A Game Theory-Based Vehicle Cloud Resource Allocation Mechanism

Henrique Andrews Prado Marques[1], **Rodolfo I. Meneguette**[1]

[1]Institute of Mathematics and Computer Science - University of São Paulo (USP)
São Carlos, SP – Brasil.

henrique.andrews.marques@usp.br, meneguette@icmc.usp.br

Abstract. *The vehicle cloud aims at efficient cooperation in communication, task allocation, and sharing of resources in VANETs since computational resources embedded in the vehicle can be used to offer resources for the provision of cloud services. This requires efficient resource management mechanisms that allocate these resources to maximize their use. Thus, in this paper, we propose a resource allocation mechanism based on Game Theory to maximize the use of resources made available by vehicles. The results obtained showed greater use of the resources made available by the vehicles compared to other works in the literature.*

1. Introduction

With each passing year, there is a significant increase in the number of vehicles around the world. As a result, the number of connected vehicles circulating on the streets among us also grows, sharing more data than ever before. It is estimated that by 2025 there could be nearly 2 billion vehicles connected to roads around the world [CISCO 2020]. Accompanied by this growth, the automotive industry, in partnership with technology industries, has been investing considerably in the addition of more and better technological resources in vehicles, such as sensors and chipsets specific to the scenario. There is greater processing power, communication, and variety of use cases [Qualcomm 2020], thus contributing to an expansion and creation of new concepts.

One of the new concepts that emerged was vehicular clouds (VCs), which aim to coordinate communication, task allocation, and sharing resources in vehicles with or without an accommodation infrastructure[Lieira et al. 2021]. Such computational resources as processing, storage and communication can be used for decision making without the need for traditional cloud computing for such [Brik et al. 2019, Pereira et al. 2021].

By adding the excess of resources that are at the edge of the network, a set of services is created that is made available to vehicles, which can be used in static and dynamic environments [Hagenauer et al. 2019, Correa et al. 2014]. Thus, vehicles become critical components as they start to act in providing services and decision-making. Therefore, mechanisms that make efficient use of VC resources are essential to use these services in the best possible way.

The main challenges involving resource allocation mechanisms in vehicular clouds for the provision of services and applications revolve around the dynamic nature of vehicular networks [Meneguette et al. 2019a]. The allocation and processing of tasks with delay-intolerant requirements, such as accident detection, or tasks that require high computational power, as traffic image processing and other multimedia applications in general, ends up being quite challenging due to the high mobility of the vehicles [Meneguette et al. 2019b].

Thus, to better take advantage of the resources made available by the vehicles, this work proposes a resource allocation mechanism based on Game Theory to maximize their use and then compare this approach with others known in the literature and verify its performance about other metrics. Evaluation beyond the use of resources.

The rest of the article is organized as follows: Section 2 presents works related to the allocation of resources and tasks. In Section 3 the formulation of the problem, the application scenario, and the developed algorithm are discussed. In Section 3.4 the results obtained through the evaluations are presented and discussed, and finally, in Section 4 the conclusions and guidelines for future work are presented.

2. Trabalhos Relacionados

There are works in the literature that address the problem of resource allocation in VCs. For example, in the work of [Pereira et al. 2019], a policy for allocation of resources in VCs in a road environment is proposed. The proposed policy aims to maximize the availability of resources in VC. Connected vehicles must cooperate to create a reservoir of resources to know the available resources, which will be provided by the vehicles and the set of fogs where more resources will be generated and services allocated.

In [da Costa et al. 2020] they propose a mechanism for task allocation in VCs based on combinatorial optimization and compare it with three other approaches. The results obtained show both more effective use of available resources and a more significant number of tasks allocated and greater gain than the other methods. A real-time urban scenario is simulated where tasks are generated throughout the execution and are allocated in VCs that are recomputed every minute due to the mobility of vehicles. For the generation of VCs and aggregation of resources, a clustering technique well known in the literature is used, the DBSCAN *Density Based Spatial Clustering of Application with Noise* [Ester et al. 1996].

In [Lieira et al. 2020] we propose an algorithm that adopts the meta-heuristic technique known as *Grey Wolf Optimization* [Mirjalili et al. 2014] to choose the best *Edge* when allocating the resources of the vehicle. In this work, it is considered that vehicles have processing, storage, time, and memory resources. The algorithm makes use of these resources to compute the adequacy of each *Edge* and decide which one to allocate to, if possible. This approach is compared with two other policies and obtained a lower number of refused services and presented a low number of locks while searching for a *Edge*.

3. A Game Theory-Based Vehicle Cloud Resource Allocation Mechanism

This section will describe the AVR - A Game Theory-Based Vehicle Cloud Resource Allocation Mechanism, which aims to use idle vehicle resources plus the computational resources available by RSUs (roadside units) to offer services to VCs. For this, we consider that vehicles traveling in the city form VCs through clustering. When a vehicle does not belong to any VC but needs to perform the processing of some task and its computational resources are not sufficient, AVR is asked for the resources necessary to fulfill the task.

A scenario composed of v vehicles is considered, where each vehicle v_i has a unique identification ($i \in [1, x]$) and is equipped with an OBU *On-Board Unit* that allows both communications between vehicles and between vehicles and MSW. RSUs collect information from vehicles in real-time and when they request resources to allocate their tasks. Through the intercommunication of the RSUs, it is possible to perform the AVR, whose role is to define, given a set of available VCs, which one can meet a certain task that requires a certain amount of computational power.

AVR executes the game theory-based mechanism to select some of the available VCs to meet the resource request and allocate the task, as quickly as possible and use the maximum VC's idle computational resources.

3.1. Problem Definition

Considering a set of tasks T with $\{t = 1, ..., n\}$, where each task t is represented by a tuple (id_t, p_t, g_t), where id_t corresponds to the unique identifier of each task, p_t the task

weight (amount of resources needed to be allocated) and g_t the gain/reward for successful allocation. In this scenario, the VCs are vehicles formed from a clustering process based on some pre-established criteria. Each VC corresponds to the sum of the resources shared by each vehicle that is part of the respective cloud.

In this work, the focus of the allocation policy based on Game Theory will be on maximizing the use of computational resources from vehicular clouds within the limitations of the game model proposed. The game type is non-cooperative, which focuses on predicting the actions and rewards of individual players and analyzing Nash's equilibrium, and simultaneous, where players act without knowing what action others will take. Thus, players seek to maximize their profits or minimize their losses. Therefore, the problem is intended to:

$$\max \sum_{t=1}^{n} p_t \alpha_t \leq \Omega_j$$

3.2. Game Modeling

A vehicle that demands a service can choose between consuming or not the offered service (allowing or not the allocation of a particular task) given the cost for such. At the same time, VC also has two options, whether or not to offer the necessary computing resources to execute the task.

The main idea of the proposed game model is that both parties involved (consumer vehicles and VCs/provider vehicles) have some gain, be it money when offering a service in which there is a cost involved or added value by providing the service requested. The secondary idea is for the game to operate according to the supply of computational resources and the weight of the tasks, thus making the cost vary at the expense of these factors. For example, if few resources are available, the price per resource increases, and if there is a large availability of resources, their price decreases.

Below are the winning equations for each combination of players' strategies:

- **Do not consume & Offer :**

$$X_{21} = -2 \cdot g_t$$
$$Y_{21} = -p_t \cdot c(p_t, \Omega_j)$$

- **Consume & Offer :**

$$X_{11} = g_t - p_t \cdot c(p_t, \Omega_j)$$
$$Y_{11} = p_t \cdot c(p_t, \Omega_j)$$

- **Do not consume & Does not offer r:**

$$X_{22} = 0$$
$$Y_{22} = 0$$

- **Consume & Do not offer :**

$$X_{12} = -g_t$$
$$Y_{12} = -2 \cdot p_t \cdot c(p_t, \Omega_j)$$

⋆ X_{ij} e Y_{ij} represent the earnings of consumers and providers respectively.
⋆ $c(p_t, \Omega_j)$is a cost per resource as a function of the amount of available computational resources Ω_j in the VC with the highest offer and p_t the weight of the task in question.

To find the Nash Equilibrium, we analyze each strategy combination:

– **Strategy combination (Do not consume & Offer)**
In this third case, in which the actions are not to consume and offer, the consumer has a double negative reward (X_{21}) as a form of penalty, and the provider has a negative reward (Y_{21}) for not have their computing resources used. As with the previous combination, this is not a Nash equilibrium for the same reasons that both players could increase their earnings by changing their actions.

– **Strategy combination (Consume & Offer)**
In this first case, the reward of the person who consumes the service is given by X_{11}, where g_t, as defined above, is the value added by the execution of the task t and p_t its weight in the number of resources needed for processing. The reward of the vehicular cloud vc_j that provides the computational resources to fulfill the task t is given by Y_{11}, where $c(p_t, \Omega_j)$ is a function that varies from according to the weight p_t of the task t and the number of available resources Ω_j of the VC. Depending on the result of X_{11}, neither player can increase their win by switching strategy, as the other wins are neutral or negative, so this combination is a possible Nash equilibrium.

– **Strategy combination (Do not consume & Do not offer)**
In the latter case, as neither party is interested in consuming or offering the service, there is no need for the penalty, and the gain for both is neutral ($X_{22} = Y_{22} = 0$). Similar to the first case, in this combination, none of the players can raise their reward by choosing the other strategy, then configuring it as a Nash equilibrium.

– **Strategy Combination (Consume & Don't Offer)**
In this second case, where there is a choice to consume but not to offer, the consumer's gain (X_{12}) is negative since his task is not fulfilled. The provider will be penalized with a double negative gain (Y_{12}) because before the formation of the game, it is considered that there are enough resources to fulfill the task t with weight p_t. This combination does not represent a possible Nash balance as either the X player could increase their reward by choosing not to consume as the Y player choosing to offer.

3.3. Implementation of the allocation mechanism

In the source code 1 it is possible to see how the mechanism works. The list of VCs W and the set of tasks T are passed as function parameters.

The policy works as long as there are tasks in the list, sorted by weight p_t in descending order. This is done to ensure that the most significant amount of resources is used, as if lighter tasks are serviced first, there may not be enough resources for heavier ones later. If the most resource-abundant VC has enough resources to fulfill the task at hand, then the game is set up, and the policy continues to work. Otherwise, it moves on to the next task.

Each game generated in the implementation follows the model described in the Game Modeling subsection. A game is created for each task and amount of resources in the most abundant VC, as each reward matrix and the players' strategy gain varies with the task value g_t, weight p_t, and cost per resource $c(p_t, \Omega_j)$.

For each game, the strategy combination(s) that make up a Nash equilibrium is found. If the combination of strategy *Consume & Offer* is a Nash equilibrium, the task allocation is performed. Therefore, the resources used are subtracted. Then, the task weight is added to the accumulated weight, the task value to the accumulated value, and the task is included in the list of allocated ones. At the end of the code, the total accumulated value, the number of tasks attended, and the total accumulated weight is returned.

Algorithm 1: Implementation of AVR

```
VCC=[]
for i,j in W.items() do
    VCC.append(j)
Tasks_allocated = []
weight_accumulated = 0
value_accumulated = 0
Tasks = sorted(T, key=weight, reverse=True)
while len(Tasks) > 0 do
    tasks = Tasks.pop(0)
    index_max = VCC.index(max(VCC))
    max_resource = max(VCC)
    if max_resource >= weight(tasks) then
        p1_11 = value(tasks) - (weight(tasks) * cost_per_weight_unit(weight(tasks),
          max_resource))
        p1_12 = -value(tasks)
        p1_21 = -2*value(tasks)
        p1_22 = 0
        player1 = np.array([[p1_11, p1_12], [p1_21, p1_22]])
        p2_11 = weight(tasks) * cost_per_weight_unit(weight(tasks), max_resource)
        p2_12 = -2 * weight(tasks) * cost_per_weight_unit(weight(tasks), max_resource)
        p2_21 = -weight(tasks) * cost_per_weight_unit(weight(tasks), max_resource)
        p2_22 = 0
        player2 = np.array([[p2_11, p2_12], [p2_21, p2_22]])
        Array_of_Reward = nash.Game(player1, player2)
        eqs =Array_of_Reward.support_enumeration()
        nash_equilibrium = list(eqs)
        if (nash_equilibrium[0][0] == [1., 0.]).all() and (nash_equilibrium[0][1] == [1.,
          0.]).all() then
            Tasks_allocated.append(tasks)
            VCC[index_max] -= weight(tasks)
            weight_accumulated += weight(tasks)
            value_accumulated += value(tasks)
    else
        continue
return value_accumulated , len(Tasks_allocated), weight_accumulated
```

3.4. Simulation and Results

The simulation was performed using SUMO *Simulation of Urban Mobility*[1] version 1.4.0, an open source urban mobility simulator. The algorithms presented were implemented in the language *Python*[2] and the connection with SUMO made through TraCI*Traffic Control Interface*[3]. In order to make the urban traffic scenario as realistic as possible, the mobility *trace Luxembourg SUMO Traffic (LuST)* [Codeca et al. 2017] was used.

This scenario contains 24 hours of urban mobility with up to approximately 5000 vehicles circulating during peak hours. The chosen time range was from 11:00 am to 12:00 pm, when low vehicular density was low. Consequently, there is a greater scarcity of shared computing resources in the VCs to assess how allocation policies operate when the situation is more challenging.

The clustering interval defined was the 60s due to the dynamics of the network and recurrent changes in the VCs. Tasks are generated when the vehicles are grouped, and then the policies allocate them to the available VCs. The occurrence of Tasks in the system, independent of each other, is defined following a Poisson distribution with

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

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

[3]https:// sumo.dlr.de/docs/TraCI.html

an average of Tasks $\lambda = 25$. The chosen communication radius was 100 meters and a minimum of 2 vehicles per vehicle cloud. There was also variation in the average weight p_t of Tasks generated in the simulations with $\mu = \{1, 5, 10, 15, 20\}$, the allocation gain g_t varied with $2 \times \mu$ and the amount of shareable computational resources per vehicle in $\omega_i = \{1, 2, 3\}$. All these parameters used are summarized in Table 1

Tabela 1. Simulation Parameters

Parameter	value
Scenery area	$155.95\ km^2$
Simulation time	1h of LuST Scenario (11h-12h)
Clustering Interval	60s
Clustering Algorithm	DBSCAN
Occurrence of Tasks	Poisson Distribution
Average number of Tasks (λ)	25
Average weight of Tasks	$\{1, 5, 10, 15, 20\}$
Average Earning of Tasks	$2 \times \{1, 5, 10, 15, 20\}$
Amount of resources per vehicle	$\{1, 2, 3\}$

Below are the metrics used to evaluate the performance of the algorithms:

1. **Task service**
   Represents the percentage of successfully allocated tasks.

$$service = \frac{\sum_{t=1}^{n} \alpha_t \mid \alpha_t \in S}{|T|} \cdot 100$$

2. **Resource usage**
   Represents the percentage of VC computational resources used.

$$usage = \frac{\sum_{t=1}^{n} p_t \mid p_t \in S}{\sum_{j=1}^{m} \Omega_j} \cdot 100$$

3. **Allocation gain**
   Represents total value added by task allocation.

$$gain = \sum_{t=1}^{n} g_t \mid g_t \in S$$

⋆ S represents the set of Tasks that were successfully allocated.

To evaluate the performance of the allocation policy of this work, there is a comparison with others, which are an approach involving dynamic programming which will be referred to in the results as DP Dynamic Programming, a greedy referred to as Greedy in which the algorithm considers only one VC and finally a greedy referred to as Greedy-N where the algorithm considers all available VCs.

Tables 2, 3 and 4, shows the percentage of tasks served by each algorithm. Task fulfillment results have similar behavior to gain, as these two metrics are somewhat related. The more tasks a policy fulfills, the trend is that the greater the total allocation gain. There may be situations in which few high value-added tasks are met and result in a high gain, which is the case with AVR as it prioritizes the attendance of heavier tasks, which tend to have a greater reward. Still, this approach ends up showing itself as not the best possible since the Greedy-N algorithm got better results.

The tables 5, 6 and 7 show the percentage of resources used by each algorithm in its allocations. When observing the results, AVR fulfills its main objective, which is

Tabela 2. Task fulfillment percentage by average task weight λ and shared resources $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	5,41%	5,41%	5,41%	2,78%	2,78%
DP	4,88%	4,88%	4,88%	2,9%	2,9%
Greedy-N	16,48%	12,38%	7,12%	8,58%	3,03%
AVR	12,66%	9,22%	6,59%	8,71%	3,15%

Tabela 3. Task fulfillment percentage by average task weight λ and shared resources $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	10,82%	8,04%	8,04%	5,41%	5,26%
DP	9,63%	6,85%	4,88%	4,88%	4,07%
Greedy-N	32,71%	20,43%	12,38%	11,07%	2,88%
AVR	24,94%	12,66%	9,22%	8,56%	1,69%

Tabela 4. Task fulfillment percentage by average task weight λ and shared resources $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	16,23%	13,6%	13,45%	13,45%	5,41%
DP	14,38%	7,65%	6,85%	11,6%	4,88%
Greedy-N	48,93%	28,9%	19,26%	17,94%	5,66%
AVR	37,22%	13,97%	12,66%	14,78%	4,47%

to maximize the use of available computational resources, having a better result than all other algorithms. However, it is visible that greater use of resources does not directly imply greater service or greater gain, since, in the other two metrics, AVR had a lower performance than Greedy-N. When prioritizing heavier Tasks, AVR inevitably ends up serving a smaller number of Tasks. Even though they are heavier, it is not guaranteed that they have a considerably higher aggregate value than other Tasks that require a smaller amount of resources from the VCs.

Tabela 5. Resource utilization percentage by average task weight λ and shared resources $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	25%	25%	16,67%	16,67%
DP	22,92%	25%	25%	18,75%	16,67%
Greedy-N	75%	79,17%	58,33%	54,17%	16,67%
AVR	62,5%	81,25%	66,67%	56,25%	16,67%

Tabela 6. Resource utilization percentage by average task weight λ and shared resources $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	16,67%	25%	12,5%	8,33%
DP	22,92%	25%	25%	25%	8,33%
Greedy-N	75%	91,67%	75%	56,25%	18,75%
AVR	62,5%	100%	79,17%	57,29%	23,96%

In the tables 8, 9 and 10, the reward values that each algorithm obtained according to the tasks that were allocated by them are presented. It is observed that in general, AVR obtained greater gains than the *Greedy* and DP approaches, which only use VC with a greater abundance of computational resources, but obtained smaller gains when

Tabela 7. Resource utilization percentage by average task weight λ and shared resources $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	22,22%	19,44%	25%	19,44%
DP	22,92%	25%	25%	25%	20,83%
Greedy-N	75%	86,11%	73,61%	72,22%	31,94%
AVR	62,5%	100%	93,75%	73,61%	34,03%

compared to the *a*pproach Greedy-N, which makes use of all available VCs to perform task allocation. It is also worth emphasizing that as the average weight of tasks increases, the gain of policies becomes closer due to the difficulty of meeting increasingly heavy tasks in VCs with the same amount of resources as before.

Tabela 8. Task reward by average task weight λ and resources per vehicle $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	2	2	2	1	1
DP	1,75	1,75	1,75	1	1
Greedy-N	6	4,5	2,5	3	1
AVR	4,5	3,25	2,25	3	1

Tabela 9. Task reward by average task weight λ and resources per vehicle $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	4	3	3	2	2
DP	3,5	3,5	3,5	3	1,5
Greedy-N	12	9	7	5,5	1
AVR	9	5,875	5	4	0,5

Tabela 10. Task reward by average task weight λ and resources per vehicle $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	6	5	5	6	2
DP	5,25	5,25	4,75	5,25	1,75
Greedy-N	18	12,5	11	9	2
AVR	13,5	11,125	8,25	7,25	1,5

4. Conclusion

In this article, he presented and detailed several essential elements of the resource allocation mechanism, starting with the system model and description of the problem in which the project is inserted, then discussing the activities carried out to reach the proposed solution. The gains in the adopted strategies and rationality in their decisions, better adapting to a reality where there is a cost involved in the use of resources and services provided in ITS and VANETs. This work also proves that greater use of computational resources is not necessarily accompanied by a more significant allocation gain or a greater number of Tasks fulfilled, a fact highlighted in the analysis of the results, meaning that there is still plenty of room for improvement and work to be done developed in this policy. As future works, we intend to improve the players' strategy to better the system to gain more significance.

Acknowledgment

The authors thank the Foundation for Research Support of the State of São Paulo (FAPESP) process number #2020/07162-0 for the financial support for the development of this research.

Referências

Brik, B., Khan, J. A., Ghamri-Doudane, Y., and Lagraa, N. (2019). Publish: A distributed service advertising scheme for vehicular cloud networks. In *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6.

CISCO (2020). Driving profits from connected vehicles. Technical report.

Codeca, L., Frank, R., Faye, S., and Engel, T. (2017). Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63.

Correa, C., Ueyama, J., Meneguette, R. I., and Villas, L. A. (2014). Vanets: An exploratory evaluation in vehicular ad hoc network for urban environment. In *2014 IEEE 13th International Symposium on Network Computing and Applications*, pages 45–49.

da Costa, J., Peixoto, M., Meneguette, R., Rosário, D., and Villas, L. (2020). Morfeu: Mecanismo baseado em otimização combinatória para alocação de tarefas em nuvens veiculares. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 505–518, Porto Alegre, RS, Brasil. SBC.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226–231. AAAI Press.

Hagenauer, F., Higuchi, T., Altintas, O., and Dressler, F. (2019). Efficient data handling in vehicular micro clouds. *Ad Hoc Networks*, 91:101871.

Lieira, D. D., Quessada, M. S., Cristiani, A. L., and Meneguette, R. I. (2020). Resource allocation technique for edge computing using grey wolf optimization algorithm. In *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6.

Lieira, D. D., Quessada, M. S., da Costa, J. B. D., Cerqueira, E., Rosário, D., and Meneguette, R. I. (2021). Tovec: Task optimization mechanism for vehicular clouds using meta-heuristic technique. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 358–363.

Meneguette, R. I., Boukerche, A., and Pimenta, A. H. M. (2019a). Avarac: An availability-based resource allocation scheme for vehicular cloud. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3688–3699.

Meneguette, R. I., Rodrigues, D. O., da Costa, J. B. D., Rosario, D., and Villas, L. A. (2019b). A virtual machine migration policy based on multiple attribute decision in vehicular cloud scenario. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61.

Pereira, R., Boukerche, A., da Silva, M. A., Nakamura, L. H., Freitas, H., Rocha Filho, G. P., and Meneguette, R. I. (2021). Foresam—fog paradigm-based resource allocation mechanism for vehicular clouds. *Sensors*, 21(15):5028.

Pereira, R. S., Lieira, D. D., da Silva, M. A., Pimenta, A. H., da Costa, J. B., Rosário, D., and Meneguette, R. I. (2019). A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In *2019 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6.

Qualcomm (2020). Connecting vehicles to everything with c-v2x. Technical report.