Optimized Assignment of Computational Tasks in Vehicular Micro Clouds

Ghaith Hattab
University of California
Los Angeles, California, USA
ghattab@ucla.edu

Seyhan Ucar
Toyota InfoTechnology Center
Mountain View, California, USA
seyhan@us.toyota-itc.com

Takamasa Higuchi
Toyota InfoTechnology Center
Mountain View, California, USA
ta-higuchi@us.toyota-itc.com

Onur Altintas
Toyota InfoTechnology Center
Mountain View, California, USA
onur@us.toyota-itc.com

Falko Dressler
Paderborn University
Paderborn, Germany
dressler@ccs-labs.org

Danijela Cabric
University of California
Los Angeles, California, USA
danijela@ee.ucla.edu

ABSTRACT

The ever-increasing advancements of vehicles have not only made them mobile devices with Internet connectivity, but also have pushed vehicles to become powerful computing resources. To this end, a cluster of vehicles can form a vehicular micro cloud, creating a virtual edge server and providing the computational resources needed for edge-based services. In this paper, we study the assignment of computational tasks among micro cloud vehicles of different computing resources. In particular, we formulate a bottleneck assignment problem, where the objective is to minimize the completion time of tasks assigned to available vehicles in the micro cloud. A two-stage algorithm, with polynomial-time complexity, is proposed to solve the problem. We use Monte Carlo simulations to validate the effectiveness of the proposed algorithm in two micro cloud scenarios: a parking structure and an intersection in Manhattan grid. It is shown that the algorithm significantly outperforms random assignment in completion time. For example, compared to the proposed algorithm, the completion time is 3.6x longer with random assignment when the number of cars is large, and it is 2.1x longer when the tasks have more varying requirements.

CCS CONCEPTS

• Information systems → Data management systems; **• Computer systems organization → Architectures**; **• Networks**;

KEYWORDS

Edge computing, task allocation, vehicular clouds, virtual edge, vehicular networks.

1 INTRODUCTION

The ever-increasing capabilities of vehicles have made their use transcend transportation. Indeed, cars are becoming infotainment systems with Internet connectivity, sensing capabilities, and powerful computing and storage capacities. For these reasons, vehicular networks are not only recognized for enabling intelligent transportation systems, e.g., enhanced safety and traffic management, but they are also envisioned to be integral for a myriad of emerging domains including smart cities [1], content-sharing applications [13], and autonomous driving [20].

To address the needs of sharing, storing, and processing a massive amount of the data expected from the aforementioned applications, fog, or edge, computing has been proposed to handle the data near its source, i.e., the end user [3]. Fog computing has traditionally relied on deploying edge servers, e.g., WiFi access points, but more recently, emphasis is put on the use of cars themselves as fog nodes because they are ubiquitous with ever-increasing computational resources. Indeed, the concept of *vehicle cloudification* has been discussed in [13], where a cloud leader, i.e., a car that runs an application, recruits other cars that can share their resources, forming a cloud to run the application. Alternatively, a large-scale vehicular cloud can be formed to cover a city [1] or to connect multiple cities [8], where the cloud members, i.e., cars, provide different services to end users, e.g., a car offering its unused CPU power for augmented reality applications on the road. Instead of forming a single *macro* cloud as in [1], vehicles can be clustered into several *micro* vehicular clouds, where each one acts as a virtual edge server, offering services ranging from memory storage, CPU power, and sensing [10]. For example, a micro cloud at an intersection can be used as a regional storage unit, where the objective is to maintain data contents in the vehicular cloud members' on-board memory storage. Another scenario is to have regional popular or relevant content processed and cached in these vehicles so that users in the area can access them any time, without the need to communicate directly with say a cellular network infrastructure.

The aforementioned works have focused on the feasibility of vehicular clouds for edge computing, highlighting the design principles and communication challenges. Recent works have addressed other aspects of vehicular clouds such as cloud formation and clustering [6, 7, 9], security and privacy [12, 15], and data management [16]. In this paper, we focus on a different component, and particularly, we study the allocation of computational tasks across the available vehicles in the micro cloud. Task allocation in fog or edge computing architectures has been extensively studied in the literature. For example, the problem can be cast as an integer

EdgeSys '19, March 25, 2019, Dresden, Germany
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6275-7/19/03…$15.00
https://doi.org/10.1145/3301418.3313937

program [11] or a mixed-integer program [5], where the objective is to find an assignment between users and fog nodes that maximizes a cost efficiency function as in [11], or to minimize the total cost of communication and deployment of edge servers as in [5]. Other formulations have centered around optimizing the energy-efficiency of fog nodes [4], minimizing service delay [17, 19], etc. These works, however, assume anchored edge computing resources, and thus the dynamics inherent in vehicular micro clouds are absent.

More recent work have studied task allocation in vehicular clouds. For example, the authors in [14] formulate a multiple-choice Knapsack problem, where the objective is to minimize the total cost of assigning tasks across multiple cars, each with different computational power and cost. However, it is assumed that the problem is solved offline, where future and past cloud dynamics are available beforehand. In this paper, we only use instantaneous information about the cloud, and implement a real-time algorithm. In [21], the authors formulate a multi-objective optimization to minimize service latency and quality loss, which is shown to be NP-hard. In our work, we cast the task allocation as a bottleneck assignment that minimizes completion time, which is solved using a low-complexity algorithm. In [18], the authors study task allocation for autonomous driving using a market-based approach. The work assumes that cars can adjust their speeds to remain in coverage with the end user, and thus tasks are not interrupted. In our work, a car may leave the cloud before the task is completed, and hence it needs to be rerun on a different car. We use traces from Manhattan grid to better capture cloud dynamics.

To summarize, we formulate a bottleneck assignment problem, where the objective is to assign computational tasks to cars, with various capabilities, so that the time it takes to complete all tasks is minimized. While we focus on stationary micro clouds, we consider both a parking structure scenario, where the cloud does not change, and an intersection scenario, where the cloud membership size can vary rapidly over time. The assignment scheduler, a central controller, is agnostic to cloud dynamics, and thus, we develop a two-stage low-complexity algorithm. In particular, the first stage sorts the tasks so that those with the highest likelihood to be completed without interruptions are passed to the cloud. This stage reduces the number of tasks passed to the assignment solver, making it equal to the number of available cars. In the second stage, we solve the bottleneck assignment, which is an integer program, via a sequence of linear programs, each of small size. The two-stage algorithm is implemented periodically to accommodate cloud dynamics, reallocate interrupted tasks, and assign new tasks arriving to the system. Using Monte Carlo simulations, we show that the proposed algorithm significantly reduces the completion time of all tasks in comparison to random assignment, particularly when the cloud size is large, and when the tasks themselves have various requirements. We also show that the presence of the sorting stage limits the queuing time of each task in the system.

2 SYSTEM MODEL

We assume that time is discretized into slots, $i = 0, 1, \cdots$, each is of duration t. In this work, we focus on stationary micro clouds, i.e., the cloud is anchored at a fixed geographical region. For example, cars in a parking structure can form a micro cloud, or any cars in a particular intersection can be part of a micro cloud.

Let M_i denote the number of cars in the vehicular micro cloud during the i-th slot. Then in the parking structure scenario, we assume $M_i = M \forall i$, i.e., the cloud dynamics are much slower than the tasks' dynamics, and hence the number of cars during task allocation and processing remains the same. In the intersection scenario, M_i changes over different slots. Thus, the cloud membership size, i.e., the number of vehicles, can tangibly vary over time, elevating the need for fast task allocation.

For computational tasks, we assume there are N_0 initial tasks. Then, additional streaming tasks start arriving to the system using the exponential arrival process, with an arrival rate of λ tasks per slot. In the simulations, we bound the total number of tasks in the system by $N_{\max}$. Note that if $N_0 = N_{\max}$, then the model becomes batch processing of tasks, i.e., tasks do not follow an arrival process but rather they are all available at once.

We consider car resources to be its CPU computational power. Specifically, each m-th car can process P_m CPU cycles per time slot, where we assume P_m is generated from an exponential distribution with mean $\bar{P}$ CPU cycles/slot. Similarly, we assume each n-th task requires τ_n CPU cycles, which is generated from another exponential distribution with mean $\bar{\tau}$. In this work, we only consider non-preemptive tasks, i.e., if a car running a task leaves the micro cloud before it finishes it, then the task needs to be restarted. We note that the proposed algorithm is developed irrespective of the commodity used as car resources, and irrespective of the distributions used to generate P_m and τ_n.

Our objective is to allocate the remaining tasks in the system to the available cars in the micro cloud, under the constraint that each car is assigned one task at a time, and each task is assigned to one car. It is straightforward to include the case of assigning a task to multiple cars, which improves robustness to cloud dynamics. In this work, we use a central controller to perform the task assignment. In particular, when a car joins the micro cloud, it informs the controller about its computational power. The controller is also assumed to know if a car is no longer part of the cloud, e.g., if the controller loses connection to the car or if the car sends a beacon when it leaves the cloud. Thus, the controller has knowledge about the *current* cars in the micro cloud, their processing capabilities, and the requirements of the remaining tasks in the system. The controller, however, does not know the future dynamics of the micro cloud. We emphasize that the notion of a *central controller* does not have to be a part of an infrastructure, e.g., edge server. Rather, we refer to the controller as any entity that has knowledge about the tasks and the current status of the cloud, e.g., a vehicle with such knowledge in the micro cloud can do the task allocation. An illustrative example of the system model is provided in Figure 1. After task assignment, the micro cloud can use one of the existing data transfer protocols to send instructions and data between vehicles and the controller [16].

3 PROPOSED TASK ALLOCATION ALGORITHM

In this section, we discuss how the task allocation problem can be cast as an assignment problem. We then develop a polynomial-time complexity algorithm to solve the problem.

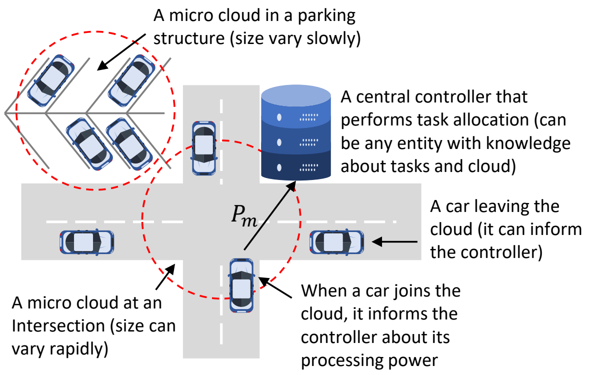

A micro cloud in a parking structure (size vary slowly)
A central controller that performs task allocation (can be any entity with knowledge about tasks and cloud)
P_m
A car leaving the cloud (it can inform the controller)
A micro cloud at an Intersection (size can vary rapidly)
When a car joins the cloud, it informs the controller about its processing power

Figure 1: An illustrative example of vehicular micro clouds.

3.1 The bottleneck assignment problem

We aim to find an assignment that minimizes the completion time of the available computational tasks. To this end, we define the cost of assigning an n-th task at an m-th car starting from the i-th time slot as

$$c_{nm,i} = \beta^{\star}_{nm,i} t, \tag{1}$$

where

$$\beta^{\star}_{nm,i} = \underset{j}{\text{argmin}} \left\{ \tau_n \leq \sum_{l=i}^{j-1} P_m \right\}. \tag{2}$$

Note that we are assuming here that P_m is fixed across time slots, yet the formulation in (2) accommodates the case where the CPU power of the car varies over time (in this case we would use $P_{m,l}$ instead of P_m).

Let us first focus on the initial assignment phase, where we have N_0 tasks and M_0 cars. Then, the bottleneck assignment is formulated as follows

$$\begin{array}{ll} \underset{\{x_{nm,0}\}}{\text{minimize}} & \underset{(n,m)}{\max}\; c_{nm,0} x_{nm,0} \\ \text{subject to} & \sum_{n=1}^{N_0} x_{nm,0} = 1,\; \forall m = 1, 2, \cdots, M_0 \\ & \sum_{m=1}^{M_0} x_{nm,0} = 1,\; \forall n = 1, 2, \cdots, N_0 \\ & x_{nm,0} \in \{0, 1\}, \end{array} \tag{3}$$

where the first constraint ensures that each car is assigned one computational task, the second constraint ensures that each task is assigned to one car, and the third constraint emphasizes that the optimizing variables are binary, i.e., $x_{nm,0} = 1$ if the n-th task is assigned to the m-th car, and $x_{nm,0} = 0$ otherwise. Since there are M_0 available cars, then only $M_0 \leq N_0$ tasks are selected. Thus, the problem in (3) finds an assignment that minimizes the maximum time it takes to finish all M_0 tasks.

The initial phase is followed by multiple assignment updates. Such updates are necessary because cars' availability vary over time, e.g., a cloud member becomes available after it finishes its assigned task, a new car joins the cloud, or an existing car leaves it. Similarly, some tasks are restarted if their assigned cars have left the cloud before task completion, while other tasks have not yet been assigned to any car or have just arrived to the system.

The update phase can be done periodically, say every T seconds, or by triggering it whenever the number of available cars in the cloud exceeds some threshold. For the assignment update during the i-th slot, we have the following problem to solve

$$\begin{array}{ll} \underset{\{x_{n'm',i}\}}{\text{minimize}} & \underset{(n',m')}{\max}\; c_{n'm',i} x_{n'm',i} \\ \text{subject to} & x_{n'm',i} \geq x^{\star}_{n'm',i-1} \end{array} \tag{4}$$

where we still enforce the same constraints as (3). Here, we use different subscripts n' and m' to emphasize that indices can change over time due to cars' and tasks' dynamics. The additional constraint in (4) implies that an m'-th car remains assigned to its task if it has not yet completed it by the i-th slot.

As we will discuss in the next section, we will rely on casting this problem as a sequence of linear assignment problems, i.e., we minimize a sum term instead of a max term and relax the binary constraints for $x_{nm,i}$. By doing so, we can use existing polynomial-time algorithms to solve the problem. The issue is that under such formulation, the problem must have the number of tasks in the system equal to the number of available cars. This is commonly handled by augmenting the problem with virtual cars. For example, if the number of cars, say M_i, is less than the number of current tasks, say N_i, then we add $N_i - M_i$ virtual cars, and we assign the cost $c_{n\tilde{m},i} = 0$ for every virtual $\tilde{m}$-th car. Then, we solve (4), and the tasks assigned to the virtual cars are dropped. The same approach can be used if the number of cars is larger than the number of tasks. In this paper, we avoid augmentation of cars, to keep the problem instance small. To this end, we only optimize over the set of available cars and a set of tasks, with cardinality equal to the number of available cars. This is discussed in details in the next section.

3.2 A two-stage polynomial-time algorithm

We break down every assignment phase into two stages: a sorting stage and an assignment one. In what follows, we denote the set of available cars and the set of remaining tasks during the i-th slot by $\mathcal{M}_i$ and $\mathcal{N}_i$, respectively. We assume $|\mathcal{M}_i| \leq |\mathcal{N}_i|$.

Let $c^{\max}_{n,i} = \max_{m \in \mathcal{M}_i} c_{nm,i}$ be the maximum time it takes the task to finish if it is assigned to any available car in the micro cloud. Let $q_{n,i}$ be the time this task has been in the system queue. We propose first to sort the tasks in the system according to $\alpha_{n,i} = c^{\max}_{n,i}/q_{n,i}$. Then, we select $|\mathcal{M}_i|$ tasks with the lowest $\alpha_{n,i}$. The rationale behind this sorting is as follows. If all tasks arrive at the same time, i.e., $q_{n,i} = q \forall i$, then we start assigning the tasks that require the least amount of time, particularly because of the uncertainty in car dynamics. By doing so, we reduce tasks' interruptions, as we assign those that have the highest likelihood to be completed before their assigned cars leave the cloud. If tasks arrive to the system at different times, then $\alpha_{n,i}$ takes into account the time the task has been in the queue, giving more priority to those that have been in the system for a very long duration. In other words, $\alpha_{n,i}$ helps pass the tasks that are unlikely to be interrupted, while still being fair to those tasks that have been in the system for a very long time.

Once $|\mathcal{M}_i|$ tasks are selected from $\mathcal{N}_i$, we use the threshold algorithm to optimally solve the bottleneck assignment problem [2], where now we have $|\mathcal{M}_i|^2$ optimizing variables instead of $|\mathcal{N}_i|^2$. The idea is as follows. The objective function is equal to one of the entries of the cost matrix: $\mathbf{C}_i \in \mathbb{R}^{|\mathcal{M}_i| \times |\mathcal{M}_i|}$, where $[\mathbf{C}_i]_{nm} = c_{nm,i}$.

Algorithm 1 A two-stage assignment algorithm during the *i*-th slot

```
1: procedure INPUT($\mathcal{N}_i$, $\mathcal{M}_i$, $\{c_{nm,i}\}$, $\{q_{n,i}\}$)
2:     Compute $\alpha_{n,i} = c^{\max}_{n,i}/q_{n,i} \forall n \in \mathcal{N}_i$
3:     Sort the tasks in increasing order according to $\alpha_{n,i}$
4:     Select $|\mathcal{M}_i|$ tasks with smallest $\alpha_{n,i}$
5:     Compute the cost threshold $\tilde{c}$ according to (5)
6:     Define the assignment costs $\tilde{c}_{nm,i} = \mathbf{1}(c_{nm,i} > \tilde{c}) \cdot c_{nm,i}$
7:     Initialize $\nu = 1$
8:     while ($\nu \neq 0$) do
9:         Solve the LP in (6) to get $x^{\star}_{nm,i}$
10:        Update $\nu = \sum_{n=1}^{|\mathcal{M}_i|} \sum_{m=1}^{|\mathcal{M}_i|} \tilde{c}_{nm,i} x^{\star}_{nm,i}$
11:        Update $\tilde{c} = \max\{\tilde{c}_{nm,i} x^{\star}_{nm,i}\}$
12:        Update $\tilde{c}_{nm,i} = \mathbf{1}(c_{nm,i} > \tilde{c}) \cdot c_{nm,i}$
13:    end while
14:    Return $x^{\star}_{nm,i} \forall n \in \mathcal{N}_i, m \in \mathcal{M}_i$
15: end procedure
```

In addition, because of the constraints in (3), every row and every column in the matrix will have a cost element that contributes to the cost of assignment. Thus, we can find a lower bound on the optimal objective function, which is

$$\tilde{c} = \max_{1 \leq k \leq |\mathcal{M}_i|} \{\min_n c_{nk,i}, \min_m c_{km,i}\} \tag{5}$$

Let $\tilde{c}_{nm,i} \triangleq [\tilde{\mathrm{C}}_i]_{nm} = \mathbf{1}(c_{nm,i} > \tilde{c}) \cdot c_{nm,i}$. Then, if we find an assignment with $\sum_{n=1}^{|\mathcal{M}_i|} \sum_{m=1}^{|\mathcal{M}_i|} \tilde{c}_{nm,i} x^{\star}_{nm,i} = 0$, we know this assignment is optimal. Otherwise, we increase the threshold to $\tilde{c} = \max\{\tilde{c}_{nm,i} x^{\star}_{nm,i}\}$, and repeat the process until the aforementioned linear sum is zero.

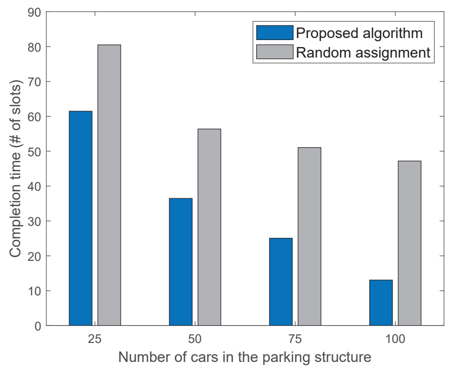
The main computations of this algorithm is finding an assignment with zero sum. This is accomplished by solving the following linear program (LP)

$$\begin{aligned} \underset{\{x_{nm,i}\}}{\text{minimize}} \quad & \textstyle\sum_{n=1}^{|\mathcal{M}_i|} \sum_{m=1}^{|\mathcal{M}_i|} \tilde{c}_{nm,i} x_{nm,i} \\ \text{subject to} \quad & \textstyle\sum_{n=1}^{|\mathcal{M}_i|} x_{nm,i} = 1, \ \forall m = 1, 2, \cdots, |\mathcal{M}_i| \\ & \textstyle\sum_{m=1}^{|\mathcal{M}_i|} x_{nm,i} = 1, \ \forall n = 1, 2, \cdots, |\mathcal{M}_i| \\ & 0 \leq x_{nm,i} \leq 1, \end{aligned} \tag{6}$$

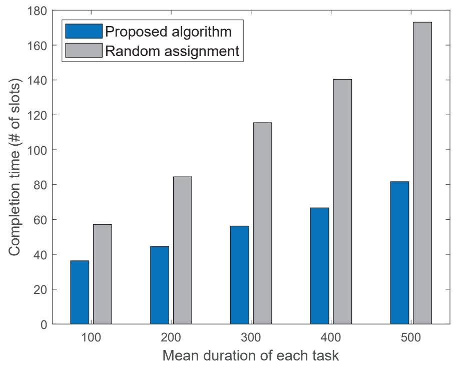
We note that this is a linear assignment problem, and although there is no explicit constraint on restricting the optimizing variables to be integral, the solution will always be integral [2]. Such problem can be solved in polynomial-time complexity using standard methods for linear programming or convex optimization. A summary of the proposed algorithm is provided in Algorithm 1. Finally, we note that the aforementioned algorithm assumes $|\mathcal{M}_i| \leq |\mathcal{N}_i|$. In the few instances where there are more available cars than tasks, we augment the system with virtual tasks as discussed earlier.

4 SIMULATION RESULTS

We evaluate the proposed algorithm with random assignment, where the controller randomly pairs tasks with available cars in the micro cloud. We run MATLAB-based simulations, where 100 realizations are generated with different car processing capabilities and tasks requirements. For the random assignment, we run 100 trials for each realization, each with different assignment, and we show the average performance across these trials. For both assignment schemes, we consider periodic assignment updates every 10 slots. Increasing assignment periodicity can further reduce completion time, yet at the expense of additional communication overhead.



(a) Completion time versus different number of cars



(b) Completion time versus duration of tasks

Figure 2: Completion time comparison for the parking structure scenario.

4.1 Scenario 1: Parking structure

In this scenario, we focus on batch processing, where the system has $N_0 = N_{\max}$ tasks that need to be processed by a large micro cloud in a parking structure. Unless otherwise stated, we assume the mean duration of tasks is $\bar{\tau} = 100$ cycles, while the mean CPU power of each car is $\bar{P} = 20$ cycles per slot.

We compare between the random and proposed assignment schemes in terms of the completion time, i.e., the time it takes to finish all tasks, averaged across realizations. In Figure 2a, we show the completion time for different cloud sizes, where $N_0 = 100$ tasks. We have two observations. First, as the number of cars increases, the completion time reduces because there will be a higher chance to have cars with powerful capabilities that can perform several tasks in a shorter time. Second, the proposed algorithm provides tangible reduction in completion time, particularly when the number of cars is larger, e.g., when there are 100 cars, the completion time is 3.6x longer with random assignment, showing that the proposed algorithm efficiently utilizes the cars' capabilities. In Figure

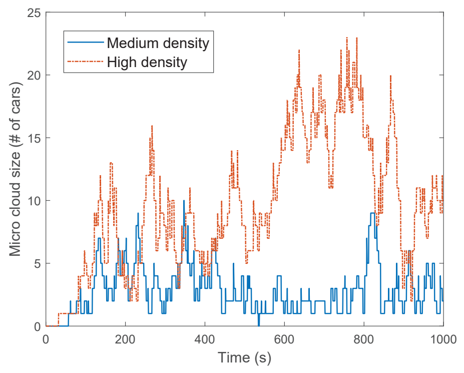


Figure 3: Micro cloud size over time at an intersection in Manhattan grid.

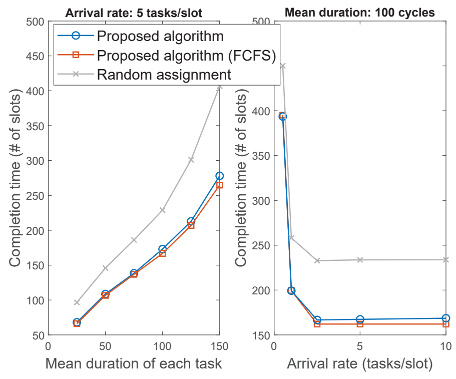


Figure 4: Completion time comparison for the intersection scenario.

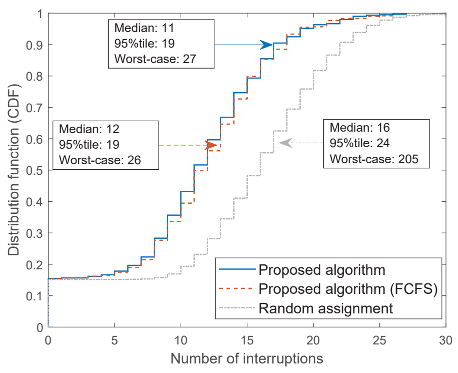


Figure 5: CDF of number of interruptions.

2b, we show the completion time for different mean task duration, $\bar{\tau}$. It is assumed that the number of cars is 50. Note that since the task duration is exponentially distributed, then increasing the mean also increases the variance, i.e., variations of computational requirements across tasks increase. It is shown that the proposed algorithm handles higher heterogeneity of tasks' durations compared to the random assignment. For example, the completion time is more than halved compared to random assignment when $\bar{\tau} = 500$.

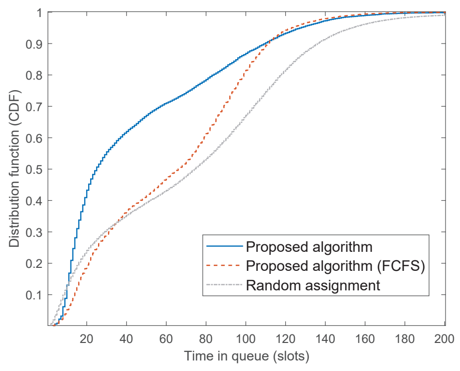
4.2 Scenario 2: Intersection in Manhattan grid

In this scenario, using Veins simulator, we generate cars in a Manhattan grid and we form a micro cloud at one of the intersections. To this end, we sample each car trace at 0.5s intervals and find when it becomes part of the cloud. In Figure 3, we show the number cars in the micro cloud at the intersection over time under two different densities. In what follows, we only show results for the high-density scenario as similar conclusions are obtained for other densities. We further assume tasks are streaming into the system with $N_0 = 20$ tasks and $N_{\max} = 200$ tasks.

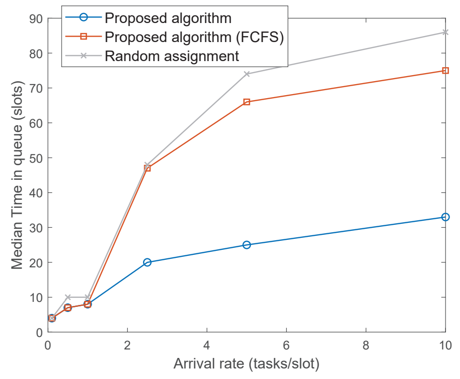
To study the impact of sorting, we also implement the proposed algorithm without sorting, where tasks are processed on a first come first served (FCFS) basis. In Figure 4, we show the completion time for different task mean duration and different arrival rates. It is evident that the proposed algorithms still outperform random assignment, even when the micro cloud has high dynamics. Note that for slow arrival rates, cars may remain idle waiting for tasks to arrive to the system, increasing the completion time. We also show in Figure 5 the cumulative distribution function (CDF) of the number of interruptions due to cars leaving the cloud before completing their tasks. It is evident that the proposed algorithm reduces the number of interruptions and limits the worst-case scenario compared to random assignment. We observe that the impact of sorting is negligible in terms of completion time. However, as evident in Figure 6, sorting significantly reduces the time a task spends in the queue. For example, in Figure 6a, we show the CDF for the time in queue when the arrival rate is 5 tasks/slot, whereas in Figure 6b we show the median wait time for different arrival rates. It is observed that sorting reduces the delay in sending the tasks to the cloud compared to the FCFS-based assignment. For example, as the arrival rate increases one order of magnitude, the median time in queue increases by 8x when sorting is implemented and increases by approximately 19x when it is absent. Such significant reduction in wait time does not come at the expense of the completion time performance.

5 CONCLUSION AND FUTURE WORK

Vehicular micro clouds can advance the fog computing architecture by creating virtual edge servers using a cluster of cars at parking structures and intersections. A key aspect in micro clouds is the assignment of computational tasks to cars of different computing resources. Because of the mobility of cars, the micro cloud is dynamic and can rapidly change over time, elevating the need for task allocation algorithms that can be solved in short time, i.e., with low complexity. In this paper, we cast the problem of task allocation as a bottleneck assignment that minimizes the completion time of tasks to be run on the micro cloud. In particular, we propose a two-stage algorithm that helps limit the computational complexity. In the first stage, we sort the tasks according to the ratio of task's completion time to its wait time, which helps reduce task interruptions while still being fair to tasks that have been long enough in the system. In the second stage, we select a subset of tasks that have the lowest



(a) CDF of the wait time in queue



(b) Median wait time versus arrival rate

Figure 6: Time in queue comparison for the intersection scenario.

ratio, and then solve a sequence of linear programs, typically with a small number of optimizing variables. Simulation results show that the proposed algorithm significantly reduces the completion time compared to the random assignment, particularly when the number of cars is large. Further, sorting tasks helps limit the time each one spends in the queue.

Ongoing and future work aim to extend on the system model and proposed algorithm. For example, the algorithm is blind to future dynamics, and hence assignments are broken down into periodic updates, each solved using only knowledge about the current cloud status. Prediction of cloud dynamics can be used as a side information to reduce the number of updates, e.g., at peak hours, cars are expected to spend more time at intersections. In addition, we only assume a centralized implementation, which is suitable for stationary clouds. However, for mobile clouds formed on a highway, distributed coordination may be more practical. Last, we aim to compare the algorithm with other assignment schemes, evaluating them using an end-to-end vehicular network simulator which also captures the impact of connectivity.

ACKNOWLEDGMENTS

The authors would like to thank Florian Hagenauer for providing car traces from Manhattan grid.

REFERENCES

[1] Onur Altintas, Falko Dressler, Florian Hagenauer, Makiko Matsumoto, Miguel Sepulcre, and Christoph Sommer. 2015. Making Cars a Main ICT Resource in Smart Cities. In *IEEE INFOCOM 2015, SmartCity Workshop*. IEEE, Hong Kong, China, 654–659. https://doi.org/10.1109/INFCOMW.2015.7179448

[2] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment Problems*. Society for Industrial and Applied Mathematics.

[3] Mung Chiang and Tao Zhang. 2016. Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal* 3, 6 (Dec. 2016), 854–864. https://doi.org/10.1109/JIOT.2016.2584538

[4] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. 2016. Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal* 3, 6 (Dec. 2016), 1171–1181. https://doi.org/10.1109/JIOT.2016.2565516

[5] Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, and Yong Xiang. 2017. Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System. *IEEE Transactions on Emerging Topics in Computing* 5, 1 (Jan. 2017), 108–119. https://doi.org/10.1109/TETC.2015.2508382

[6] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2017. Vehicular Micro Clouds as Virtual Edge Servers for Efficient Data Collection. In *ACM MobiCom 2017, CarSys Workshop*. ACM, Snowbird, UT, 31–35. https://doi.org/10.1145/3131944.3133937

[7] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2018. Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers. *Elsevier Ad Hoc Networks* 78 (Sept. 2018), 73–83. https://doi.org/10.1016/j.adhoc.2018.05.014

[8] Florian Hagenauer, Christoph Sommer, Ryokichi Onishi, Matthias Wilhelm, Falko Dressler, and Onur Altintas. 2016. Interconnecting Smart Cities by Vehicles: How feasible is it?. In *IEEE INFOCOM 2016, SmartCity Workshop*. IEEE, San Francisco, CA, 788–793. https://doi.org/10.1109/INFCOMW.2016.7562184

[9] Takamasa Higuchi, Falko Dressler, and Onur Altintas. 2018. How to Keep a Vehicular Micro Cloud Intact. In *IEEE VTC2018-Spring*. IEEE, Porto, Portugal. https://doi.org/10.1109/VTCSpring.2018.8417759

[10] Takamasa Higuchi, Joshua Joy, Falko Dressler, Mario Gerla, and Onur Altintas. 2017. On the Feasibility of Vehicular Micro Clouds. In *IEEE VNC 2017*. IEEE, Torino, Italy, 179–182. https://doi.org/10.1109/VNC.2017.8275621

[11] Boqi Jia, Honglin Hu, Yu Zeng, Tianheng Xu, and Yang Yang. 2018. Double-matching resource allocation strategy in fog computing networks based on cost efficiency. *Journal of Communications and Networks* 20, 3 (June 2018), 237–246. https://doi.org/10.1109/JCN.2018.000036

[12] Qinglei Kong, Rongxing Lu, Hui Zhu, and Maode Ma. 2018. Achieving Secure and Privacy-Preserving Incentive in Vehicular Cloud Advertisement Dissemination. *IEEE Access* 6 (2018), 25040–25050. https://doi.org/10.1109/ACCESS.2018.2827365

[13] Euisin Lee, Eun-Kyu Lee, Mario Gerla, and S.Y. Oh. 2014. Vehicular Cloud Networking: Architecture and Design Principles. *IEEE Communications Magazine* 52, 2 (Feb. 2014), 148–155. https://doi.org/10.1109/MCOM.2014.6736756

[14] Mahmudun Nabi, Robert Benkoczi, Sherin Abdelhamid, and Hossam S. Hassanein. 2017. Resource assignment in vehicular clouds. In *IEEE ICC 2017*. IEEE, Paris, France, 1–6. https://doi.org/10.1109/ICC.2017.7996577

[15] Jianbing Ni, Aiqing Zhang, Xiaodong Lin, and Xuemin Sherman Shen. 2017. Security, Privacy, and Fairness in Fog-Based Vehicular Crowdsensing. *IEEE Communications Magazine* 55, 6 (June 2017), 146–152. https://doi.org/10.1109/MCOM.2017.1600679

[16] Gurjashan Singh Pannu, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2018. Efficient Uplink from Vehicular Micro Cloud Solutions to Data Centers. In *IEEE WoWMoM 2018*. IEEE, Chania, Greece. https://doi.org/10.1109/WoWMoM.2018.8449787

[17] Yuvraj Sahni, Jiannong Cao, and Lei Yang. 2018. Data-Aware Task Allocation for Achieving Low Latency in Collaborative Edge Computing. *IEEE Internet of Things Journal* (2018), 1. https://doi.org/10.1109/JIOT.2018.2886757 available online.

[18] Z. Su, Y. Hui, and T. H. Luan. 2018. Distributed Task Allocation to Enable Collaborative Autonomous Driving With Network Softwarization. *IEEE Journal on Selected Areas in Communications* 36, 10 (Oct. 2018), 2175–2189. https://doi.org/10.1109/JSAC.2018.2869948

[19] Ashkan Yousefpour, Genya Ishigaki, Riti Gour, and Jason P. Jue. 2018. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet of Things Journal* 5, 2 (April 2018), 998–1010. https://doi.org/10.1109/JIOT.2017.2788802

[20] Kan Zheng, Qiang Zheng, Haojun Yang, Long Zhao, Lu Hou, and Periklis Chatzimisios. 2015. Reliable and efficient autonomous driving: the need for heterogeneous vehicular networks. *IEEE Communications Magazine* 53, 12 (Dec. 2015), 72–79. https://doi.org/10.1109/MCOM.2015.7355569

[21] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. 2018. Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing. *IEEE Internet of Things Journal* (2018). https://doi.org/10.1109/JIOT.2018.2875520 available online.