Efficient Pareto Optimality-based Task Scheduling for Vehicular Edge Computing

Joahannes B. D. da Costa*†, Allan M. de Souza*, Denis Rosário‡, Christoph Sommer†, Leandro A. Villas*
*University of Campinas (UNICAMP), Institute of Computing (IC), Brazil
‡Federal University of Pará (UFPA), Institute of Natural Sciences (ICEN), Brazil
†TU Dresden, Faculty of Computer Science, Germany
Contact: {joahannes.costa, allan.souza, leandro}@ic.unicamp.br, denis@ufpa.br, cms-labs.org/people/sommer

***Abstract*—Vehicular Edge Computing is a promising paradigm that provides cloud computing services closer to vehicular users. Vehicles and communication infrastructure can cooperatively provide vehicular services with low latency constraints through vehicular cloud formation and use of these computational resources via task scheduling. An efficient task scheduler needs to decide which cloud will run the tasks, considering vehicular mobility and task requirements. This is important to minimize processing time and, consequently, monetary cost. However, the literature solutions do not consider these contextual aspects together, degrading the overall system efficiency. This work presents EFESTO, a task scheduling mechanism that considers contextual aspects in its decision process. The results show that, compared to state-of-the-art solutions, EFESTO can schedule more tasks while minimizing monetary cost and system latency.**

I. Introduction

Vehicles are becoming more autonomous, intelligent, and connected [1]. In this context, the demand for vehicle-based services will grow and increase bandwidth consumption, which can cause network core congestion and data delivery failures [2]. In this sense, it is essential to guarantee low latency levels for such services to meet the application demands with real-time requirements, such as mechanisms based on machine learning or services oriented to Artificial Intelligence for connected vehicles. However, meeting these restrictions is not straightforward due to the vehicular network dynamics [3].

The Vehicular Edge Computing (VEC) paradigm emerges by baking advantage of the vehicular communication capacity provided by Vehicular Networks (VANETs) and the computational power also included in modern connected vehicles. VEC aims to use computational resources in vehicles and communication infrastructures as cloud services, bringing computational power closer to vehicular users [4], [5]. In a VEC architecture, there is a network controller for each predefined region of the city, called a VEC controller, which builds and maintains knowledge about vehicles through Vehicle-2-Everything (V2X) communication between vehicles and communication infrastructures, *i.e.*, Base Stations (BSs), allowing more dynamic and precise control rules [2]. In this way, the VEC controller forms Vehicular Clouds (VCs) with computational resources available in vehicles and BSs.

The VEC controller can run VC Formation, and Task Scheduling processes [3]. In the first process, the VCs are formed by grouping and managing of computational resources (vehicles and BSs), specifically processing and storage resources. The second process refers to efficiently use these computational resources by means of a task scheduling mechanism, which will be focus on this work. In short, task scheduling makes it possible to use resources transparently by applications/services that require computational power above the locally supported [5].

However, due to common properties of VANETs (*i.e.*, high vehicular mobility and intermittent connections), performing task scheduling in VCs has some challenges. In this sense, at least three points must be taken into account for efficient task scheduling in VCs: *(1) High Vehicular Mobility* causes unexpected disconnections between vehicles and VEC controllers, causing interruptions and delays in return for scheduling results. With this, vehicular mobility information can assist in selecting more stable VCs in this process, with less variation in its resources over time. *(2) Deadline Constraints*, where after processing the task in a VC, the result must be returned before a specific deadline. Otherwise, this result becomes useless [6]. Therefore, scheduling must consider task deadline constraints in their decision process to increase its success rates. *(3) Monetary Cost*, where it is necessary to pay for the use of another computational resource. Therefore, scheduling mechanisms must always minimize costs for end-users.

In this context, considering mobility aspects, deadline constraints, and monetary costs for decision-making in a task scheduling process is an open point in the literature. Observing vehicular mobility is fundamental to estimating the future availability of resources in each VC. In the same direction, the tasks' deadline constraints can be better observed and considered from a resource availability estimate in the VCs, making it possible to infer the processing time and the service probability. Finally, the monetary cost estimates can be minimized by knowing the processing time of the tasks.

Considering the challenges mentioned, this paper presents *EFESTO* (EFficient parEto optimality-baSed Task scheduling for vehicular edge cOmputing). The mechanism runs on VEC controllers to select tasks for scheduling based on Pareto optimality. The EFESTO considers the context regarding the vehicle mobility to provide resources for the VCs and the task requirements. Also, the scheduling employed by EFESTO is concerned with minimizing processing time in VCs, reducing resource utilization time and, consequently, its monetary cost.

IEEE 96th Vehicular Technology Conference (VTC-Fall 2022)

Therefore, EFESTO goals include maximizing the number of tasks scheduled in VCs and minimizing monetary costs through decision-making based on joint minimizing processing time and task deadline. We compared the performance of EFESTO with other approaches in the literature, and the results indicate its ability to schedule a greater number of tasks, minimize the monetary cost, and minimize the tasks' processing time.

In brief, the key contributions of this paper are:

- Introduction of a task scheduling mechanism that maximizes the number of tasks scheduled without increasing the monetary cost of using computational resources.
- Efficient use of contextual aspects (mobility and task requirements) in the decision-making process.
- In a detailed performance evaluation with a realistic mobility trace, we show the necessity and outline the benefits of joint optimization for the task scheduling process compared to other state-of-the-art approaches.

This paper is organized as follows. Section II discusses the related works. Section III presents the system model, problem definition, and EFESTO's operation. Section IV discusses the performance evaluation and results obtained. Finally, Section V provides final remarks and discusses future work.

II. Related Work

Several works have been proposed for efficient task scheduling in VEC environments, utilizing vehicular and edge infrastructure resources. For example, Hattab et al. introduced a polynomial-time algorithm for task scheduling in VCs with different computational resources [7]. First, the algorithm classifies the tasks according to the completion and wait times ratio. Afterward, it selects a task subset with the lowest proportion and then solves a sequence of Linear Programs. Also, a bottleneck assignment problem is formulated, where the goal is to minimize the completion time of the scheduled tasks in the available VC. However, this work does not consider the mobility of vehicles for VC formation, *i.e.*, a VC is stationary, and the proposed algorithm considers only one VC.

Da Costa et al. presented a task scheduling mechanism for VCs based on combinatorial optimization [4]. After a VC formation process, a controller located at a higher level in the network receives the requests for resources and schedules them for processing in the available VCs using a pseudo-polynomial algorithm for the 0/1 Knapsack Problem. However, the authors do not consider contextual aspects in the scheduling decision process. That is, the impacts of vehicular mobility and the time requirements of the tasks are disregarded.

Pereira et al. introduced a mechanism using the Analytic Hierarchy Process (AHP) to select the best task set to be scheduled in the VCs, called FORESAM [8]. FORESAM aims to maximize vehicular resource use, considering all the task requirements in its decision process. FORESAM considers VCs composed of vehicles and Roadside Units (RSUs). However, FORESAM considers only one VEC controller and does not define where it is inserted in the network. Depending on the controller level, the requests will still pass through the network core, which can cause congestion and, consequently, delays in services. To circumvent problems with mobility, the authors prioritize processing time in decision-making.

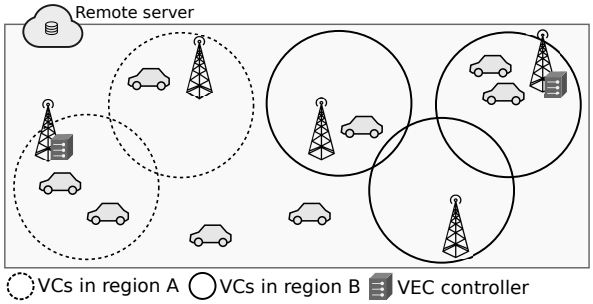


Fig. 1. System architecture.

Luo et al. presented an analysis of the delay and cost for task scheduling in VCs [3]. A scheduling framework with communication and computation for VCs is established, considering tasks with different requirements. Thus, a multi-objective problem is formulated to minimize both delay and cost. For this, a scheduling algorithm based on Particle Swarm Optimization is proposed to obtain Pareto-optimal solutions. However, due to its bio-inspired approach, its convergence time can affect the solution's overall performance. Furthermore, the authors do not consider vehicular mobility a requirement in the scheduling process.

Wu et al. investigated the task scheduling considering the vehicular mobility effects in the VEC environment [9]. Specifically, the authors formulate a joint optimization problem in a Min-Max perspective to reduce the overall latency in task scheduling. Furthermore, mobility prediction is considered to obtain better results considering a vehicle's relatively stable movement patterns in a short period. However, the mobility model is unrealistic as vehicles need constant acceleration during task scheduling and resource allocation.

Based on the state-of-the-art analysis, it is possible to observe that these works, for the most part, do not consider vehicular mobility in their scheduling decision processes. Depending on specific situations, deploying in more dynamic environments is challenging. Furthermore, even considering some task requirements for decision-making, such as task size and deadline constraints, the works do not consider them together with other contextual information, such as vehicular mobility that dictates the dynamics of resources in the network.

III. System Overview

This section describes the EFESTO mechanism, which considers VEC controllers for selecting the best VCs for task scheduling. Also, EFESTO uses Pareto optimality to select the set of tasks to be scheduled, joint minimizing processing time and monetary costs.

A. Network and System Model

Figure 1 presents the system architecture composed of vehicles, BSs, VEC controllers, and a Remote Server (RS)

in the Internet cloud. The scenario is composed of a set of x vehicles, denoted as $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Also, a set of BSs deployed in the city is considered, denoted as $b_y \in B = \{b_1, b_2, \ldots, b_p\}$, where p is the total number of BSs. All BSs have wired communication with the Internet cloud, and BSs can provide processing power and storage capacity at the network edge.

Considering the presence of the 5G network, each BS has an *Xn* interface, which allows the exchange of information between neighboring BSs. Each vehicle has an On-Board Unit (OBU) that allows vehicle-to-everything (V2X) communication. For example, vehicles can associate with a BS and communicate with a Remote Server (RS) to access the Internet and request resources. The Maximum Signal-Interference-Noise Ratio (Max-SINR) is used to associate the vehicles to a BS. In this case, a vehicle will be associated with a BS that provides the Max-SINR. In summary, for a receiving vehicle i located somewhere in the space covered by at least one BS y, its corresponding $\text{SINR}(i, y) = \frac{P_i}{I_i + N}$, where P_i is the signal strength of the BS in the vehicle i, I_i is the interference power, and N is the Gaussian white noise power spectral density [10].

In this scenario, the city is divided into R regions, and each region has at least one BS. Furthermore, we consider a set of VEC controllers, denoted as $c_o \in C = \{c_1, c_2, \ldots, c_e\}$, where $e = |R|$ is the total number of controllers, and is directly related to the number of regions R. Therefore, after the association between vehicles and BS, the BS sends this information to the RS. BS information is updated as the number of vehicles in its coverage changes. In this way, each VEC controller is responsible for managing the BSs in its city region.

In the VCs formation process, the VEC controller needs to ask the RS for information on BSs and vehicles to build its regional knowledge. In this case, the *Publish/Subscribe* paradigm is considered to obtain the relevant information without introducing unwanted traffic into the network. Considering that the number of VCs is the same as the number of BSs, the VCs can be denoted by $v_j \in V = \{v_1, v_2, \ldots, v_m\}$, where m is the total number of VCs. So $m = p$. In summary, a VC consists of a set of nodes (*i.e.*, vehicles and BS) capable of sharing processing power ω (CPU cycle frequency in Millions of Instructions Per Second (MIPS) and storage capacity ϕ in Megabytes (MB).

In summary, ω_i denotes the vehicle's CPU cycle frequency, ϕ_i denotes the vehicle's storage capacity, ω_y denotes the BS's CPU cycle frequency, and ϕ_y denotes the BS's storage capacity. Therefore, each VC is represented by a tuple $\{id_j, vehicles_j, bs_j, total_j\}$, where id_j is the VC unique identification, $vehicles_j$ are vehicular resources (ω_i and ϕ_i), bs_j are BS resources (ω_y and ϕ_y), and $total_j$ is the sum of the resources in the VC (Ω_j and Φ_j). That is, the total amount of processing power Ω_j and storage capacity Φ_j of each VC v_j is the sum of the shared resources of vehicles and BS that make up these VCs.

Concerning the VCs' resource variability over time, we use vehicular mobility information to estimate the vehicles' dwell-time in the BS coverage. An optimal mobility prediction approach is used in this work, as mobility prediction is not our focus. Information on future vehicle mobility is collected from the vehicular dataset considering a K time window. However, we added a white Gaussian noise for each collected data [11]. In this sense, as we now estimate the resources available in the VCs in $k \in K$ time units, the resources available in the VCs can be represented by Ω_{jk} and Φ_{jk}.

B. 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 represents the unique ID of the task, s_l^t (in MB) denotes the size of the task's input data, w_l^t in Millions of Instructions (MI) is the number of CPU cycles required to process the task, and D_l^t is its deadline constraint. According to the literature [6], the processing time d_{lj}^t (*i.e.*, execution time of a task in a given computational configuration) can be obtained based on the required CPU cycle w_l^t divided by the CPU cycle frequency of the server Ω_j, as

$$d_{lj}^t = \frac{w_l^t}{\Omega_j}, \forall t_l \in v_j. \tag{1}$$

As VC's computational resources are shared among the various tasks, the Ω_j considered for processing time calculation for a given task must be updated according to the degree of sharing of this resource within the VC, represented by Ψ_j. So, Ω_j is divided by the number of tasks $|T_j'|$ that were scheduled in this VC to yield

$$\Psi_j = \frac{\Omega_j}{|T_j'|}, j \in V. \tag{2}$$

Each task has a D_l^t deadline constraint in its configuration. This constraint represents a time limit that the task can wait to be fulfilled. If $d_{lj}^t \leq D_l^t$, then it means that the task can be scheduled and executed in the VC v_j. 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

$$\mathcal{C}_l = d_{lj}^t \times (w_l^t \times \mathcal{P}(t_l)). \tag{3}$$

Where d_{lj}^t is the t_l processing time in $v_j \in V$ and w_l^t is its CPU cycles required. $\mathcal{P}(t_l)$ indicates the resource price used and is set to 11.444 (if t_l uses BS's resources) or 5.016 (if t_l uses vehicles' resources). The prices are based on instances with GPU capacity available on Amazon EC2[1], such as *g4ad* and *g3* for BS and vehicle, respectively.

In summary, when a task arrives in the system, the VEC controller must select the best VC to process that task. This selection should consider the VC's processing power over time and the task requirements. So, to consider these different objectives, a task scheduling problem was formulated that primarily seeks to maximize the number of tasks scheduled

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

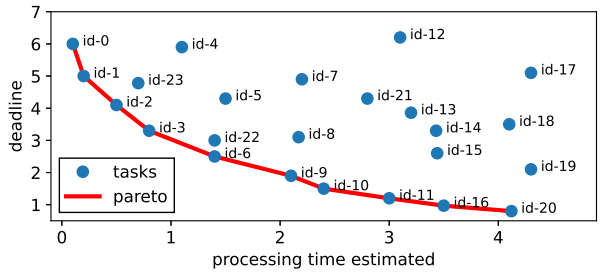


Fig. 2. Pareto set example.

considering constraints that directly impact the monetary costs, as follows:

$$\text{maximize} \sum_{l=1}^{n} t_l,\ l \in T, \tag{4}$$

$$\text{subject to } d_{lj}^t \leq D_l^t,\ l \in T',\ j \in V, \tag{5}$$

$$\sum_{l=1}^{n} s_l^t \leq \Phi_{jk},\ l \in T',\ j \in V,\ k \in K, \tag{6}$$

$$\sum_{l=1}^{n} w_l^t \leq \Omega_{jk},\ l \in T',\ j \in V,\ k \in K. \tag{7}$$

The constraint (5) guarantees that the task deadline is respected and helps reduce the monetary cost, avoiding rescheduling. Also, constraints (6) and (7) ensure that VCs' storage and processing limits during the k required processing time intervals are respected.

C. EFESTO's operation

The problem of scheduling tasks in VCs needs to consider more than one aspect during the decision process. Furthermore, the literature argues that the task scheduling problem in VCs is $\mathcal{NP}$-hard [12]. Thus, EFESTO considers a Pareto optimality approach to select the best task set to schedule in a determined VC, minimizing de processing time and monetary costs associated with this scheduling.

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, EFESTO 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, EFESTO needs to know the processing time of all queued tasks. In this step, Equation (1) estimates the processing time since it is unknown how many tasks will be scheduled in this VC.

In short, EFESTO searches for the Pareto set using as a criterion the joint minimization of task processing times and task deadlines and 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 2 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 $\mathcal{O}(n \log n)$ [13].

Algorithm 1 describes the EFESTO'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 P for the estimated processing time, and the second D for deadlines (Lines 3 and 4). EFESTO calls the procedure PARETOSET with configuration for joint minimization of vectors P 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. Also, the total number of resources needed for this returned set is calculated (Line 6). After that, for each task in the set, it is verified if the VC will have available resources until its deadline D_l^t (Line 8). If not, that task is removed from the set $\mathcal{P}$ (Line 9). If so, its actual processing time is calculated (Line 12). If the processing time is longer than its deadline, the task is removed from $\mathcal{P}$ and will be rescheduled in the next round. Otherwise, set $\mathcal{P}$ is added to the scheduled tasks list T'.

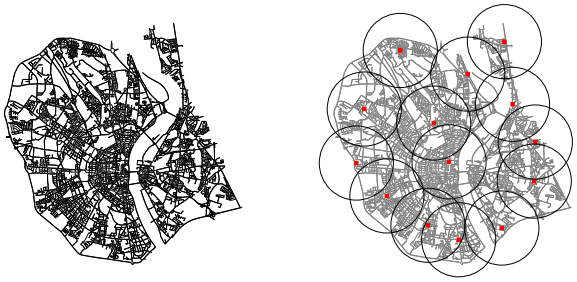
Algorithm 1: EFESTO

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

1 $V \leftarrow$ decreasing order of available resources
2 **foreach** $v_j \in V$ **do**
3 $P \leftarrow$ processing time (t_l, v_j) for each $t_l \in T$
4 $D \leftarrow$ deadline of each $t_l \in T$
5 $\mathcal{P} \leftarrow$ PARETOSET($\{P, D\}$, objective=$[min, min]$)
6 totalResources $\leftarrow \sum_{l=1}^{n} w_l^t, \forall t_l \in \mathcal{P}$
7 **foreach** $t' \in \mathcal{P}$ **do**
▷ Use predicted vehicular information here
8 **if** $(t' + totalResources) < v_j$ *until* D_l^t **then**
9 $\mathcal{P} \leftarrow \mathcal{P} \setminus \{t'\}$
10 totalResources $\leftarrow$ totalResources $- t'$
11 **else**
12 $d_{lj}^t \leftarrow$ Equation (1)
13 **if** $d_{lj}^t > D_l^t$ **then**
14 $\mathcal{P} \leftarrow \mathcal{P} \setminus \{t'\}$
15 **else**
16 $T' \leftarrow \mathcal{P}$
17 **return** T'

IV. EVALUATION

This section describes the methodology and metrics used to evaluate the EFESTO's efficiency. Simulations were performed on a realistic mobility trace of Cologne, Germany, to analyze



(a) Cologne sub-map. (b) BSs deployed in the city.

Fig. 3. Simulation scenario.

the efficiency of the EFESTO compared to other approaches to task scheduling.

A. Scenario description and Methodology

The experiments were carried out with the Simulator of Urban Mobility (SUMO), in version 1.10.0. The algorithms were implemented in Python 3.8.10 and connected to SUMO through the TraCI interface. The mobility trace of the TAPAS Cologne[2] project was used, which reproduces vehicle traffic in the city of Cologne, Germany. However, only a central sub-map of 114 km^2 was considered (Figure 3(a)), with 2 hours of vehicular mobility and up to 700 vehicles at peak times. The simulation time was 600 seconds, with 100 initial seconds of warm-up. The simulations were run 33 times to obtain a 95% confidence interval.

The Bag-of-Tasks (BoT) applications were considered since they can be executed outside the submission order. The tasks' deadline varies between $3, 5$, and 7 seconds. This is important to generalize the representation of possible application classes. VC formation intervals were 5 seconds. The arrival rate of tasks λ in the system is 1 tasks/second, following a Poisson distribution. The communication ranges of vehicles and BSs were 250 and 2000 meters, respectively.

Furthermore, the size assigned to the tasks was $s_l^t = [1, 10]$ (MB) and the CPU cycles required varying in $w_l^t = [1, 10]$ (MI). The number of processing units per vehicle is 1 CPU, which without loss of generality represents 1 MIPS. Each vehicle's storage capacity has been simplified to 1 MB. 14 BSs were used, and each one is capable of sharing processing and storage resources, where such values were configured at 15 MIPS and 15 MB, respectively. 4 VEC controllers were considered, and each can manage up to 4 neighboring BSs. The BSs were arranged in the city following positioning information provided by the TAPAS Cologne project (Figure 3(b)).

The metrics used for evaluation were: i) *Scheduled Tasks* represents the percentage of tasks successfully completed; ii) *Monetary Cost* refers to the time of use of the resources; iii) *System Latency* refers to the processing time of the task in a given computational configuration plus the queue waiting time; and iv) *CPU Time* represents the time required for the controller to make the scheduling decision.

We compared the performance of EFESTO with three approaches for task scheduling in VCs, namely: *FCFS* [7], which implements a policy based on First Come, First Served (FCFS) for scheduling tasks in the VEC environment; *CRATOS* [4], which uses a pseudo-polynomial combinatorial optimization approach in the task scheduling process; and *FORESAM* [8], which uses a multi-criteria approach in its decision process.

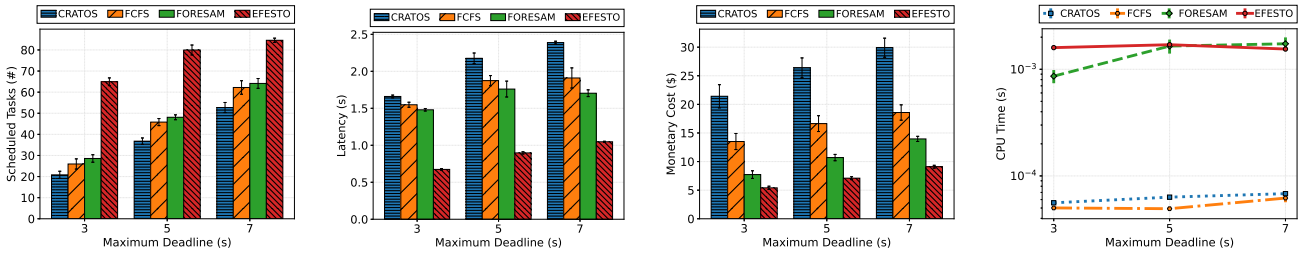
B. Simulation Results

Figure 4(a) shows the percentage of successfully scheduled and executed tasks. All approaches improve their performance as the maximum deadline increases. This means that the mechanisms have more time for decision-making and can make more mistakes until the deadline is reached. EFESTO can schedule more than 85% of tasks in configurations with less complex time constraints (maximum deadline 7). FORESAM and FCFS operate statistically similarly in this assessment. CRATOS has the worst performance in all scenarios. This is due to its decision strategy, which is only concerned with the task's size and an associated reward value employed by its Knapsack Problem strategy, disregarding fundamental aspects such as deadline and processing time. Tasks scheduled with CRATOS have restrictions that are not considered in its decision-making process. In the most challenging configuration, with a maximum deadline equal to 3, EFESTO is still superior, but managing to schedule only 65% of tasks.

Figure 4(b) shows the results regarding the system latency, which includes queue waiting time and task processing time. The lower the latency, it means the scheduling rounds are efficient. In all the evaluations, it can be noticed that the EFESTO reduces the system latency. This is mainly due to the VCs selection employed by EFESTO, which minimizes the task processing time and the deadline constraint. Based on this, VCs will process tasks in less time. In this way, EFESTO can handle a more significant number of tasks in a shorter time than other approaches. Besides, mobility information helps estimate future resources, guaranteeing EFESTO lower error rates in its scheduling process. CRATOS has higher latency due to its decision-making prioritizes the task size. On the other hand, FORESAM and FCFS operate similarly. However, FCFS has a slight advantage in the least challenging scenario (maximum deadline 7). In summary, EFESTO better managed computational resources by jointly considering contextual aspects of tasks and VCs in its decision-making process.

Figure 4(c) presents the monetary cost of using the VCs' computational resources. In general, to minimize monetary costs, approaches choose first to select vehicular resources for scheduling. It can be noted that the EFESTO minimizes the monetary cost in all evaluations performed. In this case, it is natural that the performance of all mechanisms drops due to the percentage of scheduled tasks that also fit into more challenging scenarios (Figure 4(a)). The best performance of EFESTO is due to the selection of tasks considering the future resources available in the VC. The application of Pareto

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



(a) Scheduled Tasks (b) System Latency (c) Monetary Cost (d) CPU time measuring in the VEC controllers

Fig. 4. Simulation results for different deadlines.

Optimality allows the best task set to be scheduled in the same VC with minimum processing time and deadline constraints.

Finally, Figure 4(d) presents the CPU usage 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, FCFS 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, EFESTO 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.

V. Conclusion

In this paper, we proposed a task scheduling mechanism for VEC environments, using the Pareto optimality principle to select the best task set to be scheduled in VCs. This selection is based on contextual aspects, such as the processing time and deadline of the tasks, as well as vehicle mobility information to estimate the resources in each VC. Simulation results show that EFESTO fulfills its objectives to maximize the number of scheduled tasks while minimizing the total processing time and monetary cost of using VEC resources. Future works include machine learning techniques for mobility prediction and resource estimation in each cloud. Also, we intend measurement of network metrics about the maintenance of knowledge by the VEC controllers.

Acknowledgments

This work is supported by the grants #2015/24494-8, #2018/16703-4, and #2021/13780-0 of the São Paulo Research Foundation (FAPESP), and the National Council for Scientific and Technological Development (CNPq).

References

[1] A. M. de Souza, H. F. Oliveira, Z. Zhao, T. Braun, A. A. Loureiro, and L. A. Villas, "Enhancing sensing and decision-making of automated driving systems with multi-access edge computing and machine learning," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, pp. 44–56, 2022.

[2] 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, 2021.

[3] 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.

[4] 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 (VTC Spring)*. IEEE, 2020, pp. 1–5.

[5] 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, 2020.

[6] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2020.

[7] 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.

[8] R. Pereira, A. Boukerche, M. A. da Silva, L. H. 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, 2021.

[9] X. Wu, S. Zhao, R. Zhang, and L. Yang, "Mobility prediction-based joint task assignment and resource allocation in vehicular fog computing," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[10] C. Li, B. Zhang, and X. Tian, "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, 2021.

[11] Y. Zhang, M. Li, Y. Zhang, Z. Hu, Q. Sun, and B. Lu, "An enhanced adaptive unscented kalman filter for vehicle state estimation," *IEEE Transactions on Instrumentation and Measurement*, 2022.

[12] 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, 2019.

[13] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *17th international conference on data engineering*. IEEE, 2001, pp. 421–430.