Improving Fairness and Performance in Resource Usage for Vehicular Edge Computing

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

***Abstract*—Vehicular Edge Computing (VEC) has emerged to offer cloud computing services closer to vehicular users by combining vehicles and edge computing nodes into Vehicular Clouds (VCs). In this scenario, an intelligent task scheduler must decide which VC will run which tasks, considering contextual aspects like vehicular mobility and tasks' requirements. This is important to minimize both processing time and monetary costs. However, such direct optimization can lead to unfairness in resource usage, easily leading to (as we will show) decreased performance. Towards this end, in this work, we propose FARID, a task scheduling mechanism that considers contextual aspects of its decision process and applies a probabilistic selection function on VCs to balance the processing load and increase the fairness in the use of vehicular resources. Compared to state-of-the-art solutions, FARID has a higher level of fairness and can schedule more tasks while minimizing monetary costs and system latency.**

I. Introduction

Vehicular Edge Computing (VEC) emerged as a paradigm to provide computing power and storage close to vehicular users [1]. VEC allows applications with strict latency requirements to be provided cloud computing services at lower levels in the network hierarchy, thus mitigating problems such as transmission delays, poor connection reliability, and high bandwidth usage. For this purpose, VEC needs to aggregate the resources and make them available on the network [2].

In this context, the computational resource aggregation from vehicles and Base Stations (BSs) with computational capabilities is known as *Vehicular Cloud (VC) formation* [3]. This process typically considers the vehicular mobility pattern to group vehicles in coverage of a BS and, with this, can maintain stable resource availability [4]. The utilization of these aggregated computational resources, in turn, is referred to as *task scheduling* [5]. This process is performed in real-time and is designed to optimize the use of available resources while ensuring that tasks are completed promptly and efficiently [6].

One of the main challenges of task scheduling in VCs is the dynamic nature of the vehicular network, as vehicles move and change their position, making it challenging to predict resource availability in a specific location. To tackle this challenge, advanced strategies are used to determine the best task placement based on factors such as communication costs and mobility prediction [2].

However, much less attention has been paid to questions of fairness in resource usage in the scheduling process [7]. We treat this as an issue of load balancing among the nodes processing tasks (vehicles and BSs), that is, distributing the workload evenly across available computational resources [8]. Load balancing is performed either when a task arrives or once it has already been queued [9]. It ensures that tasks are allocated efficiently to prevent resource over or underutilization [10]. By balancing the load, the system can achieve better resource utilization and response times. Additionally, load balancing can help mitigate congestion and handle varying demand patterns, resulting in a more efficient and effective task scheduling process [4]. Therefore, another important challenge of VCs is to meet user demands by maintaining fair load balancing among available computational resource usage [7], [8] while still considering node mobility in the scheduling process.

Towards this end, this work introduces FARID (FAir Resource usage in vehIcular clouDs). FARID runs on VEC controllers and uses Pareto optimality to schedule tasks in different VCs. The mechanism splits the set of tasks into different parts to improve the system efficiency with parallel management, obtaining k different Pareto sets and being able to make k decisions at the same time, where k is the number of threads running in each VEC controller. FARID aims to minimize processing time within VCs, thus reducing resource utilization and, subsequently, monetary costs. Also, it considers contextual aspects in its decision process, such as resource mobility in each VC and task's requirements. We assessed the efficiency of FARID compared to other mechanisms, and the results indicate its capability to schedule a larger quantity of tasks, minimize monetary costs, and reduce overall system latency. Lastly, FARID employs better load balancing in the scheduling process, resulting in greater fairness in the resource usage of VCs.

In summary, the contributions of this work are:

- a task scheduling mechanism that maximizes the tasks scheduled while maintaining a fair load balancing in the use of computational resources;
- the use of multithreading for parallel resolution of scheduling subproblems, aiming to reduce system latency; and
- the utilization of contextual information to aid in the decision-making process.

98th IEEE Vehicular Technology Conference (VTC2023-Fall)

II. Related Work

Numerous studies have created diverse strategies to optimize task scheduling in VEC systems. Also, some of these strategies try to balance the scheduling load in the VCs. For instance, Hattab *et al.* [11] introduced an optimized algorithm for scheduling tasks in VCs with varied resources, which mainly uses the First-Come, First-Served (FCFS) standard from the literature. The algorithm classifies tasks by completion-to-waiting time ratio, identifies those with the lowest ratio, and uses Linear Programs to solve them. It aims to minimize task completion time within VC's resources, assuming a static VC and focusing solely on a single VC. In this case, load balancing is trivial, as there is only one VC for decision-making.

Some works consider Multi-Criteria Decision Making (MCDM) methods for selecting the VC to perform the scheduling process. Such MCDM methods tend to balance the selected options to insert a load balance during the decision process. Aligning with this approach, Mishra *et al.* [12] introduced two Analytic Hierarchy Process (AHP)-based resource allocation policies, named SECA and AHP-EV. Resource allocation can also be defined as task scheduling, depending on the scenario. The proposed schemes consider the network load and compute load during decision-making, aiming to minimize each task's delay. These policies differ in assigning weights to each criterion, but both have similar results. AHP-EV employs pre-set weights for computing power and network, whereas the SECA dynamically determines criteria weights.

Other works utilize multi-objective optimization methods for the task selection process that will be scheduled. Da Costa *et al.* [6] presented a task scheduling mechanism for VCs based on Pareto optimality, named EFESTO. EFESTO employs vehicle mobility information to estimate the resources available in each VC, thereby enabling more precise decision-making. Moreover, it uses Pareto optimality to select the optimal subset of tasks for joint minimization of deadline and processing time, which minimizes the monetary cost associated with resource usage. However, EFESTO prioritizes the VCs with more available resources without considering possible overloads that this continuous selection can cause.

Similarly, Luo *et al.* [13] evaluated delay and cost implications associated with task scheduling in VCs. The study established a scheduling framework that accommodates communication and computation within VCs, considering tasks with varied requirements. Consequently, a multi-objective problem was designed to minimize both delay and cost. A Particle Swarm Optimization-based scheduling algorithm was suggested to derive Pareto-optimal solutions. However, due to the bio-inspired nature of the approach, the convergence time could impact the solution's overall performance. Additionally, the authors did not factor in vehicular mobility as a prerequisite in the scheduling process.

Ribeiro *et al.* [14] introduced a metaheuristic approach that models VCs as a coalition. In game theory, a coalition refers to a group of players who agree to cooperate by combining their strategies to enhance their joint payoff. In this context, the authors propose a coalition game to maximize resource utilization while dynamically balancing the load among the VCs. Firstly, the mechanism establishes a strategy based on the Shapley value to determine the sequence in which tasks are scheduled. After that, the mechanism uses a queue to schedule tasks within VCs based on values calculated. Nonetheless, the authors overlook the contextual factors in the scheduling decision process. They do not consider the implications of vehicular mobility and task deadline constraints.

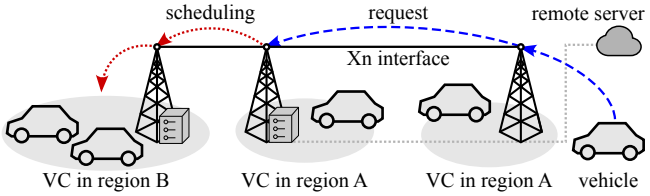


Fig. 1. A system architecture employed by FARID, presenting its main components, such as vehicles, Base Stations (BSs), Remote Server (RS), Vehicular Clouds (VCs), and Vehicular Edge Computing (VEC) Controllers.

Based on the state-of-the-art analysis, it is possible to observe that these works mostly do not consider mobility information in scheduling decision processes. Also, it is essential to consider load balancing during the task scheduling decision process to increase fairness in using resources and not overload specific regions in the city, increasing its maintenance costs.

III. System Overview

This section describes FARID, which considers Pareto optimality and a probabilistic selection function to maximize scheduled tasks, load balancing, and fairness in resource usage.

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. The scenario has a set of x vehicles, denoted as $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Also, there is a set of p BSs deployed in the city, denoted as $b_y \in B = \{b_1, b_2, \ldots, b_p\}$. Each BS can communicate with the RS via optical fiber link.

To improve the management of BSs, the city is divided into R regions, and each region has at least one BS. Furthermore, we consider a set of $|R|$ VEC controllers, since each region is managed by exactly one controller. Therefore, after the association between the vehicle and BS, the BS sends this information to the RS. The association process considers the Max-SINR approach. BS information only is updated as the number of vehicles in its coverage changes.

In the resource aggregation process (VC formation), the VEC controller needs to request the RS about BSs and vehicles information to build its regional knowledge [3]. In this way, the system employs a *Publish/Subscribe* scheme to obtain the relevant information without introducing unwanted traffic into the network. The set of VCs can be denoted by $v_j \in V = \{v_1, v_2, \ldots, v_m\}$, where m is the total number of VCs. We consider that the number of VCs is the same number of BSs, as the positioning of the BS defines where the VC will act. In

summary, a VC consists of a set of vehicles and BS capable of sharing processing power ω in Million of Instructions Per Second (MIPS) and storage capacity ϕ in Megabytes (MB). 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.

Due to VC's resource variability over time, we utilize accurate vehicular mobility data to determine each vehicle's stay within BSs' coverage. Even though mobility prediction is not the primary focus of this study, we employ an optimal mobility prediction method. Future vehicle mobility data is gathered from the vehicular dataset within a time window Z. To simulate prediction errors, we introduce a white Gaussian noise to each collected data [6]. Finally, as we now estimate the available resources in the VCs at $z \in Z$ time units, these resources can be denoted by Ω_{jz} and Φ_{jz}.

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 identification number, s_l^t denotes the input data size (in MB), w_l^t is the number of Central Processing Unit (CPU) cycles required to process the task, and D_l^t is its deadline constraint. The processing time d_{lj}^t (*i.e.*, execution time of a task in a specific computational configuration) can be obtained based on the required CPU cycle w_l^t divided by the server's CPU cycle frequency Ω_j, as $d_{lj}^t = \frac{w_l^t}{\Omega_j}, \forall t_l \in v_j$.

As VC's computational resources are shared among different tasks, the Ω_j considered for getting processing time for a given task must be updated according to the degree of sharing of this resource within the VC, represented by Ψ_j. Thus, Ω_j is divided by the number of tasks $|T_j'|$ that were scheduled in this VC to yield $\Psi_j = \frac{\Omega_j}{|T_j'|}, j \in V$.

Additionally, each task has a deadline constraint D_l^t. This deadline represents a time limit that the task can wait to be processed. If $d_{lj}^t \leq D_l^t$, 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

$$\mathcal{C}_l = d_{lj}^t * (w_l^t * ResourcePrice(t_l)). \tag{1}$$

where d_{lj}^t is the t_l processing time in $v_j \in V$ and w_l^t is its CPU cycles required. $ResourcePrice(t_l)$ indicates the resource price used and is set to \$14.309 (if t_l uses BS's resources) or \$6.27 (if t_l uses vehicles' resources). The prices are based on instances available on Amazon EC2 [1] (Region Europe, Frankfurt), such as *g4ad* (BS) and *g3* (vehicle).

In summary, when a task arrives in the system, it is queued and waits until it is scheduled. The VEC controller must select the VC to process this task. This selection decision 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 considering constraints that directly impact the monetary costs, as follows:

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

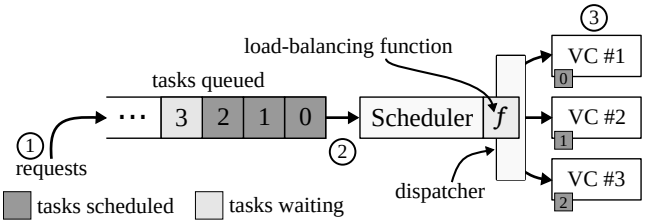


Fig. 2. The task scheduling pipeline with a load-balancing function integrated into the scheduler approach.

$$\text{maximize} \sum_{l=1}^{n} t_l,\ l \in T, \tag{2}$$

$$\text{subject to } d_{lj}^t \leq D_l^t,\ l \in S',\ j \in V, \tag{3}$$

$$\sum_{l=1}^{n} s_l^t \leq \Phi_{jz},\ l \in S',\ j \in V,\ z \in Z, \tag{4}$$

$$\sum_{l=1}^{n} w_l^t \leq \Psi_{jz},\ l \in S',\ j \in V,\ z \in Z. \tag{5}$$

The constraint (3) guarantees that the task deadline is respected and helps reduce the monetary cost, avoiding rescheduling. Also, constraints (4) and (5) ensure that VCs' storage and processing limits during the z required processing time intervals are respected.

C. FARID's operation

Figure 2 presents a simplified task scheduling process pipeline in VCs. In summary, tasks are queued on the VEC controller as soon as they arrive in the system (Label ①). In this phase, the task can assume two states, waiting and scheduled. A task can assume a scheduled state and return to a waiting state if the VC fails to complete its processing and the task's deadline is $D_l^t > 0$. From that moment on, the task deadline must be observed constantly since the task is waiting for the scheduler's decision. Based on its criteria, the scheduler decides which VC the task will be processed. This scheduler can employ a load-balancing strategy to increase fairness levels in utilizing the VEC's computational resources. A Dispatcher has the role of distributing the tasks to the corresponding VCs (Label ②). The task distribution ratio is stored by VEC controller for load-balancing purposes in future decisions. Finally, the VCs receives the tasks and starts processing (Label ③).

In this context, FARID seeks the Pareto set using a dual-criteria approach: it simultaneously minimizes tasks' processing times and deadlines. A distinct vector is created for each criterion (processing time and deadline), arranged in a 2-dimensional plane to find the Pareto set. We can obtain a Pareto set in 2-dimensional in polynomial time $\mathcal{O}(n \log n)$ [6]. Besides, as the objective is to maximize the number of tasks scheduled, we simplify our problem to an instance of the Bin Covering Problem (BCP), which solves this issue [3].

Furthermore, FARID divides the task queue and set of VCs into k parts to make the scheduling process more efficient. Now, each k part is responsible for calculating a Pareto set and the universe of available VCs is smaller for the BCP application. FARID makes k decisions simultaneously. This division is performed in the Dispatcher. The `Dispatcher` Function in Algorithm 1 exemplifies this splitting process.

With the scheduling decision taken, carrying out a load balance among the available VCs to increase fairness in resource usage is essential. With this in mind, we consider a selection probabilistic function when the scheduler needs to select the VC in the scheduling process. After a VC is selected, its selection probability decreases. Thus, when FARID selects the VCs in the next round, an ordering of the probability vector is performed, and the VC with the highest current probability is selected. With that, after m rounds, all VCs are selected at least once during the execution of FARID, where m is the total number of VCs.

In summary, every VC has an original probability of being selected during the scheduling process, according to

$$P_j^0, \quad j = 1, 2, \ldots, m \tag{6}$$

The selection probability is halved (0.5) if the VC is selected in the current round. Otherwise, the previous probability is maintained, according to Equation (7).

$$P_j^z = \begin{cases} 0.5 \cdot P_j^{z-1}, & \text{if item } j \text{ was selected} \\ P_j^{z-1}, & otherwise \end{cases} \tag{7}$$

To avoid the scenario that after a high number of rounds, the probability of a VC is reduced to zero and it is never selected again, a minimum nonzero probability is considered as

$$P_j^z = \begin{cases} 0.0001, & \text{if } P_j^z \leq 0 \\ P_j^z, & otherwise \end{cases} \tag{8}$$

Algorithm 1 describes the operations of FARID in a VEC controller. The controller gets the VC set V, the task set T, and the number of threads k, which gives the S scheduled task set as output. The `Dispatcher` Function splits the T and V into k parts and starts the execution of the `Scheduling` Function in parallel for each defined k (Lines 2 and 4). In the `Scheduling` Function, the VCs' probability vector is sorted in descending order to select VC with the highest selection probability (Line 6). For each v checked and selected, the probability must be updated (Line 8). Two vectors are created based on T, the first R for the estimated processing time, and the second D for deadlines (Lines 9 and 10). FARID calls the procedure PARETOSET with configuration for joint minimization of vectors R and D (Line 11). The procedure returns a set $\mathcal{P}$ containing the id of the tasks in the Pareto set. After that, FARID runs BCP to select the best subset $S' \in \mathcal{P}$ that best fits in V (Line 12). Also, the total number of resources needed for this returned set is calculated (Line 13). For each task in the set S', it is verified if the VC will have available resources until its deadline D_l^t (Line 15). If not, that task is removed from the set S' (Line 16). If so, its actual processing time is calculated (Line 19). 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 S (Line 23).

Algorithm 1: FARID

```
   Input: task set T, VC set V, and number of threads k
   Output: scheduled tasks set S
1  Function Dispatcher(T, V, k):
2      Split task set T into k parts
3      Split VC set V into k parts
4      Running Scheduling (T_k, V_k) for each thread k
5  Function Scheduling(T, V):
6      V ← descending order of selection probability
7      foreach v ∈ V do
8          Update v's probability using Equation (7)
9          R ← tasks' processing time for v
10         D ← tasks' deadline
11         P ← ParetoSet({R, D}, obj=[min, min])
12         S' ← BinCoveringProblem(P, v)
13         totalResources ← Sum all resources in S'
14         foreach t' ∈ S' do
15             if totalResources < v_j until D_l^t then
16                 S'.remove(t')
17                 totalResources ← totalResources − t'
18             else
19                 d_lj^t ← As shown in Section III-B
20                 if d_lj^t > D_l^t then
21                     S'.remove(t')
22                 else
23                     S ← S'
24     return S
```

IV. EVALUATION

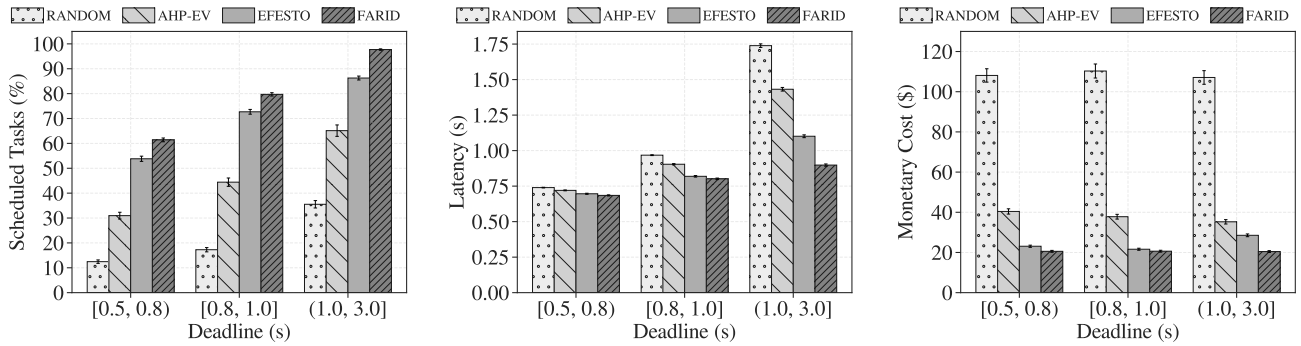
This section describes the methodology and metrics used to evaluate the efficiency of FARID.

A. Scenario description and Methodology

The experiments were carried out with the Simulation of Urban MObility (SUMO) 1.16.0. The algorithms were implemented in Python 3.8.10 and connected to SUMO through the TraCI interface. We used a central sub-map of 114 km^2 from TAPASCologne trace[2], which reproduces vehicle traffic in the city of Cologne, Germany. We consider 2 hours of vehicular mobility and up to 700 vehicles. The simulation time was 800 seconds, with 100 initial seconds of warm-up. We ran the simulations 33 times to obtain a 95% confidence interval.

The Bag-of-Tasks (BoT) applications were considered since they can be executed outside the arrival order. The tasks' deadline varies between $[0.5, 0.8)$, $[0.8, 1.0]$, and $(1.0, 3.0]$ seconds.

[2]https://sumo.dlr.de/docs/Data/Scenarios.html



(a) Scheduled Tasks (b) System Latency (c) Monetary Cost

Fig. 3. Simulation results considering different deadline constraints.

This is important for generalizing different application classes. VC formation intervals were 5 seconds. The arrival rate of tasks λ is 5 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 $w_l^t = [1, 30]$ Million of Instructions (MI). The number of CPU per vehicle is 1, 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 can share processing power equal 15 MIPS and storage capacity equal 15 MB. 4 VEC controllers were considered, and each can manage up to 4 neighboring BSs. The BSs deployment positions followed the information provided by the TAPASCologne project.

We compared the performance of FARID with three approaches, namely: *RANDOM*, which combines a policy based on First-Come, First-Served (FCFS) [11] with a randomized policy to select VCs [15]; *EFESTO* [6], which uses Pareto optimality in the task scheduling process and selects VCs with more resources every round; and *AHP-EV* [12], which uses the AHP multi-criteria approach in its decision-making process.

The metrics used for evaluation were: i) *Scheduled Tasks* represent the percentage of tasks successfully completed; ii) *Monetary Cost* refers to the resources usage price; iii) *System Latency* refers to the processing time plus the queue waiting time; and iv) *Fairness* represents Jain's fairness index. This metric is widely used to measure how fair the use of resources is in a computational system.

B. Simulation Results

Figure 3(a) shows the percentage of successfully scheduled and executed tasks. The performance of all approaches enhances as the maximum deadline extends. This means that the mechanisms have more time for decision-making and can make more unsuccessful attempts until the deadline is reached. FARID can schedule more tasks in all observed scenarios. Also, our mechanism can schedule more than 98% of tasks in configurations with less complex time constraints (maximum deadline $(1.0, 3.0]$). The use of mobility information helps in more accurate decisions, ensuring that the task will complete its processing in the selected VC. EFESTO is the second mechanism that can schedule more tasks. However, the fact that it only considers one Pareto set can make the decision process difficult, as possible better subsets are disregarded. The AHP-EV reaches lower levels of scheduling, compared to the two previous mechanisms, due to its decision strategy that selects only one pair (task, VC) at each round. This approach aims to integrate the criteria, which are the task requirements, and select the best task for the current VC. RANDOM has the worst performance in all scenarios. This is due to its decision strategy, which is only concerned with the task's size employed by FCFS, disregarding fundamental aspects such as deadline and processing time. Besides, the selection of VCs is carried out at random to increase fairness in the use of resources. In the most challenging configuration, with a maximum deadline equal to $[0.5, 0.8)$, FARID is still superior, but managing to schedule only 61% of tasks.

Figure 3(b) shows the results regarding the system latency, which includes queue waiting time and task processing time. In this metric, the lower latency means the scheduling rounds are more efficient. In all the evaluation scenarios, it can be observed that FARID reduces the system latency. The best result is mainly achieved due to the selection of VCs employed by FARID, which minimizes the task processing time and the deadline constraint jointly. Also, the use of k decision-makings at the same time allows a smaller universe of tasks to be explored and helps to reduce system latency. In this way, FARID can handle a more significant number of tasks in a shorter time than other approaches. Mobility information helps estimate future resources, guaranteeing lower error rates in its scheduling process. RANDOM has higher latency in all evaluations due to its decision-making prioritizes the task size. EFESTO has an advantage over AHP-EV in the least challenging scenario (maximum deadline $(1.0, 3.0]$). In summary, FARID improved resources management by incorporating contextual aspects of tasks and VCs into its decision-making process.

Figure 3(c) shows the monetary cost of using the VCs' computational resources. As the resource prices differ, as discussed in Section III-B, the approaches prioritize vehicle

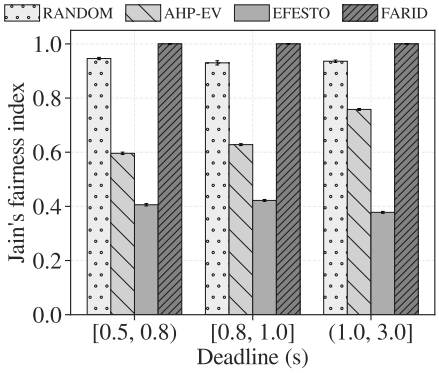


Fig. 4. Jain's fairness index considering different deadline constraints.

resources to minimize the final monetary cost. It can be noted that FARID minimizes the monetary cost in all evaluations performed. The best performance of FARID is due to the selection of tasks considering the future resources available in the VC. Pareto optimality allows the best task set to be scheduled in the same VC with minimum processing time and deadline constraints. Furthermore, separating the queued tasks into k parts makes the checks more efficient, and FARID makes fewer errors during the scheduling decision. RANDOM performs worse on this metric because its selection of VCs is entirely random, which does not guarantee that the task will be completed on the VC selected.

Finally, Figure 4 presents the fairness index obtained by the approaches when selecting VCs for scheduling. FARID obtains the best fairness index in all considered evaluation scenarios. The probability-based VC selection function allows all VCs to be selected during the scheduling process, thus increasing the network's ability to meet user demands (as seen in Figure 3(a)). The RANDOM approach obtained the second-best performance in this metric because it randomly selects a VC each round. This way, the chance of all VCs being selected increases significantly. However, it is crucial to view these metrics holistically. It is not sufficient for a scheduling approach to have a high fairness index if it can not satisfy even 15% of user demands in more complex scenarios. EFESTO has the worst fairness index since it prioritizes VCs with more resources in each scheduling round.

V. Conclusion

In this paper, we proposed a task scheduling mechanism for Vehicular Edge Computing (VEC) environments, using the Pareto optimality principle to select the best task set to be scheduled in Vehicular Clouds (VCs). We combine Pareto optimality with Bin Covering Problem (BCP) to find the most suitable fit between tasks and VC resources. The task selection is based on contextual aspects, such as the processing time and task deadline. Besides, we consider vehicular mobility information to estimate the resources in each VC. The proposed approach also guarantees high levels of fairness in using vehicular resources, applying a probability function for load balancing on the selected VCs. Compared to state-of-the-art solutions, FARID has a higher level of fairness and can schedule more tasks while minimizing monetary costs and system latency.

Future works include exploring other approaches to load balancing and fairness. Also, we intend to evaluate network metrics to get the maintenance/knowledge traffic in the VEC controllers.

Acknowledgments

This work is supported by the São Paulo Research Foundation (FAPESP), grants #2018/16703-4 and #2019/19105-3.

References

[1] J. Xue, Q. Wang, H. Zhang, N. An, and C. An, "Idle-parked vehicles assisted collaborative resource allocation in VEC based on Stackelberg game," *Ad Hoc Networks*, vol. 142, p. 103 069, Apr. 2023.

[2] Y. Ju *et al.*, "Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5555–5569, May 2023.

[3] J. B. D. da Costa, A. M. de Souza, R. I. Meneguette, E. Cerqueira, D. Rosário, C. Sommer, and L. Villas, "Mobility and Deadline-Aware Task Scheduling Mechanism for Vehicular Edge Computing," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[4] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

[5] N. Keshari, D. Singh, and A. K. Maurya, "A survey on Vehicular Fog Computing: Current state-of-the-art and future directions," *Vehicular Communications*, vol. 38, p. 100 512, Dec. 2022.

[6] J. B. D. Da Costa, A. M. de Souza, D. Rosário, C. Sommer, and L. A. Villas, "Efficient Pareto Optimality-based Task Scheduling for Vehicular Edge Computing," in *IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, IEEE, Sep. 2022.

[7] C. Chen, H. Li, H. Li, R. Fu, Y. Liu, and S. Wan, "Efficiency and Fairness Oriented Dynamic Task Offloading in Internet of Vehicles," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1481–1493, Sep. 2022.

[8] K. Hejja, S. Berri, and H. Labiod, "Network slicing with load-balancing for task offloading in vehicular edge computing," *Vehicular Communications*, vol. 34, p. 100 419, Apr. 2022.

[9] S. McClure, A. Ousterhout, S. Shenker, and S. Ratnasamy, "Efficient scheduling policies for Microsecond-Scale tasks," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1–18.

[10] M. H. Kashani and E. Mahdipour, "Load Balancing Algorithms in Fog Computing: A Systematic Review," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505–1521, Mar. 2023.

[11] 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, Mar. 2019.

[12] S. Mishra, M. N. Sahoo, S. Bakshi, and J. J. P. C. Rodrigues, "Dynamic Resource Allocation in Fog-Cloud Hybrid Systems Using Multicriteria AHP Techniques," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8993–9000, Sep. 2020.

[13] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2897–2909, Sep. 2022.

[14] A. Ribeiro, G. P. Rocha Filho, D. L. Guidoni, R. E. de Grande, S. Sampaio, and R. I. Meneguette, "A Shapley Value-based Strategy for Resource Allocation in Vehicular Clouds," in *IEEE Global Communications Conference (GLOBECOM 2022)*, IEEE, Dec. 2022, pp. 5801–5806.

[15] R. Beraldi, C. Canali, R. Lancellotti, and G. Proietti Mattia, "Randomized Load Balancing under Loosely Correlated State Information in Fog Computing," in *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, Nov. 2020, pp. 123–127.