 2020.

[14] J. B. D. da Costa, A. M. de Souza, D. Rosário, E. Cerqueira, and L. A. Villas, "Efficient data dissemination protocol based on complex networks' metrics for urban vehicular networks," *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 1–13, Aug. 2019.

[15] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–33, Sep. 2020.

[16] A. M. de Souza, H. F. Oliveira, Z. Zhao, T. Braun, L. Villas, and A. A. F. Loureiro, "Enhancing Sensing and Decision-Making of Automated Driving Systems With Multi-Access Edge Computing and Machine Learning," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 1, pp. 44–56, Jan. 2022.

[17] A. M. de Souza, T. Braun, L. C. Botega, L. A. Villas, and A. A. F. Loureiro, "Safe and Sound: Driver Safety-Aware Vehicle Re-Routing Based on Spatiotemporal Information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3973–3989, Sep. 2020.

[18] A. Ip, L. Irio, and R. Oliveira, "Vehicle Trajectory Prediction based on LSTM Recurrent Neural Networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, Apr. 2021.

[19] B. Brik and A. Ksentini, "Toward Optimal MEC Resource Dimensioning for a Vehicle Collision Avoidance System: A Deep Learning Approach," *IEEE Network*, vol. 35, no. 3, pp. 74–80, May 2021.

[20] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 2, pp. 598–611, Feb. 2022.

[21] L. Chen and J. Xu, "Task Replication for Vehicular Cloud: Contextual Combinatorial Bandit with Delayed Feedback," in *IEEE Conference on Computer Communications (INFOCOM 2019)*. IEEE, Apr. 2019, pp. 748–756.

[22] I. Sorkhoh, D. Ebrahimi, R. Atallah, and C. Assi, "Workload Scheduling in Vehicular Networks With Edge Cloud Capabilities," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8472–8486, Sep. 2019.

[23] G. Hattab, S. Ucar, T. Higuchi, O. Altintas, F. Dressler, and D. Cabric, "Optimized Assignment of Computational Tasks in Vehicular Micro Clouds," in *2nd International Workshop on Edge Systems, Analytics and Networking (EdgeSys 2019)*. ACM, 2019, pp. 1–6.

[24] R. Pereira, A. Boukerche, M. A. C. da Silva, L. H. V. Nakamura, H. Freitas, G. P. Rocha Filho, and R. I. Meneguette, "FORESAM—FOG Paradigm-Based Resource Allocation Mechanism for Vehicular Clouds," *Sensors*, vol. 21, no. 15, p. 5028, Jul. 2021.

[25] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, "A Probabilistic Approach for Cooperative Computation Offloading in MEC-Assisted Vehicular Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 899–911, Feb. 2022.

[26] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.

[27] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Transactions on Vehicular Technology*, pp. 1–12, 2022.

[28] S. Misra and S. Bera, "Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2071–2078, Feb. 2020.

[29] X. Wu, S. Zhao, R. Zhang, and L. Yang, "Mobility Prediction-Based Joint Task Assignment and Resource Allocation in Vehicular Fog Computing," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, May 2020.

[30] S. A. Kazmi, T. M. Ho, T. T. Nguyen, M. Fahim, A. Khan, M. J. Piran, and G. Baye, "Computing on wheels: A deep reinforcement learning-based approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22 535–22 548, 2022.

[31] J. Leng, D. Yan, Q. Liu, K. Xu, J. L. Zhao, R. Shi, L. Wei, D. Zhang, and X. Chen, "Manuchain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 182–192, 2019.

[32] X. Yan, M. Ma, and R. Su, "Efficient group handover authentication for secure 5g-based communications in platoons," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[33] X. Tian, B. Zhang, and C. Li, "Throughput-Optimal Dynamic Broadcast for SINR-Based Multi-Hop Wireless Networks With Time-Varying

Topology," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11 962–11 975, Nov. 2021.
[34] Z. Li, C. Wang, and C.-J. Jiang, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2217–2228, 2017.
[35] P. Sun, N. Aljeri, and A. Boukerche, "Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks," *IEEE Network*, vol. 34, no. 3, pp. 178–185, May 2020.
[36] I. Rasheed and F. Hu, "Intelligent super-fast Vehicle-to-Everything 5G communications with predictive switching between mmWave and THz links," *Vehicular Communications*, vol. 27, p. 100303, Jan. 2021.
[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[38] H. Guo, L.-l. Rui, and Z.-p. Gao, "V2v task offloading algorithm with lstm-based spatiotemporal trajectory prediction model in svcns," *IEEE Transactions on Vehicular Technology*, pp. 1–16, 2022.
[39] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[40] Y. Sun, S. Zhou, and Z. Niu, "Distributed Task Replication for Vehicular Edge Computing: Performance Analysis and Learning-Based Algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1138–1151, Feb. 2021.
[41] P. Schwerin and G. Wäscher, "The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP," *International Transactions in Operational Research*, vol. 4, no. 5–6, pp. 377–389, Nov. 1997.
[42] S. Borzsony, D. Kossmann, and K. Stocker, "The Skyline operator," in *17th International Conference on Data Engineering*. IEEE, 2001, pp. 421–430.
[43] V. Sethi and S. Pal, "Feddove: A federated deep q-learning-based offloading for vehicular fog computing," *Future Generation Computer Systems*, vol. 141, pp. 96–105, 2023.
[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

Joahannes B. D. da Costa received the B.Sc. degree in Information Systems from the Federal University of Para (UFPA), Brazil, in 2016. He received the M.Sc. degree in Computer Science from the Federal University of Para (UFPA), Brazil, in 2018. He is currently doing a Ph.D. in Computer Science from the University of Campinas (UNICAMP), Brazil. His research interests include machine learning, intelligent transportation systems, vehicular clouds, resource allocation, and task scheduling.

Allan M. de Souza received his Ph.D. degree in computer science from University of Campinas, Brazil and University of Bern, Switzerland, in 2021. He also received his M.Sc degree in computer science from University of Campinas, in 2016. Currently, he is postdoctoral research at University of Campinas and his research interests are in the field of data dissemination, congestion detection, congestion control and re-routing in VANETs and Traffic Management Systems.

Rodolfo I. Meneguette is a professor at University of São Paulo (USP). He received his Bachelor's degree in Computer Science from the Paulista University (UNIP), Brazil, in 2006. He received his master's degree in 2009 from the Federal University of São Carlos (UFSCar). He received his doctorate from the University of Campinas (Unicamp), Brazil, in 2013. In 2017 he did his post-doctorate in the PARADISE Research Laboratory, University of Ottawa, Canada. His research interest are in the areas of vehicular networks, resources management, flow of mobility, and vehicular clouds.

Eduardo Cerqueira received the Ph.D. degree in informatics engineering from the University of Coimbra, Portugal, in 2008. He is currently an Associate Professor with the Faculty of Computer Engineering, Federal University of Para (UFPA), Brazil, and an Invited Researcher with the Network Research Laboratory, UCLA, USA, and the Centre for Informatics and Systems, University of Coimbra, Portugal. His publications include five edited books, five book chapters, four patents, and over than 180 articles in national/international refereed journals/conferences. His research interests include multimedia, future Internet, quality of experience, mobility, and ubiquitous computing. He has been serving as a guest editor for six special issues of various peer-reviewed scholarly journals.

Denis Rosário received the Ph.D. degree in electrical engineering from the Federal University of Pará, Brazil, with joint supervision undertaken by the Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland, in 2014. He is currently a Professor with the Federal University of Pará. His current research interests include multimedia, wireless networks, FANET, vehicular networks, vehicular clouds, mobility, quality of experience, and software defined networks.

Christoph Sommer is a full professor and holds a chair at the Faculty of Computer Science, School of Engineering Sciences, TU Dresden, and is heading the Networked Systems Modeling group. He received his Ph.D. degree in engineering (Dr.-Ing., with distinction) and his M.Sc. degree in computer science (Dipl.-Inf. Univ.) from the University of Erlangen in 2011 and 2006, respectively. His research is focused on protocol and system designs of wirelessly connected mobile systems exhibiting high topology dynamics. He also authored the Cambridge University Press textbook Vehicular Networking.

Leandro Villas finished his doctoral degree in Computer Science at the Federal University of Minas, Brazil, in 2012. He had the doctoral thesis selected among the top 6 in the Brazilian Computer Society. He was awarded the Excellence Research Award from PARADISE Laboratory - University of Ottawa. Currently, he is a CNPq researcher level 1D, Associate Professor of Computer Science at the Institute of Computing at the University of Campinas and coordinator of the Hub of Artificial Intelligence and Cognitive Architectures at Unicamp[3]. Dr. Villas has already published 80+ papers in international journals and 160+ in conferences. Nine of those papers received the best paper award in IEEE/ACM and SBC conferences. He was recently named among the Top 1% Scientists with the most significant impact in the world, according to a study by Stanford Univ./Plos Biology.

[3]https://hiaac.unicamp.br/en/