Heuristic-Driven Resource Allocation in Vehicular Fog Computing via Reinforcement Learning and Gradient Optimization

Jiahao Pang
Computer science of Brock University
St Catharines, Canada
jp17gh@brocku.ca

***Abstract*—Vehicular Fog Computing(VFC) which utilizes the potential resources between the cloud and the edge of Vehicular Networks to improve the ability to compute for the edge in fog is a significant development of intelligent transportation. VFC has characteristics: low latency, high dynamic, a large number of nodes, heterogeneity, and so on. Due to decentralization and high mobility, the resource is hard to manage. High mobility is a big challenge for resource allocation. Some requests require resource vehicles to stay in the fog during process time. Vehicles in processing leave fog will cause request failure. Therefore, in this article, the problem of reducing the probability of losing resource vehicles in resource allocation is formulated. Then a heuristic algorithm that considers the resource amount, position, and mobility is proposed to sort the resource vehicle in the pool, and reinforcement learning with the gradient method is used to optimize the heuristic algorithm. The simulated result shows that the algorithm can reduce the probability of losing resource vehicles while the request is being processed.**

I. Introduction

Over the last two decades, mobile communication changed our lifestyle, which transmits more information quickly between more things. Meanwhile, the automotive industry is also developed by technological innovations [4]. Vehicles have more sensors and stronger computing ability. Vehicular Cloud Computing which is controlled by the cloud uses vehicles as servers to provide computation resources and data. Vehicular networks are recognized as a significant component of intelligent transportation systems [8] and they support various mobile services. With the development of more advanced technologies and equipment, more applications and services come out(such as self-driving) that increase the demand for more computing and data with low latency.

Fog is a new layer between the cloud and the edge of the network. Vehicular Fog Computing (VFC) extends the Vehicular Cloud Computing (VCC) paradigm to the edge of the vehicular network. VFC using vehicles as nodes has closer computing resources which do not need to communicate with the cloud bringing out low latency. Due to the dispersed of vehicles, aggregating idle computing resources of the individual vehicles and the computing resources of Road Sides Units(RSU) can enhance the Quality Of Services(Qos) greatly [1].

Although VFC has a huge opportunity for computing resources, its heterogeneity and mobility of vehicular nodes are challenges. The challenges stem from the heterogeneity of VFC and the high mobility of vehicular nodes. As vehicles move at high speeds and frequently change location, ensuring continuity and quality of service becomes a significant challenge. Furthermore, the diversity in computation and storage resources among vehicles raises the question of how to efficiently manage and schedule these resources to meet the needs of different applications.

Some requests require not losing a single resource vehicle during processing time. If a resource vehicle leaves, it will cause the request to fail and need to send the request again. Due to the high mobility of VFC, the situation become more often than traditional resource allocation. To reduce the probability, in this article, a heuristic-driven reinforcement learning and gradient optimization (HRG)algorithm is proposed. The heuristic algorithm considers the matching of resource and request, the distance between controller(RSU) and resource vehicles, and the predicted stay time which is calculated by the position changing. The controller will sort the resource vehicles when receiving requests by the heuristic algorithm, and service the request with resource vehicles in sorted order. According to the result of request processing, the heuristic algorithm will be optimized. simulations are used to evaluate the effect of the proposed algorithm. To get the most generalized results possible, simulations use an actual map, different densities of vehicles, and different random vehicle trips.

II. Background

A. Vehicular Network

The vehicular network is an emerging network. In vehicular networks, the nodes are usually vehicles and equipment on roads, which have high mobility. VANETs also called Vehicular Ad Hoc Networks are a subclass of the MANETs also called Mobile Ad Hoc Networks. VANETs provide wireless connection in vehicles to vehicles (V2V); vehicles to infrastructure (V2I); mix V2I and V2V. In VANETs, they communicate by a variety of wireless communication technologies, such as short-range radio, cellular, and WiMax.

B. Vehicular Cloud Computing

Vehicular cloud computing (VCC) is a distributed computing paradigm which has the advantage of cloud computing and vehicular networks. In VCC, vehicles are mobile nodes which can provide computation resources and process data to the other vehicles, as well as centralized to the cloud.

C. Vehicular Edge Computing

Vehicular edge computing (VEC) uses edge devices, such as on-board units (OBU) in vehicles, or RSU, and other forms of edge infrastructure to process and store data, rather than relying on centralized remote cloud infrastructure. VEC extends cloud capabilities and services to the edge of the network for a wide range of applications.

D. Vehicular Fog Computing

Fog computing architecture (the fog) distributes the tasks from the distant central management system in the cloud to the intermediate nodes which contain computational resources, to reduce the latency caused by transmitting messages between the front-end IoT devices and the back-end cloud [13].

VFC emerged as a way to combine the benefits of fog computing and vehicular networks to provide services and applications. Vehicles covered the vehicular network as fog nodes while providing decentralized local resources. The main characteristics of the Fog are Low latency, Wide-spread geographical distribution, a large number of nodes and heterogeneity [9]. Compared with VCC, the computation resource is typically at the edge of the network such as OBU or RSU rather than the remote cloud, so VFC has shorter transport distances, which come with less latency. Besides, fog computing has lower costs on deployment compared with cloud computing [10].

III. Related Works

Similar to the collaboration of the fog nodes concept, there are some offloading techniques [11] [12]. They introduce models that partition the fog landscape into distinct fog clusters, each one encompassing a centralized fog control node along with other distributed fog units. Furthermore, every individual fog unit embodies a visualized fog node that's equipped with computational, networking, and storage capabilities which are essential for coordinating an assortment of terminal IoT devices. [11]Using Maximal Resource Utilization-Based Allocation and Task Priority-Based Resource Allocation based on the task process time, but did not consider the mobility of the fog node.

In [1], they consider vehicles as infrastructures for communication and computation and propose a novel system of VFC. Four scenarios about utilizing moving and parked vehicles were provided. Two of them is using moving vehicles as infrastructures and the others are using parking vehicles. Each type of vehicle can be used to communicate and compute.

Moving vehicles transmit information by moving and communicating with different vehicles by using VANETs. Slow-moving vehicles work with RSU as local cloudlets and RSU connect to remote clouds. This hybrid cloud can provide computation capability to vehicles.

Parked vehicles can serve as static backbones and service infrastructures to improve connectivity. Besides, parked vehicles are able to service large computation demands.

Chaogang Tang, Shixiong Xia, Qing Li, Wei Chen, and Weidong Fang [2] propose pooling the vehicles together to share their computing resources in a community. Meanwhile, with the development of VFC, vehicles have more choices to join communities at the same time, they provide a genetic algorithm-based strategy to optimize that process depending on how much benefit they can earn by joining the community.

Compared to other wireless networks, vehicular networks is a highly dynamic topology, short transmission time and so on. Pereira et al. [4] propose an allocation and management resource policy for vehicular cloud(NANCY) to decide if to allocate the available resource to the request, which is based on the mathematical method Multiple Attribute Decision.

Lee and Lee [5] suggest utilizing parked vehicles to minimize service latency and present a heuristic algorithm that combines with reinforcement learning to solve the solutions the formulation set from the problem of allocating the limited fog resources.

IV. System Model

In an intelligent urban area, many vehicles are moving on the road. Each Roadside Unit(RSU) as a controller in fog maintained a resource pool and allocated resources. RSU will broadcast messages periodically to update vehicle information. When vehicles get into the RSU range and get the messages, they will send a message to tell RSU to join the resource pool. While the vehicles are in the RSU range, they will keep updating their information at each certain time (t_{update}) including their coordination, request resources, available resources, and so on. There are three kinds of resources in the resource pool (application, bandwidth, memory). If RSU does not receive the update message from the vehicle in a certain time, the vehicle will be removed from the resource pool. When vehicles have a request to the pool, they will send a message to RSU with what and how many resources they need and how long they need. When RSU revises the request, it will allocate resources from the pool to it. In my scenario, each request can be served by more than one resource vehicle, and each resource and resource vehicle can be allocated to more than one request.

In reality, some time-sensitive requests do not allow disconnect with resource vehicles while the request is in processing, otherwise, it will cause the request to fail. The proposal of the article object to reduce the probability of this kind of failure.

V. Heuristic-driven resource allocation via Reinforcement Learning and Gradient Optimization

A. Heuristic-driven resource allocation

Resources allocation is an NP problem. Using a heuristic algorithm is higher time efficiency with a not-bad solution. There are three parts in the heuristic algorithm: using Q,D, and T of each resource vehicle to sort the resource vehicle pool.

The first part Q is sorted by resource. The quantity of the requested resources($R^{\text{a}}_{\text{request}}$) and the quantity of each available($R^{\text{a}}_{\text{serve}}$) resource will be compared. The same one will have the lowest scores. $R^{\text{a}}_{\text{request}}$ bigger than $R^{\text{a}}_{\text{serve}}$ will have higher scores than $R^{\text{a}}_{\text{request}}$ smaller than $R^{\text{a}}_{\text{serve}}$. The smaller differences will have lower scores. The purpose of this section is This part aims to use as less as possible vehicles to serve the requests. Each resource will have a score . Fewer service vehicles indicated less probability of losing service vehicles.Q is the sum of all scores of resources in a vehicle(1).

$$Q = \sum_{1}^{\text{a}} Q_{\text{a}} \tag{1}$$

The second part D is sorted by distance. The distance between the resource vehicle and RSU can be calculated by their coordinate(2).

$$D = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{2}$$

The third part T is sorted by the predicted stay time of the resource vehicle. Assume the vehicle moves in a straight line. It updates its coordinate twice in t_{update}. Then RSU can calculate the slope and intercept of the equation of the line given the two points $(x1, y1)$ and $(x2, y2)$. Substitute the equation of the line into the equation of the circle(RSU coordinate$(x0, y0)$, radius(r)) to obtain a quadratic equation. Solve this equation to find the intersection($(x_\alpha, y_\alpha), (x_\beta, y_\beta)$) of the line and the circle(3). Use (2) to find $dist1$: $((x_\alpha, y_\alpha), (x1, y1))$, $dist2$: $((x_\alpha, y_\alpha), (x2, y2))$, $dist3$: $((x_\beta, y_\beta), (x1, y1))$, $dist4$: $((x_\beta, y_\beta), (x2, y2))$. Using (4) (5) get T.

$$(\frac{(y2 - y1)}{(x2 - x1)} * x + (y1 - \frac{(y2 - y1)}{(x2 - x1)} * x1) - y0)^2 + (x - x0)^2 = r^2 \tag{3}$$

$$dist = \begin{cases} dist2, & \text{if } dist2 < dist1 \\ dist4, & \text{if } dist4 < dist2 \end{cases} \tag{4}$$

$$T = \frac{dist}{v} = \frac{dist}{\frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{t_{\text{update}}}} \tag{5}$$

The RSU will sort each part to get three sorted lists and the smallest score on the top. Q_0 , D_0, T_0 are the serial numbers of each sorted list. Now each resource vehicle has (Q_0 , D_0, T_0 $\in$amount of all resource vehicles) . RSU will use (6) get S. k1,k2,k3 is default 10. Finally, RSU will use S from each resource vehicle to sort the resource pool for the request and allocate from the smallest S resource vehicle. **Algorithm 1** shows the procedure of heuristic-driven resource allocation.

$$S = k1 * Q_0 + k2 * D_0 + k3 * T_0 \tag{6}$$

Algorithm 1 Heuristic Resource Allocation

```
1: $k_1, k_2, k_3 \leftarrow 10, 10, 10$
2: procedure HeuristicMethod($RSU\_data$, $vehicles$)
3:     for all $vehicle \in vehicles$ do
4:         $Q \leftarrow$ CalculateQ()
5:         $D \leftarrow$ CalculateDistance()
6:         $T \leftarrow$ CalculateTime()
7:         $vehicle.score \leftarrow k_1 \cdot Q + k_2 \cdot D + k_3 \cdot T$
8:     end for
9:     SortVehiclesByScore(vehicles)
10:    $allocations \leftarrow$ AllocateResources($vehicles$)
11:    return $allocations$
12: end procedure
```

B. Reinforcement Learning and Gradient Optimization

VFC runs in a constantly changing environment, so reinforcement learning methods are suitable to be used here. The heuristic method only can not suit that environment well. Moreover, changes in this environment are continuous in time and space, so the normal distribution is introduced to change the parameters continuously. Hence, Reinforcement Learning and Gradient Optimization are used to optimize the heuristic algorithm to minimize the failure rate λ_{failure} of allocation in my proposal. This RL is an online RL, the algorithm works by calculating the observed failure rate after every 100 resource allocations to adjust parameters.

Using Reinforcement Learning and Gradient Optimization to optimize the sorting function(6), where $k1$, $k2$, and $k3$ are weight parameters. $k1$ is set to 10 at the beginning and will not change in the algorithm. $k2$, and $k3$ are randomly generated by their normal distribution in each allocation. The parameters of the normal distribution (μ mean, σ standard deviations) are $(10, 0.5)$ at the beginning. σ will not change in the algorithm. μ will adjust in the algorithm in each 100 times allocations ($t_{\text{calculate}}$).

RSU will calculate the failure rate(λ_{failure}) when it allocates resources 100 times($t_{\text{calculate}}$). When a serving vehicle leaves the resource pool while the request is still in time t_{need}, the number of failures (N_{failure}) plus 1, N_{total} is total request in $t_{\text{calculate}}$. After each $t_{\text{calculate}}$, RSU have a $\lambda_{\text{failure}} = N_{\text{failure}} \div N_{\text{total}}$, and then reset N_{failure} and N_{total} to zero.

The algorithm operates iteratively, adjusting μ of normal distributions based on the observed λ_{failure}. Before the iteration, μ_{current} is 10. The algorithm allocates 100 times and gets $\lambda^{\text{old}}_{\text{failure}}, \mu_{\text{oldCaculate}}$ to begin the iteration. At the beginning of the interaction, the algorithm allocates 100 times to get $\lambda^{\text{new}}_{\text{failure}}, \mu_{\text{tcalculate}}$. After that the λ_{failure} is observed. Depending on the change in λ_{failure} between iterations, the means of the normal distributions are updated. If λ_{failure} decreases from $\lambda^{\text{old}}_{\text{failure}}$ to $\lambda^{\text{new}}_{\text{failure}}$ (indicating performance improvement), the mean of each normal distribution changes like (7). If λ_{failure} increases $\lambda^{\text{old}}_{\text{failure}}$ to $\lambda^{\text{new}}_{\text{failure}}$ (indicating performance decline), the mean of each normal distribution changes like (8). Then set $\lambda^{\text{new}}_{\text{failure}}$ to $\lambda^{\text{old}}_{\text{failure}}$ and $\mu_{\text{tcalculate}}$ to $\mu_{\text{oldCaculate}}$ and μ_{new} to μ_{current}.

$\mu_{\text{tcalculate}}$ are the means of the corresponding k in last periods. $\mu_{\text{oldCaculate}}$ are the means of the corresponding k in the period before last periods. ω is the update rate which is a constant. **Algorithm 2** shows the procedure of Reinforcement Learning and Gradient Optimization for resource allocation.

$$\mu_{\text{new}} = \mu_{\text{current}} + \omega * (\lambda^{\text{old}}_{\text{failure}} - \lambda^{\text{new}}_{\text{failure}}) * (\mu_{\text{tcalculate}} - \mu_{\text{oldCaculate}}) \tag{7}$$

$$\mu_{\text{new}} = \mu_{\text{current}} - \omega * (\lambda^{\text{new}}_{\text{failure}} - \lambda^{\text{old}}_{\text{failure}}) * (\mu_{\text{tcalculate}} - \mu_{\text{oldCaculate}}) \tag{8}$$

Algorithm 2 Reinforcement Learning and Gradient Optimization for Resource Allocation

```
1:  Initialize $k1 = 10$, $\sigma_{k2} = \sigma_{k3} = 0.5$, $\omega = 0.3$
2:  $\mu_{\text{current}} = 10$                                   ▷ Before iteration starts
3:  Allocate resources 100 times to get initial $\lambda^{\text{old}}_{\text{failure}}$ and $\mu_{\text{oldCaculate}}$
4:  for each iteration do
5:      Allocate resources 100 times to calculate $\lambda^{\text{new}}_{\text{failure}}$, $\mu_{\text{tcalculate}}$
6:      if $\lambda^{\text{new}}_{\text{failure}} < \lambda^{\text{old}}_{\text{failure}}$ then       ▷ Performance improvement
7:          Eq(7)
8:      else if $\lambda^{\text{new}}_{\text{failure}} > \lambda^{\text{old}}_{\text{failure}}$ then       ▷ Performance decline
9:          Eq(8)
10:     end if
11:     $\lambda^{\text{old}}_{\text{failure}} \leftarrow \lambda^{\text{new}}_{\text{failure}}$
12:     $\mu_{\text{oldCaculate}} \leftarrow \mu_{\text{tcalculate}}$
13:     $\mu_{\text{current}} \leftarrow \mu_{\text{new}}$
14: end for
```

VI. PERFORMANCE EVALUATION

The approach: Heuristic-Driven Resource Allocation via Reinforcement Learning and Gradient Optimization was evaluated by simulation. For the simulator tool, we used VEINS (5.0) [14], SUMO (1.2.0) [15], and OMNeT++ (5.5.1) [16]. The map is an urban area map from Erlangen, Germany. The map area size is 3500*3500 m^2. Fig. 1 shows the map and the position of RSU in the simulation. The circle is the RSU communication range. The vehicle's movement was produced by pseudo-random parameters, which have random trips on the map. TABLE 1 shows the parameters setting in the simulation and evaluation.

A. Simulation Scenario and Methodology

RSU will send a message to all vehicles in fog in each $1s$ (t_{update}). When the vehicles receive the message, they will send back a message to RSU. This message has coordinates, available resources, and requested resources with request time. When the requested resources are not zero, the available resources are zero. It is 50% the request resources is not zero. When RSU receives this message, it will keep the vehicle in the resource pool, and add available resources to the pool or process the request. If RSU does not receive the message

TABLE I
SIMULATION PARAMETERS

Parameter	value
RSU Density	1
RSU Coordinate	RSU(1250,1350,3)
Vehicle Communication Range	500 m
RSU Communication Range	500 m
Vehicle Speed	0-120 km/h
Traffic Volume(period)	4,4.5,5 s/car arrival
Max hop count	1
RSU PHY model	IEEE 802.11p
Communication Method	V2I
Simulation time	30000s, 5000s
Request Resource	random(0,100)
Available Resource	random(0,100)
Resource Request time	random(0,60)s
$t_{\text{calculate}}$	100 times allocation
t_{update}	1 s
ω Update Rate	0.3

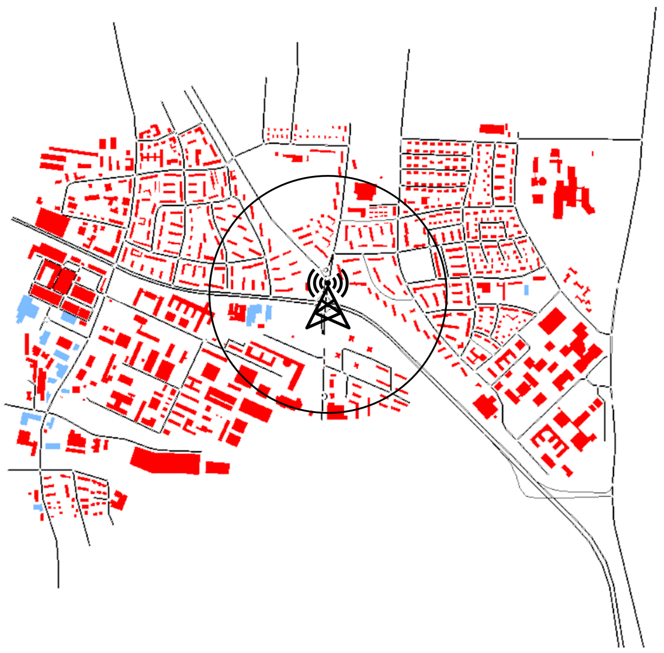

Fig. 1.

from the vehicle in 3 s, the vehicle will be deleted from the resource pool. When RSU processes the request, if it uses a heuristic method via Reinforcement learning and gradient optimization(HRG), it will sort the resource pool and then allocate resources to the request. If RSU does not use the HRG algorithm, it will allocate the available resources in order of last come first served(STACK). RSU will calculate the λ_{failure} in each $t_{\text{calculate}}$.

For simulation, the vehicle movement is generated by the tool of SUMO: randomTrips.py. Three traffic Volume situations are used in our scenario. Traffic Volume is defined by the period parameter. In our scenario, It means vehicles are generated every 4, 4.5, and 5 seconds. There are two parts simulation. For experiment 1, I applied STACK, HRG without reinforcement learning gradient optimization(H), and HRG in those three traffic volumes and got their λ_{failure}. Each simulation time is 5000 s.

For experiment 2, to observe the effection of the algorithm 2. I chose traffic volumes of 5 to run 6 times by HRG and H got their λ_{failure}. Each simulation time is 30000 s.

B. Performance Metrics

The proposal aims to reduce the probability of losing resource vehicles while the request is being processed, so λ_{failure} is used to evaluate.

C. Computational Complexity

For **Algorithm 1**: Heuristic algorithm, the main part is to rank vehicles by calculating different scores. The time complexity of the sorting operation itself is $O(nlogn)$, which is also the complexity of Algorithm 1.

For **Algorithm 2**: Reinforcement Learning and Gradient Optimization, the main part is to calculate the mean and standard deviation. It depends on how many k values in t_{caculate}. Therefore, it is $O(n)$.

D. Results

Fig.2 to Fig.4 are the simulation results of three traffic volumes: 4, 4.5, and 5. Their simulation time is all $5000s$. Y-axis is λ_{failure}, and X-axis show which interval of λ_{failure} in order of time. Each interval is t_{update} long. Each graph has three lines, they are λ_{failure} changing of STACK, H, and HRG in simulation time. STACK allocates available resources in order of last come first served without Algorithm 1 and Algorithm 2. H is only Algorithm 1. HRG are both Algorithm 1 and Algorithm 2.Fig.2 shows traffic volumes of 4, Fig.3 shows traffic volumes of 4.5, and Fig.4 shows traffic volumes of 5. In average λ_{failure} of traffic volumes: 4, HRG is 36.74% less than STACK. In average λ_{failure} of traffic volumes: 4.5, HRG is 31.24% less than STACK. In average λ_{failure} of traffic volumes: 5, HRG is 32.55% less than STACK. HRG and H all have similar λ_{failure}.

Fig.5 shows the average λ_{failure} of HRG and H in 6 times simulations. Each time of simulation is 30000s. The average λ_{failure} of HGR in 6 times simulations is 24.331%. H is 24.494% which is 0.163% more than HGR.

VII. CONCLUSION

This article proposes a heuristic-driven resource allocation via reinforcement learning and gradient optimization (HRG) approach. It has two parts: Algorithm 1 and Algorithm 2. According to the simulation result, HRG and H can reduce the λ_{failure} from STACK. The reinforcement learning and gradient optimization part of HRG works better when the running time is longer.

REFERENCES

[1] Hou, Xueshi, et al. "Vehicular fog computing: A viewpoint of vehicles as the infrastructures." IEEE Transactions on Vehicular Technology 65.6 (2016): 3860-3873.

[2] Tang, Chaogang, et al. "Resource pooling in vehicular fog computing." Journal of Cloud Computing 10.1 (2021): 1-14.

[3] YYi, Shanhe, Cheng Li, and Qun Li. "A survey of fog computing: concepts, applications and issues." Proceedings of the 2015 workshop on mobile big data. 2015.

[4] Pereira, Rickson S., et al. "A novel fog-based resource allocation policy for vehicular clouds in the highway environment." 2019 IEEE Latin-American Conference on Communications (LATINCOM). Ieee, 2019.

[5] Lee, Seung-seob, and SuKyoung Lee. "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information." IEEE Internet of Things Journal 7.10 (2020): 10450-10464.

[6] Yi, Sui, Li Yuhe, and Wang Yu. "Cloud computing architecture design of database resource pool based on cloud computing." 2018 International Conference on Information Systems and Computer Aided Education (ICISCAE). IEEE, 2018.

[7] Whaiduzzaman, Md, et al. "A survey on vehicular cloud computing." Journal of Network and Computer applications 40 (2014): 325-344.

[8] Khabazian, Mehdi, Sonia Aissa, and Mustafa Mehmet-Ali. "Performance modeling of message dissemination in vehicular ad hoc networks with priority." IEEE Journal on Selected Areas in Communications 29.1 (2010): 61-71.

[9] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012.

[10] Whaiduzzaman, Md, et al. "A survey on vehicular cloud computing." Journal of Network and Computer applications 40 (2014): 325-344.

[11] H. Tran-Dang and D. -S. Kim, "FRATO: Fog Resource Based Adaptive Task Offloading for Delay-Minimizing IoT Service Provisioning," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 10, pp. 2491-2508, 1 Oct. 2021, doi: 10.1109/TPDS.2021.3067654.

[12] O. Skarlat, M. Nardelli, S. Schulte and S. Dustdar, "Towards QoS-Aware Fog Service Placement," 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 2017, pp. 89-96, doi: 10.1109/ICFEC.2017.12.

[13] A. Y. Zomaya, A. Abbas, and S. U. Khan, "Fog Computing: Theory and Practice." Hoboken, NJ: John Wiley Sons, Inc., 2020.

[14] C. S. R. German. and F. Dressler., "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," IEEE Transactions on Mobile Computing, vol. 10, no. 1, p. 3–15, 2011.

[15] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotter ¨ od, ¨R. Hilbrich, L. Luck ¨ en, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using sumo," in Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2575–2582.

[16] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in Proceedings of the IEEE International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops (Simutools), 2008, p. 60:1–60:10.

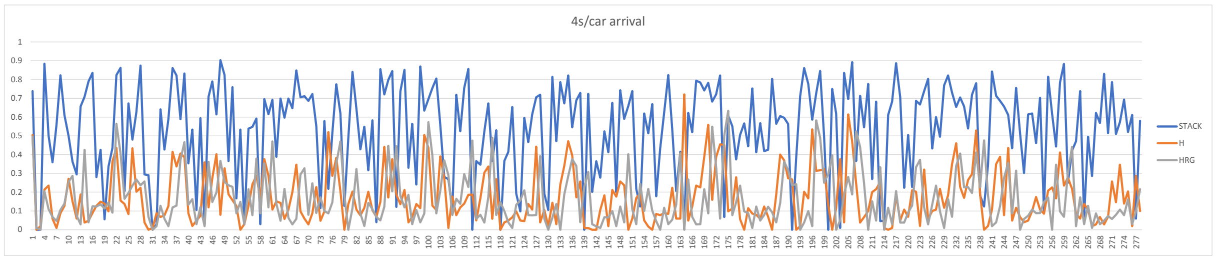

Fig. 2.

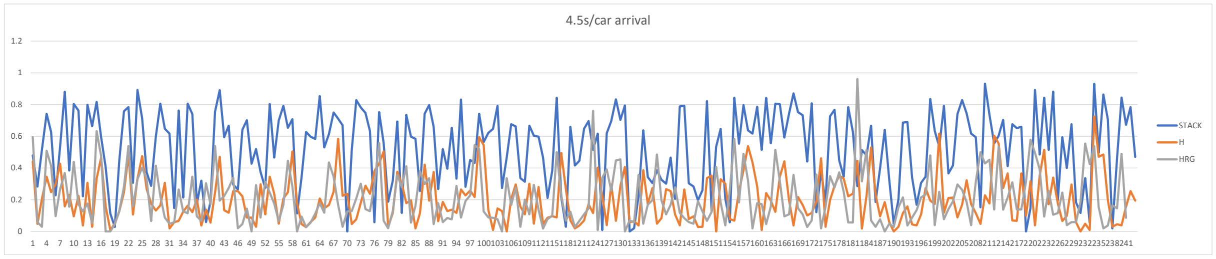

Fig. 3.

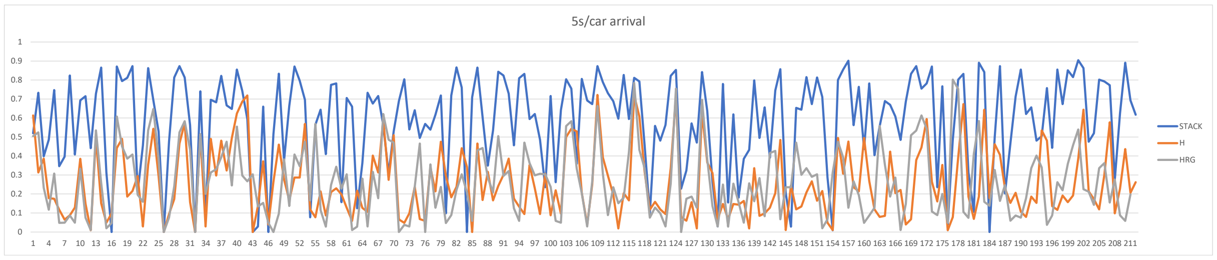

Fig. 4.

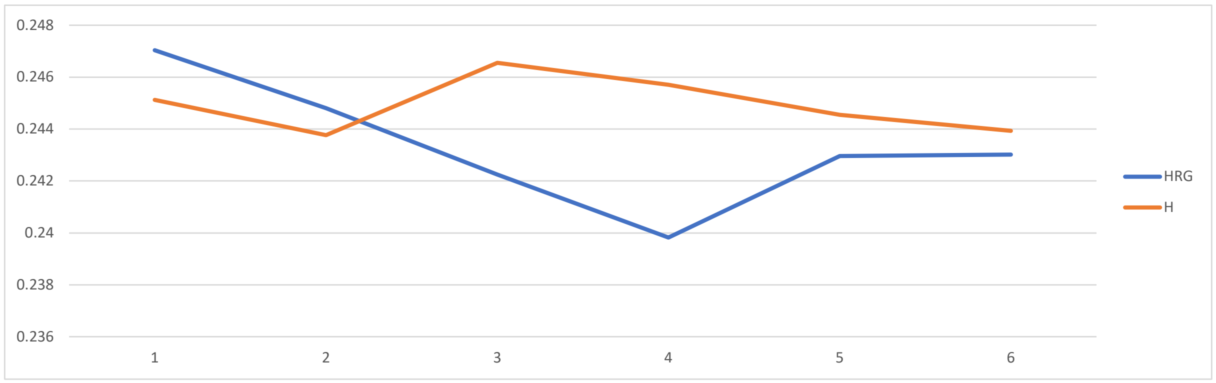

Fig. 5.