E-PODS: A Fast Heuristic for Data/Service Delivery in Vehicular Edge Computing

by

Akshaj Gupta, Joseph John Cherukara, Deepak Gangadharan, BaekGyu Kim, Oleg Sokolsky, Insup Lee

Report No: IIIT/TR/2021/-1

Centre for Others
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2021

2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) | 978-1-7281-8964-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/VTC2021-Spring51267.2021.9448649

E-PODS: A Fast Heuristic for Data/Service Delivery in Vehicular Edge Computing

Akshaj Gupta*, Joseph John Cherukara*, Deepak Gangadharan*, BaekGyu Kim†, Oleg Sokolsky‡ and Insup Lee‡
*IIIT Hyderabad, †Toyota Motor North America, ‡University of Pennsylvania
Email: *{akshaj.gupta, joseph.cherukara}@research.iiit.ac.in, *deepak.g@iiit.ac.in, †baekgyu.kim@toyota.com,
‡{sokolsky, lee}@cis.upenn.edu

***Abstract*—With the rise in state-of-the-art communication modes for vehicles such as vehicle to vehicle (V2V), vehicle to infrastructure (V2I) and vehicle to cloud (V2C), modern vehicles are increasingly being connected to cloud and fog/edge nodes. These vehicle connectivity modes have enabled the realization of *Vehicular Edge Computing* (VEC) paradigm, whereby vehicles can leverage fog/edge node resources for storage/computation. In a VEC system, vehicles receive very important and large quantity of data from edge nodes, which is termed as data delivery. In addition, edge nodes can execute some services and send the results back to the vehicle, which is called service delivery. Fast and efficient edge resource allocation for data/service delivery is important in order to serve as many vehicles as possible in the VEC system. However, edge resource allocation is complex with large number of edges and vehicles, while also considering vehicle flow parameters. In this work, we propose *Edge-Pairwise Optimal Data/Service Delivery (E-PODS)*, which is a fast and efficient heuristic for data/service delivery. Through experiments with synthetic and real vehicular traces, we demonstrate that E-PODS is considerably faster than the optimal approach, while making resource allocations that are close to optimal in terms of *total edge bandwidth cost* and *number of serviced vehicles*.**

***Keywords*-V2I; Vehicular Edge Computing, Resource Allocation, Road Side Units, Connected Vehicles, Bandwidth Cost**

I. Introduction

In the past few years, the automotive industry has focused its research and development on technologies that enable the realization of connected and automated vehicles. This effort has seen lots of innovation in the development of novel networking technologies and applications that enhance driver safety and experience, such as intelligent driving, driver behaviour analysis and predictive maintenance. Many of these applications either receive large amounts of data [1] from cloud or are computation intensive functions [2]. Therefore, storage and computation resources are very critical to the efficient execution of these applications. Although it is best to execute these applications on the vehicle platform to minimize latency, it may be computationally infeasible due to the limited capacity of storage and compute resources available as many concurrently running applications result in resource contention.

One solution to the computation problem has been to offload the execution of an application partially or entirely to the cloud server (known as mobile cloud computing (MCC) [3]) due to its massive computational capacity. The results of the computation are then sent back to the vehicle. However, this approach results in excessive turnaround time due to communication latency. Additionally, with more vehicles offloading their computation, there will be contention for communication bandwidth. The above two problems are addressed by the paradigm of Mobile Edge Computing (MEC) [4] and further by Vehicular Edge Computing (VEC) [2] in the context of vehicles. In VEC, there are edge nodes, which sit in between the cloud and the vehicles and are physically located closer to the vehicles. Any data (such as update data [5], [6]) that is sent from the cloud to a group of vehicles can be sent via the edge nodes, which will result in reduction of bandwidth usage as the same data need not be sent multiple times to different vehicles. The vehicles can then retrieve their data as they pass through the coverage area of the edge. Similarly, computation can be offloaded [2] from the vehicle to the edge instead of the cloud, which considerably reduces the turnaround time.

In the context of edge computing-based data/service delivery to the vehicles, one important question that needs to be answered is how to allocate resources on the edge nodes. In our problem, the edge nodes are road side units (RSUs). The storage, computational and bandwidth capacities of the RSUs are constrained in comparison to the resource capacities on the cloud. Therefore, it is essential to allocate the edge resources such that maximum number of vehicles are serviced at minimal cost. The main contributions of this work are as follows:

1) Firstly, we derive the optimal allocation of data for a vehicle to a pair of two edges.
2) Then, we use the above optimal allocation and design a fast and efficient heuristic called E-PODS for data/service delivery to N vehicles over M edges.
3) We demonstrate the effectiveness of the proposed E-PODS heuristic in comparison to the baseline optimal approach and show that it has far less run time complexity and is near optimal in terms of total bandwidth cost and number of serviced vehicles.

II. Related Work

The research that is closest to our work are the papers on data delivery from infrastructure to vehicles [7] and online resource allocation to deliver services like computation offloading [8] for mobile nodes. In [7], the authors propose a trajectory-based forwarding scheme to deliver data from infrastructure nodes to moving vehicles in vehicular adhoc networks. An online edge cloud resource allocation algorithm is proposed in [8], which considers arbitrary user movement and variation in resource prices. Due to arbitrary user movement, the cloud does not a priori know the route that the vehicle will take. In contrast, it is a very likely scenario that a driver knows beforehand the route to the destination. In such circumstances, the cloud can exploit the route information to deploy data/service, which has been used in [7]. However, both these techniques do not consider any vehicle flow model, which characterizes the movement of traffic near the edge or between edges.

Our prior work [9] proposed an optimization framework considering vehicle flow model in order to allocate resources on the edge for data/service delivery. However, the time complexity of the optimization framework is very high. In this work, we address the above problem by proposing a near optimal fast heuristic to perform data/service delivery to vehicles via edge nodes/RSUs.

III. Problem Formulation

In this section, we describe the VEC system consisting of the *vehicle model*, *edge model* and the *delay model*. The cloud is connected to all the edges to deliver any data or service requested by the vehicles. Each edge connected to the cloud has a coverage area under which it can serve the request of the vehicles passing through that edge. A request to the cloud can be either for some data that needs to be downloaded or some computation offloading to the edge by the vehicle.

In our work, we consider that a request by a vehicle is served only by the edges, i.e., we only consider V2I communication mode. The data requested by a vehicle can be divided into several chunks and delivered to the vehicle through different edges. However, for service delivery only one of the edges in the route serves the request of the vehicle. We also consider that the cloud receives information about the route from the vehicles. Therefore, the cloud can send the required information for data and service delivery to the appropriate edges through which the vehicle passes.

A. VEC Model

Let us consider that there are N vehicles and M edges in our VEC model. Further, we will define our VEC model in terms of vehicle model, edge model and delay model.

Vehicle Model: The vehicle model is defined by a tuple $\alpha = \langle V_i, x_{i,j}, M_i, S_i \rangle$. Here, V_i denotes the ith vehicle. The vector $x_{i,j}$ contains the route information of the vehicle. If vehicle V_i passes through the edge E_j then $x_{i,j}$ is 1 otherwise the value of $x_{i,j}$ is 0. The total data size that a vehicle requests for download is represented as M_i. S_i is another tuple which has the information about the service request of a vehicle. S_i is defined by $\langle d_i, r_i, p_i \rangle$, where d_i is the data that needs to be transferred to the edge for processing, r_i is the data received by the vehicle from the edge and p_i is the number of processing units(in terms of VMs) required by the vehicle.

Edge Model: The edge model is defined by the tuple $\beta = \langle E_j, L_j, M_j, P_j, B_j, M_j^{occ}, P_j^{occ} \rangle$. E_j is the jth numbered edge. L_j is the coverage area of the edge. M_j is the memory capacity and P_j is the processing capacity(in terms of VMs) of the edge. B_j is the network bandwidth provided by the edge. Some of the resources of an edge may be already occupied. Therefore, we denote M_j^{occ} as the occupied memory and P_j^{occ} as the occupied processing capacity of the edge. These resources get utilized when a vehicle requests for data or service delivery.

Delay Model: The delay model is denoted by $\gamma = \langle t\_trv_{i,j}, t\_comm_{i,j} \rangle$. $t\_trv_{i,j}$ is the earliest time taken by a vehicle V_i to reach the edge E_j after sending the request for some data download. The time taken by the cloud to send the data chunk to the edge E_j is denoted by $t\_comm_{i,j}$. In case of service delivery the information that needs to be sent by the cloud to the edges is a binary value and therefore the time taken is negligible.

A vehicle data request can be served by multiple edges where each edge stores a part of the total data requested termed the data chunk. The data chunk that an edge E_j delivers to a vehicle V_i is denoted by the decision variable $m_{i,j}$. For a service request, the decision variable is $serv_{i,j}$. If an edge E_j processes a service request of a vehicle V_i, then the value of $serv_{i,j}$ will be 1, otherwise it will be 0.

A request for data download by a vehicle to the cloud contains the vehicle id V_i, route $x_{i,j}$ and size of the data M_i. In case of service delivery, the request to the cloud contains vehicle id V_i, route $x_{i,j}$ and the service request S_i.

Our goal is to find the values of $m_{i,j}$ and $serv_{i,j}$ while minimizing the total bandwidth cost of all the edges.

B. Optimization Framework

Now we present the optimization framework which was introduced in a previous work [9]. Firstly, we present the objective function followed by the constraints that need to be satisfied for successful data and service delivery. The optimization problem is given by

$$minimize \sum_{j=1}^{M} bw_j^{cost} \tag{1}$$

s.t.

$$\begin{aligned} m_{i,j} &= 0, i = 1..N, j = 1..M : x_{i,j} = 0 \\ m_{i,j} &\geq 0, i = 1..N, j = 1..M : x_{i,j} = 1 \\ m_{i,j} &\leq M_i, i = 1..N, j = 1..M : x_{i,j} = 1 \end{aligned} \tag{2}$$

$$\sum_{j=1}^{M} m_{i,j} \times x_{i,j} = M_i, i = 1..N \tag{3}$$

$$\sum_{j=1}^{M} serv_{i,j} \times x_{i,j} = 1, i = 1..N \tag{4}$$

$$m_{i,j} \times t\_comm_{i,j} \leq m_{i,j} \times t\_trv_{i,j}, \\ i = 1..N, j = 1..M \tag{5}$$

$$\sum_{j=1}^{N} m_{i,j} + M_j^{occ} \leq M_j, j = 1..M \tag{6}$$

$$\sum_{j=1}^{N} serv_{i,j} \times (d_i + r_i) + M_j^{occ} \leq M_j, j = 1..M \tag{7}$$

$$\sum_{j=1}^{N} serv_{i,j} \times p_i + P_j^{occ} \leq P_j, j = 1..M \tag{8}$$

$$m_{i,j} \leq D_{i,j}^{min,data}, i = 1..N, j = 1..M \tag{9}$$

$$serv_{i,j} * (d_i + r_i) \leq D_{i,j}^{min,serv}, i = 1..N, j = 1..M \tag{10}$$

where $bw_j^{cost} = \delta \times (1 + bw_j^{util})^2$,

$$bw_j^{util} = \frac{v_{i,j}}{L_j \times B_j} \sum_{i=1}^{N} m_{i,j} + \frac{1}{B_j} \sum_{i=1}^{N} \frac{d_i + r_i}{(\frac{L_j}{v_{i,j}} - t_{pi})},$$

$D_{i,j}^{min,data} = \frac{B_j}{k_j^{jam} \times v_{i,j}}$ and $D_{i,j}^{min,serv} = \frac{B_j \times (\frac{L_j}{v_{i,j}} - t_{p_i})}{k_j^{jam} \times L_j}$.

The objective function shown in Eq. (1) is the total bandwidth cost function considering all the edges in the VEC system. We use a non linear pricing model for bandwidth cost based on a pricing model in [10] used to balance bandwidth load across all edges. Here, Eq. (2) represents range constraint in which upper limit for data chunk $m_{i,j}$ is M_i and lower limit is 0. Eq. (3) and Eq. (4) represent accumulation constraints. In Eq. (3), the sum of data chunks received by a vehicle from all the edges should be equal to the required data M_i. In Eq. (4), it is ensured that only one edge can provide the service request of the vehicles, in case of service delivery. The constraint in Eq. (5) ensures that the data chunk reaches to the edge before the vehicle reaches to the same edge. The constraints in Eq. (6), Eq. (7) and Eq. (8) represent the edge resource constraints where the total allocated resources (memory, processing units) at an edge should be less than or equal to the total available resources. In Eq. (9) and Eq. (10), bandwidth schedulability constraints are ensured in which the data received by a vehicle should not exceed the minimum number of bytes that the vehicle can receive while in the coverage area of the edge. These constraints were discussed in detail in the previous work [9].

IV. E-PODS: Heuristic Based Data/Service Delivery

In this section, we present a novel fast and efficient heuristic for data/service delivery called E-PODS. E-PODS algorithm is based on obtaining edge-pairwise optimal solutions for data/service delivery, i.e., deriving optimal solutions for each vehicle considering pairs of edges at every iteration until all the edges are considered. The time complexity grows rapidly if we increase the number of edges beyond a pair in every iteration, which coincides with the optimization framework if M edges are considered.

In real-time scenarios, a large number of vehicles can enter the VEC network and request for data/service delivery. In order to derive the optimal resource allocations for data/service delivery, the optimization performed in the cloud must be fast enough to handle several vehicle requests. With increase in the number of vehicles, the time complexity for optimization framework would grow exponentially. Hence, we attempt to address this issue by proposing the fast heuristic approach E-PODS.

Let us recall that the bandwidth cost for an edge E_j is

$$bw_j^{cost} = \delta \times (1 + \frac{a_j}{k_j})^2 \tag{11}$$

where δ is the bandwidth cost factor, a_j is the total memory used at the edge E_j for all the vehicles (requiring data/service delivery) passing through the edge. k_j is a constant for an edge whose value is $\frac{v_{i,j}}{L_j \times B_j}$ as $v_{i,j}, L_j$ and B_j are all constants for a particular edge. The following theorem provides us the optimal allocation of data for a vehicle considering a pair of edges.

Theorem 1. *Given a vehicle V_i with data requirement M_i and a pair of edges with a_1 and a_2 as their total memory used, the minimum total bandwidth cost considering the 2 edges E_1 and E_2 is obtained when the first edge is allocated the value below*

$$m_1 = \begin{cases} \frac{\frac{1}{k_2} - \frac{1}{k_1} + \frac{M_i + a_2}{k_2^2} - \frac{a_1}{k_1^2}}{\frac{1}{k_1^2} + \frac{1}{k_2^2}}, & \textit{if } m_1 > 0 \\ 0, & \textit{otherwise} \end{cases}$$

Here k_1 and k_2 are the constants described above for the edges E_1 and E_2 respectively.

Proof: Let us assume that m_1 is the amount of memory allocated on edge E_1 and M_i - m_1 amount of memory is allocated on edge E_2. Then the total bandwidth cost will be given by

$$Cost = \delta(1 + \frac{m_1 + a_1}{k_1})^2 + \delta(1 + \frac{M_i - m_1 + a_2}{k_2})^2 \tag{12}$$

The above equation is quadratic in nature for variable m_1. Therefore, it attains a minimum value when its derivative w.r.t. m_1 is zero. On equating its derivative with zero we get

$$\frac{1}{k_1} + \frac{1}{k_1^2}(m_1 + a_1) = \frac{1}{k_2} + \frac{1}{k_2^2}(M_i - m_1 + a_2) \quad (13)$$

On solving Eq. (13) we get,

$$m_1 = \frac{\frac{1}{k_2} - \frac{1}{k_1} + \frac{M_i + a_2}{k_2^2} - \frac{a_1}{k_1^2}}{\frac{1}{k_1^2} + \frac{1}{k_2^2}} \quad (14)$$

If the above expression becomes negative then we take 0 as the solution for m_1 since memory cannot be negative. ■

Now we present the E-PODS algorithm using Theorem 1 which considerably reduces time complexity. In Algorithm 1, we calculate the $m_{i,j}$ values for all the vehicles passing through the edges using the Theorem 1. In Algorithm 1, we first find the number of edges that each vehicle goes through denoted by C_i as shown in line 1. Then, we compute the demand ratio of each vehicle as M_i/C_i, where M_i is the data requirement of vehicle V_i. The demand ratio signifies the magnitude of resource requirement of each vehicle per edge. Then, we sort all the vehicles in increasing order of their demand ratio as shown in line 2. We iterate through this list and for each vehicle we prepare a list of edges that the vehicle passes through as shown in lines 19 and 20. Then we sort the list of edges in descending order of their available memory as presented in line 21 obtaining $SortedList$. We iterate through this $SortedList$ and calculate the value of m_1 using the first two edges in the list as shown in the function $CalculateData$ from lines 7 to 18 using Theorem 1.

In the function $CalculateData$, we check for the edge resource and bandwidth schedulability constraints for m_1 as shown in lines 14 to 16. Then we write this value m_1 in the allocation matrix $MemMatrix$ and increase the value of memory occupancy for edge e_1 by m_1 as shown in lines 26 and 27. If only two edges are left in the list then we assign the memory value $M_i - m_1$ to the last edge as shown in line 30 and increase the value of memory used for edge e_2 by m_2 as shown in line 31. Then we update the value of M_i with $M_i - m_1$ as m_1 amount of data requirement is fulfilled by edge e_1 as shown in line 34. Finally, we remove this edge e_1 from the $SortedList$ and iterate again with the remaining edges. This algorithm is used to calculate the $m_{i,j}$ values for data delivery. In case of service delivery, since only one of the edges serves the request of the vehicle, we find the solution using brute force approach.

The time complexity for E-PODS is $O(NM^2 \log M + N \log N)$. Initially, as we sort the list of all vehicles in increasing order of their demand ratio we get the term $O(N \log N)$ in the complexity. In the algorithm, as we iterate over all the vehicles and for each vehicle we iterate over the edges it passes through, the complexity becomes $O(NM)$. Further, as we sort the list of edges for each vehicle, the complexity multiplies by $O(M \log M)$ and becomes $O(NM^2 \log M)$. After summation we get the total complexity as $O(NM^2 \log M + N \log N)$.

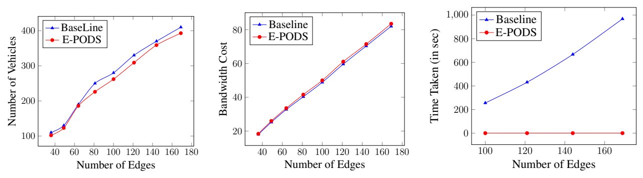
Algorithm 1: Calculating $m_{i,j}$ values for all the vehicles

```
Result: $m_{i,j}$ values for all the vehicles
1  $C_i \leftarrow$ Number of edges through which the vehicle $V_i$ passes.
2  $V_{set} \leftarrow$ Sorted list of vehicles in increasing order of their demand ratio which is $\frac{M_i}{C_i}$
3  $A \leftarrow$ List of memory used for all the edges.
4  $K \leftarrow$ List of $k_j = \frac{v_{i,j}}{L_j \times B_j}$ for each of the edge $E_j$
5  $AvlMem \leftarrow$ List of available memory for all the edges
6  $MemMatrix \leftarrow$ Empty matrix of size $N \times M$ for $m_{i,j}$ values
7  Function CalculateData(List,M):
8      $e_1 \leftarrow List[0]$
9      $e_2 \leftarrow List[1]$
10     $m_1 \leftarrow \frac{\frac{1}{K[e_2]} - \frac{1}{K[e_1]} + \frac{M + A[e_2]}{K[e_2]^2} - \frac{A[e_1]}{K[e_1]^2}}{\frac{1}{K[e_1]^2} + \frac{1}{K[e_2]^2}}$
11     if $m_1$ <0 then
12         $m_1$ = 0
13     end
14     if $m_1 > D_{i,j}^{min,data}$ || $m_1 > AvlMem[e_1]$ then
15         $m_1 = min(D_{i,j}^{min,data}, AvlMem[e_1])$
16     end
17     return $m_1, e_1, e_2$
18 End Function
19 while $v_i$ in $V_{set}$ do
20     $EdgeList \leftarrow$ List of edges through which $v_i$ passes
21     $SortedList \leftarrow$ Sorted $EdgeList$ in descending order of available memory
22     $M_i \leftarrow$ Data required by vehicle $v_i$
23     $len \leftarrow$ length of $SortedList$
24     for $j = 0; j \leq len - 2; j++$ do
25         $m_1, e_1, e_2 \leftarrow$ CalculateData($SortedList, M_i$)
26         $MemMatrix[v_i][e_1] \leftarrow m_1$
27         $A[e_1] += m_1$
28         $AvlMem[e_1] = AvlMem[e_1] - m_1$
29         if j==len-2 then
30             $MemMatrix[v_i][e_2] \leftarrow M_i - m_1$
31             $A[e_2] += M_i - m_1$
32             $AvlMem[e_2] = AvlMem[e_2] - (M_i - m_1)$
33         end
34         $M_i \leftarrow M_i - m_1$
35         $SortedList.remove(e_1)$
36     end
37 end
```

V. Experimental Setup and Results

In this section, firstly we present the experimental setup. The results obtained by E-PODS heuristic and a baseline optimal approach are then presented and inferences are drawn from them. First, we conducted our experiments on a synthetic data set and then we used the vehicular traces from a real data set of Luxembourg city [11] from SUMO. We conducted our experiments in Matlab and the optimization solvers for the baseline approach were SDPT3 and Gurobi. The baseline case here will be the optimal allocation that we get by using the above mentioned optimization tools.

A. Synthetic Data set



(a) (b) (c)

Figure 1: Synthetic data set results. (a) maximum number of vehicles served successfully for a given number of edges, (b) total bandwidth cost for a given number of edges, and (c) run-time complexity

In this subsection, we discuss the experiment and results for the synthetic data set. We modeled our edge network in square grids of different sizes. Below is the list of values that we generated synthetically, for the parameters.

- We used different values for the number of edges, M ranging from 36 (6×6) up to 169 (13×13).
- The routes of 450 vehicles were generated randomly such that the vehicles pass through different edges.
- The vehicle density at jam (k_j^{jam}) was taken as 35.
- Memory requirement (M_i) of each vehicle was generated randomly between 60 Mbits to 80 Mbits.
- Coverage distance (L_j) of the edges was randomly generated using uniform distribution between 0.6 miles to 1.6 miles.
- Memory capacity (M_j) of the edges was randomly generated between 400 Mbits to 500 Mbits using uniform distribution and the occupied memory (M_j^{occ}) was randomly generated between 0 Mbits to 150 Mbits.
- The computation capacity of the edges (P_j in number of VMs) was randomly generated between 24 and 40 using uniform distribution and occupied computation capacity was randomly generated between 1 and 3.
- The vehicle's processing requirement (p_i) was randomly generated between 1 and 10 and the processing time (t_{pi}) was randomly generated between 1 - 10 sec.
- The data that the vehicle sends (d_i) to the edge and the data that it receives (r_i) from the edge were randomly generated between 1 Mbits to 15 Mbits.
- The bandwidth of the edges (B_j) was randomly generated between 8 Mbps to 15 Mbps.
- Free flowing velocity (v_j^f) of the vehicles was randomly generated between 50mph to 70mph as taken from [12].

In synthetic dataset 100% of the vehicles request for data download. The results obtained from the experiments and our inferences from them are described below.

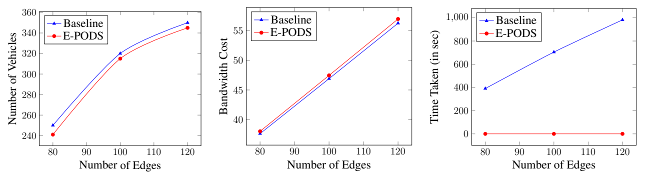
Maximum number of vehicles served with varying number of edges: For a given number of edges in a network, there is a maximum limit to the number of vehicles that can be serviced under the constraints. The higher the value, the better is the resource allocation. We compare this maximum limit while using our E-PODS heuristic and the given baseline approach. From the plot in Fig. 1(a) we can see that even though the baseline accommodates more vehicles, E-PODS serves almost similar number of vehicles. In the worst case, E-PODS serves only 9.6% less vehicles than the baseline approach (24 vehicles less) for 81 edges. We can also observe that the difference in number of vehicles serviced successfully is larger for higher number of edges. This is because the total number of vehicles has also increased thereby making it more difficult to accommodate them.

Variation of bandwidth cost with varying number of edges: In this section, we present the total bandwidth cost that is incurred for data delivery with varying number of edges while using E-PODS and the baseline approaches. Currently, we do not use a monetary value to compute bandwidth cost. We observe an increase in the bandwidth cost with increasing number of edges for both methods. It can seen from the plot in Fig. 1(b) that the E-PODS method has a slightly greater bandwidth cost compared to the baseline, with the worst case being a difference of 1.51 units for 169 edges, which is only 1.8% higher.

Time taken/Complexity of the method: In this section we present a comparison of the time complexity to obtain edge resource allocation using the baseline approach and our proposed E-PODS heuristic. We can observe from the plot in Fig. 1(c) that there is a significant reduction in the time taken by E-PODS when compared to the baseline approach. The time taken is in the order of milliseconds for E-PODS while it is in the order of several 100 seconds for the baseline. On an average, the baseline approach takes 580.29 secs more than the E-PODS heuristic to obtain edge resource allocations. Both methods show an increase in the time taken with increase in number of edges even though the increase is very small and in milliseconds for our proposed E-PODS heuristic.

B. Real Data set

In this part, we present our experiments and results for the baseline approach and our proposed E-PODS heuristic using a real case scenario. For this scenario, we used the data set of traffic traces from Luxembourg city [11] that



(a) (b) (c)

Figure 2: Real Dataset Results. (a) Maximum number of vehicles serviced successfully for a given number of edges, (b) Total bandwidth cost for a given number of edges and (c) Run-Time Complexity

was simulated using SUMO. From this data set, we obtained the route of the vehicles, the distances that these vehicles are covering and the number of vehicles. The number of edges were varied from 80 to 120 in steps of 20. The other parameters such as the coverage distances (L_j), memory capacity (M_j), processing capacity (P_j), bandwidth (B_j) etc., were generated similarly as discussed in the synthetic data set due to unavailability of these parameters in the real data set. Here, 80% of the vehicles request for data download while 20% of the vehicles request for service delivery. This ratio remains same for all the number of edges and vehicles. We now present the results for the real case scenario below.

Maximum number of vehicles served with varying number of edges: In this result, we show the variation of successfully served number of vehicles with increasing number of edges. From the plot in Fig. 2(a) we can see that, the baseline serves more vehicles than E-PODS. In the worst case, E-PODS serves only 3.6% less vehicles (i.e., 9 vehicles less) than the baseline approach for the case of 80 edges.

Variation of bandwidth cost with varying number of edges: It can be seen from the plot in Fig. 2(b) that our proposed E-PODS heuristic incurs a slightly greater bandwidth cost compared to the baseline approach. In the worst case the bandwidth cost using E-PODS heuristic is 0.707 units higher for 120 edges, which is only 1.25% higher than the bandwidth cost using baseline approach. The bandwidth cost increases with the increase in the number of edges for both approaches.

Time taken/Complexity of the method : We can observe from the above plot in Fig. 2(c) that there is a significant reduction in the time taken to run E-PODS when compared to the baseline approach. The time taken is in the order of milliseconds for E-PODS while it is in the order of seconds for the baseline approach. The average reduction in the time taken is 692.053 sec using the E-PODS heuristic compared to the baseline approach. Both methods show an increase in the time taken with an increase in the number of edges, even though the increase is minimal with E-PODS.

VI. Conclusion

In this work, we proposed a fast and efficient heuristic for data/service delivery called E-PODS. The proposed E-PODS heuristic consists of an algorithm to allocate edge resources to the vehicles in a network with lesser time complexity than previously proposed optimization framework. Finally, we conducted experiments with both synthetic and real data set and demonstrated that E-PODS had considerably lesser time complexity, while the number of vehicles serviced and bandwidth cost were comparable to the optimal approach.

References

[1] Y.-C. Liu and B. Kim, "An optimization framework to select edge servers for automotive connected services," in *IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–2.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, pp. 1–24, 2020.

[3] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[4] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[5] T. Carlfalk, "How does edge computing benefit connected cars?" https://www.wirelesscar.com/how-does-edge-computing-benefit-connected-cars/, 2019.

[6] S. Shurpali, "Role of edge computing in connected and autonomous vehicles," https://www.einfochips.com/blog/role-of-edge-computing-in-connected-and-autonomous-vehicles/, 2020.

[7] J. Jeong, S. Guo, Y. Gu, T. He, and D. H. Du, "Trajectory-based statistical forwarding for multihop infrastructure-to-vehicle data delivery," *IEEE Transactions on Mobile Computing*, vol. 11, no. 10, pp. 1523–1537, 2012.

[8] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *37th IEEE International Conference on Distributed Computing Systems*, 2017, pp. 1281–1290.

[9] D. Gangadharan, O. Sokolsky, I. Lee, B. Kim, C.-W. Lin, and S. Shiraishi, "Bandwidth optimal data/service delivery for connected vehicles via edges," in *11th IEEE International Conference on Cloud Computing (CLOUD)*, 2018, pp. 106–113.

[10] A. Ghavami, Z. Li, and H. Shen, "On-demand bandwidth pricing for congestion control in core switches in cloud networks," in *9th IEEE International Conference on Cloud Computing (CLOUD)*, 2016, pp. 867–870.

[11] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 1–8.

[12] W. L. Tan, W. C. Lau, O. Yue, and T. H. Hui, "Analytical models and performance evaluation of drive-thru internet systems," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 207–222, 2011.