A Context-Oriented Framework and Decision Algorithms for Computation Offloading in Vehicular Edge Computing

Alisson Barbosa de Souza[1,2], **Paulo Antonio Leal Rego**[1], **José Neuman de Souza**[1]

[1]Programa de Pós-Graduação em Ciência da Computação
Universidade Federal do Ceará (UFC)
Fortaleza, CE, Brasil

[2]Universidade Federal do Ceará (UFC)
Quixadá, CE, Brasil

{alisson,pauloalr,neuman}@ufc.br

Abstract. *Some increasingly popular vehicular applications have critical time requirements. As vehicles still do not have enough computation power, they cannot satisfy these demands satisfactorily. One option to deal with this problem is to enable vehicles to transfer computational tasks to cooperating devices through the offloading technique. However, performing this technique in vehicular scenarios is challenging due to the fast movement of network nodes and the frequent disconnections. Thus, we propose a context-oriented framework and decision algorithms to reduce the execution time of vehicular applications reliably through computation offloading in vehicular edge computing systems. Experimental results show that our solutions can significantly improve the execution time of vehicular applications.*

1. Problem Statement and Contributions

The advent of autonomous vehicles and new and popular applications demand massive computing resources to deal with complicated data processing and critical latency requirements [Boukerche and Sotoro 2020, Liu et al. 2020]. Unfortunately, despite technological advances, vehicles do not yet have sufficient on-board computing resources to handle all the vehicular applications requirements in a feasible time [Zhang and Letaief 2019].

One manner to assist vehicles with latency and processing requirements is the Vehicular Edge Computing (VEC) system. In this system, computational processing can be done on vehicles or edge servers [Souza et al. 2020a]. With these resources available, the computation offloading technique can be applied to improve applications' execution time and decrease processing overload. This technique offloads smaller parts or tasks of an application to remote devices. Then, these servers process the tasks and return the result to the client vehicle [Rego et al. 2017, Xu et al. 2018].

Nevertheless, it is challenging to carry out computation offloading in VEC systems. Servers chosen to process tasks may already be overloaded. Overloaded servers and the rapid movement of nodes in these networks lead to frequent disconnections and offloading failures [Al-Sultan et al. 2014]. Furthermore, finding the optimal way to assign computation tasks to different servers for maximum reduction in execution time of vehicular applications is a Non-Deterministic Polynomial-Time (NP)-hard problem. In

addition, the entire computation offloading process must be reliable, i.e., offloading failures must be avoided, minimized, or handled. Thus, the problem addressed in this thesis can be summarized as: How to minimize the execution time of vehicular applications reliably through computation offloading in VEC systems? Based on this problem, the main contributions of this thesis are:

- A context-oriented framework for computation offloading in VEC systems.
- Three decision and task assignment algorithms to minimize the execution time of vehicular applications reliably.
- Use of several contextual parameters, including a new one called known routes of vehicles.
- Simultaneous use of WAVE and 5G technologies.
- An extensive literature review to guide future research in the area [Souza et al. 2020a].

2. Related Work

Although some works present computation offloading solutions for vehicular networks, some consider only a single type of communication technology [Sun et al. 2019] or a single server type [Rahman et al. 2020]. In addition, to improve the computation offloading process, other works consider only greedy or limited gain solutions [Feng et al. 2018, Qiao et al. 2018]. Finally, some works evaluate their solutions only in a single type of scenario [Liu et al. 2020] or vehicular density [Chen et al. 2020].

On the other hand, our solutions fill the gaps left by the other works. For example, we simultaneously use different communication technologies and server types when offloading. With this, we increase the network's transmission capacities and took advantage of all available computing resources. In addition, we use several contextual information, such as CPU capacity and availability and known routes of vehicles. This last information refers to vehicle trajectories from their on-board navigation systems shared between devices. It is used to predict vehicle positioning more accurately and avoid offloading failures. Finally, we use algorithms to assign tasks so that the best available servers execute them to minimize the applications' execution time and the number of offloading failures in any scenario.

3. Computation Offloading Framework and Decision Algorithms

Intending to improve the execution time of vehicular applications, satisfying mobility and energy constraints, we present below our framework and decision algorithms for computation offloading in vehicular edge computing systems.

3.1. Computation Offloading Framework

Figure 1 shows the *Application* and *Partitioner* modules and conceptual architecture of the framework. The box with dotted lines represents an application running on a vehicle, along with the *Partitioner* module and the proposed framework. The latter is represented by the smaller box with a gray background. The arrows indicate the direction of the information flow between the modules.

The *Application* module sends data to a *Partitioner* module in order to analyze whether the application workload can be partitioned. If it is possible, the *Partitioner*

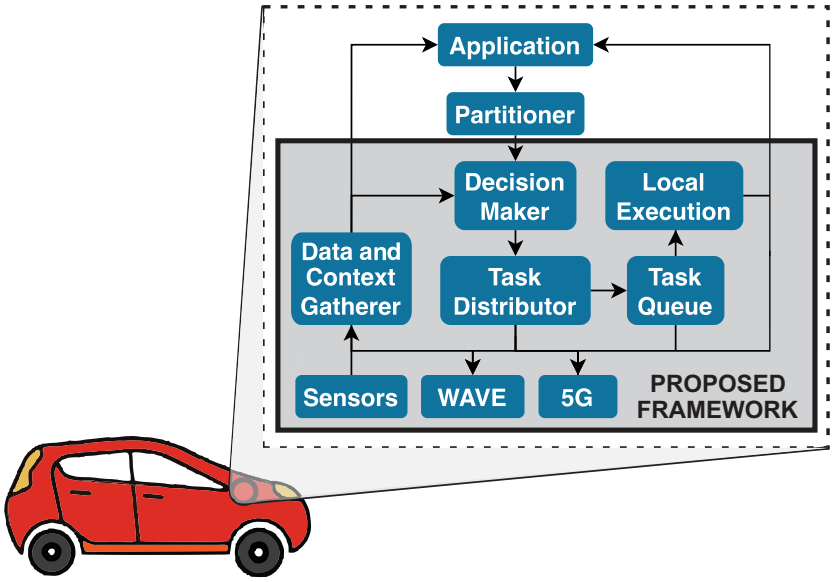

Figure 1. Architecture of the framework.

divides the application workload into smaller tasks that can be executed on different devices and in a parallel, asynchronous, and independent way. The application workload then moves to the *Decision Maker* module, which decides where each task should execute and reports the decision to the *Task Distributor* module. The latter distributes tasks for local or remote execution. After the workload has been processed, the *Application* receives the results through the *Local Execution* or the *Data and Context Gatherer* module, when the results come from remote devices. The *Application* also receives information from local sensors and other devices. This information is captured through the *Data and Context Gatherer* module.

Such framework manages all the steps of the computation offloading process, which are: (1) Resources discovery, (2) Offloading decision, (3) Send/receive tasks, and (4) Failure recovery. In step 1, the *Decision Maker* module receives tasks from the *Application/Partitioner* and triggers the resources discovery. At this time, the client vehicle sends requests through different communication technologies. If the servers meet the previously established criteria, they accept to participate in the offloading process and send their contextual information.

After waiting a certain amount of time, the client proceeds to the task assignment/scheduling decision (step 2). Then, after congregating all the necessary information, the *Decision Maker* module executes an algorithm to decide the assignment/scheduling of tasks. It must assign each task to a server (local or remote) to execute and choose the communication technology to be used.

In step 3, after the decision, the *Decision Maker* module transfers the tasks to the *Distributor* module that forwards the tasks to the appropriate modules (for local or remote execution). In this way, as shown in Figure 2, the client (red) distributes tasks to remote servers (vehicles and edge server) to start execution. Thus, multiple servers can simultaneously collaborate to provide computing services to the client. In this approach, server vehicles may be more likely to cause offloading failures due to mobility. However,

they help to parallelize task processing and are important sources of computational power. Furthermore, if the distribution of tasks is well calculated and coordinated, failures can be minimized.

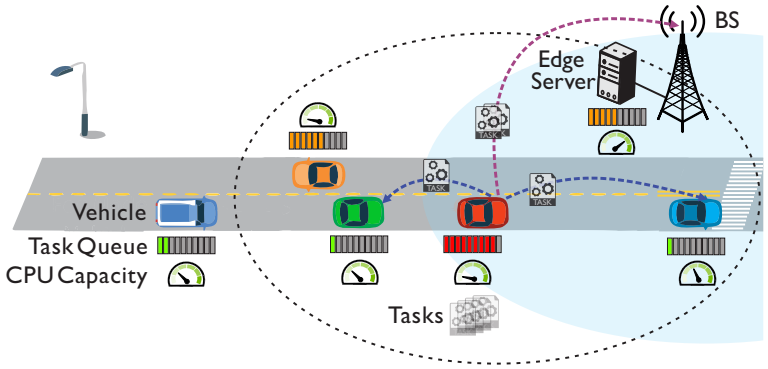

Figure 2. Sending of tasks to the chosen servers.

Following processing, each chosen server returns the processing result to the client. Suppose any failure is detected in the offloading process. In that case, the *Data and Context Gatherer* module informs the *Decision Maker* module in step 4. Then, the *Task Distributor* module, which has copies of the lost tasks, distributes them to be executed locally on the client. Hence, even if the client loses contact with some servers, the lost tasks are re-executed, ensuring that all are processed.

3.2. Decision Algorithms

The decision process is the core of the framework described in the previous section. This process assigns each application task to execute on a server. With enough contextual information at its disposal, the *Decision Maker* module executes an algorithm to decide the assignment/scheduling of tasks. The objective of the decision is to minimize the applications' execution time, trying to optimize task distribution and following established constraints to avoid offloading failures and lack of energy on the network nodes. The following sections present the developed decision algorithms.

3.2.1. GCF Algorithm

Greedy for CPU Free (GCF) is a decision and task assignment algorithm that prioritizes sending tasks to servers with the highest processing availability (lowest queuing time) and the shortest distances to the client. This prioritization is done by sorting the set of feasible servers. Such sorting is done only once for each workload. In the event of a tie in the server evaluation, the algorithm breaks the tie by the following order of priority: edge servers, client, and nearby vehicles. After sorting, the algorithm goes through the set of feasible servers and allocates as many tasks as possible to its servers.

3.2.2. GTT Algorithm

Greedy Task by Task (GTT) is a decision and task assignment algorithm that seeks the best possible server to execute each application task. The algorithm classifies a server as

better mainly by the following context parameters: CPU capacity, CPU availability, and distance to the client. Thus, the best server tends to have high CPU capacity, low queue time, and a short distance to the client. Moreover, the algorithm updates the best servers list in real-time as it decides where to send tasks and uses information about known routes of vehicles. Therefore, the main characteristics of the GTT that differentiate it from the GCF are: use of contextual information of known routes of vehicles and CPU capacity; case-by-case analysis to allocate each task (first traversing the set of tasks and not the set of feasible servers); update of the best server at each allocation with different criteria.

3.2.3. BCV Algorithm

Artificial Bee Colony (ABC) for Computation Offloading in Vehicular Edge Computing (BCV) is a decision, task scheduling, and intelligent algorithm based on the ABC meta-heuristic. This algorithm is inspired by the behavior of honey bees when searching for food sources. As shown in Figure 3, the BCV considers a food source (nectar from a flower) as a feasible solution. This solution is an association between each computation task in the workload (τ_1, τ_2, τ_3, τ_4) and the server where it will be executed. An infeasible solution is one that can cause offloading failures. These possible failures are predicted by calculations of the algorithm. They indicate a future lack of connectivity between client and server or of energy of some node.

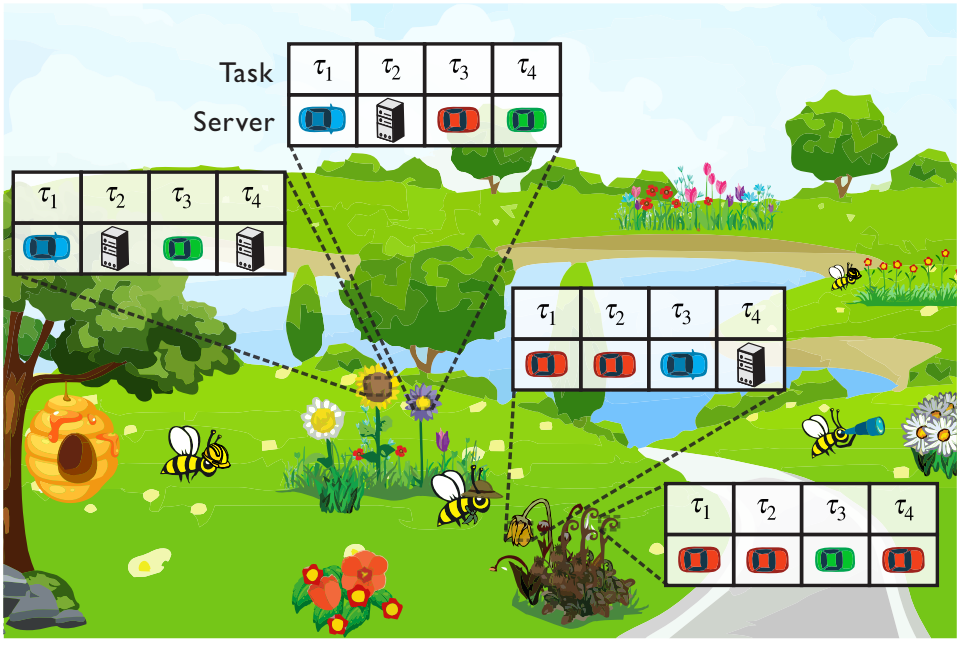

Figure 3. Figurative depiction of bees looking for solutions in the BCV search space.

Figure 3 uses the scenario in Figure 2, which shows the servers available for the client (red vehicle). The algorithm evaluates the fitness of each solution based on the sum of the time to execute each workload task on the different chosen servers (t_{total}). The lower t_{total}, the greater the fitness of the solution.

4. Evaluation

We employed the SUMO[1] and ns-3[2] simulators with a 5G/mmWave[3] module to evaluate our offloading algorithms. Simulations were used because they reduce the financial costs of evaluations, avoid logistical difficulties, provide increasingly robust and realistic models, allow reproducible and better-controlled tests, and are the most used technique to perform network experiments [Souza et al. 2020a]. The simulations were performed on a computer with an Intel Xeon E5645 processor @ 2.40 GHz and 32 GB RAM. In terms of scenarios, we used an adapted stretch of a Brazilian highway with 5 km in length and an adapted urban stretch of 2 km^2 of Manhattan, New York, USA. Besides, we used three types of vehicular densities in each scenario: 11, 55, and 120 vehicles/km in the highway scenario and 25, 120, and 250 vehicles/km^2 in the urban scenario.

Initially, we made some preliminary assessments of parameters such as the communication range, configurations of the BCV algorithm, and the percentage of known routes of vehicles. Regarding the last parameter, although some public vehicles (e.g., buses) already make this information available, not all drivers of private vehicles may be willing to make it available for privacy reasons. Therefore, we carried out experiments to evaluate how the algorithms that use this information (GTT and BCV) are impacted according to different percentages of vehicles that provide their routes. From this evaluation, we concluded that the more vehicles with known routes, the greater the reduction of computation offloading failures.

Then, we compared the performance of BCV with other four algorithms: FIFO (First In, First Out - used to show the behavior of choosing servers at random) [Souza et al. 2021], HVC (Hybrid Vehicular edge Cloud - an of the main and most cited state-of-the-art solutions) [Feng et al. 2018], GCF (Greedy for CPU Free) [Souza et al. 2020b] and GTT (Greedy Task by Task) [Souza et al. 2021]. The performances of the different algorithms were evaluated with a massive amount of experiments according to the two main metrics of this thesis: *tasks by occurrence type* and *reduction in execution time*. The first metric evaluates the percentage of successes and failures in the offloading processes. The second metric evaluates how much each algorithm has reduced in terms of the workload execution times when compared to the totally local execution by the client. We saw that the developed algorithms outperformed the totally local execution and the FIFO and HVC algorithms in the two metrics.

In the *tasks by occurrence type* metric, the BCV, GTT, and GCF outperformed FIFO and HVC in 41, 40, and 39 of 60 cases of failure rates, respectively. In some groups of experiments, the BCV, GTT, and GCF presented, respectively, up to 0.0 %, 0.1 %, and 1.1 % of failures. As can be observed, there is no guarantee that all the offloading processes will be successful. The offloading failures occur because vehicular mobility is not entirely and precisely predictable. Even so, the number of failures was minimized with the three developed algorithms. Moreover, all the lost tasks by network partition were recovered and executed locally on client vehicles.

In the *reduction in execution time* metric, the Table 1 shows that the BCV outper-

[1]https://sumo.dlr.de/

[2]https://www.nsnam.org

[3]https://apps.nsnam.org/app/mmwave/

formed the FIFO and HVC algorithms in 59 of 60 groups of experiments, with a reduction record of 75.6 % and showing the best performance in 48 of 60 groups of experiments. From the evaluation, we concluded that the BCV algorithm had the best performance for different workloads and presented the best ways to reduce the execution time of vehicular applications reliably.

Table 1. Results summary of the algorithms in *reduction in execution time* metric.

	Outperformed FIFO and HVC	Reduction Record	Best Performance
BCV	59/60	75.6 %	48/60
GTT	51/60	73.9 %	28/60
GCF	36/60	66.5 %	1/60

The results presented are significant because they show that the computation offloading technique, when well managed, can reliably improve the performance of vehicular applications through other cooperating devices in the vehicular environment. Therefore, we believe that the work developed in this thesis can offer a promising alternative to enable the execution of these applications.

5. Publications

The list of publications related to this thesis is as follows.

1. **A. B. Souza**, P. A. L. Rego, T. Carneiro, P. H. G. Rocha, and J. N. Souza. "A Context-Oriented Framework for Computation Offloading in Vehicular Edge Computing using WAVE and 5G Networks". **Vehicular Communications**, volume 32, 2021. 96 % (Scopus highest percentile).

2. **A. B. Souza**, P. A. L. Rego, T. Carneiro, J. C. Rodrigues, P. P. R. Filho, J. N. Souza, V. Chamola, V. H. C. Albuquerque, and B. Sikdar. "Computation Offloading for Vehicular Environments: A Survey". **IEEE Access**, volume 8, 2020. 87 % (Scopus highest percentile).

3. **A. B. Souza**, P. A. L. Rego, P. H. G. Rocha, T. Carneiro, and J. N. Souza. "A Task Offloading Scheme for WAVE Vehicular Clouds and 5G Mobile Edge Computing". IEEE Global Communications Conference - **IEEE GLOBECOM**, 2020, Taipei, Taiwan. A1 (Qualis[4]).

4. **A. B. Souza**, P. A. L. Rego, and J. N. Souza. "Exploring Computation Offloading in Vehicular Clouds". IEEE International Conference on Cloud Networking - **IEEE CloudNet**, 2019, Coimbra, Portugal. B1 (Qualis).

5. A. M. Sousa, **Souza, A. B.**, P. A. L. Rego, J. N. Souza, and J. A. F. Macedo. "Um Algoritmo de Offloading Computacional para Nuvens Veiculares". Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - **SBrT**, 2017, São Pedro, Brasil.

[4]Period: 2013-2016; Area: Computer Science.

References

Al-Sultan, S., Al-Doori, M. M., Al-Bayatti, A. H., and Zedan, H. (2014). A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392.

Boukerche, A. and Sotoro, V. (2020). Computation offloading and retrieval for vehicular edge computing: Algorithms, model and classification. *ACM Computing Surveys (CSUR)*, 53(4):1–35.

Chen, C., Chen, L., Liu, L., He, S., Yuan, X., Lan, D., and Chen, Z. (2020). Delay-optimized v2v-based computation offloading in urban vehicular edge computing and networks. *IEEE Access*, 8:18863–18873.

Feng, J., Liu, Z., Wu, C., and Ji, Y. (2018). Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling. *IEEE Vehicular Technology Magazine*, 14(1):28–36.

Liu, Y., Wang, S., Zhao, Q., Du, S., Zhou, A., Ma, X., and Yang, F. (2020). Dependency-aware task scheduling in vehicular edge computing. *IEEE Internet of Things Journal*, 7(6):4961–4971.

Qiao, G., Leng, S., Zhang, K., and He, Y. (2018). Collaborative task offloading in vehicular edge multi-access networks. *IEEE Communications Magazine*, 56(8):48–54.

Rahman, A. U., Malik, A. W., Sati, V., Chopra, A., and Ravana, S. D. (2020). Context-aware opportunistic computing in vehicle-to-vehicle networks. *Vehicular Communications*, 24:100236.

Rego, P. A., Costa, P. B., Coutinho, E. F., Rocha, L. S., Trinta, F. A., and de Souza, J. N. (2017). Performing computation offloading on multiple platforms. *Computer Communications*, 105:1–13.

Souza, A. B., Rego, P. A., Carneiro, T., Rodrigues, J. D. C., Rebouças Filho, P. P., De Souza, J. N., Chamola, V., De Albuquerque, V. H. C., and Sikdar, B. (2020a). Computation offloading for vehicular environments: A survey. *IEEE Access*, 8:198214–198243.

Souza, A. B., Rego, P. A. L., Carneiro, T., Rocha, P. H. G., and de Souza, J. N. (2021). A context-oriented framework for computation offloading in vehicular edge computing using wave and 5g networks. *Veh. Commun.*, 32:100389.

Souza, A. B., Rego, P. A. L., Rocha, P. H. G., Carneiro, T., and de Souza, J. N. (2020b). A task offloading scheme for wave vehicular clouds and 5g mobile edge computing. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE.

Sun, Y., Guo, X., Song, J., Zhou, S., Jiang, Z., Liu, X., and Niu, Z. (2019). Adaptive learning-based task offloading for vehicular edge computing systems. *IEEE Transactions on Vehicular Technology*, 68(4):3061–3074.

Xu, D., Li, Y., Chen, X., Li, J., Hui, P., Chen, S., and Crowcroft, J. (2018). A survey of opportunistic offloading. *IEEE Communications Surveys & Tutorials*, 20(3):2198–2236.

Zhang, J. and Letaief, K. B. (2019). Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*, 108(2):246–261.