Global Edge Bandwidth Cost Gradient-based Heuristic for Fast Data Delivery to Connected Vehicles under Vehicle Overlaps

Akshaj Gupta*, Joseph John Cherukara*, Deepak Gangadharan*, BaekGyu Kim†, Oleg Sokolsky‡ and Insup Lee‡
*IIIT Hyderabad, †DGIST, ‡University of Pennsylvania
Email: *{akshaj.gupta, joseph.cherukara}@research.iiit.ac.in, *deepak.g@iiit.ac.in, †bkim@dgist.ac.kr,
‡{sokolsky, lee}@cis.upenn.edu

***Abstract*—The emergence of vehicle connectivity technologies and associated applications have paved the way for increased consumer interest in connected vehicles. These modern day vehicles are now capable of sending/receiving vast amounts of data and offloading computation (which is one possible service) to servers thereby improving safety, comfort, driving experience, etc. In the early stages of connectivity, all the data communication and computation offloading happened between the cloud server and the vehicles. However, this is not feasible in scenarios having strict timing requirements and bandwidth cost constraints. Vehicular Edge Computing (VEC) demonstrated an efficient way to tackle the above problem. In order to optimally utilize the resources of the edge servers for data delivery, an efficient edge resource allocation framework needs to be developed. In a recent work, data/service delivery to connected vehicles assumed a worst-case scenario that all vehicles with routes passing through an edge appear in the edge coverage region simultaneously. However, this worst-case scenario is very pessimistic, which results in overestimation of edge resources. We address this by precisely computing the set of vehicles which simultaneously appear in the coverage region of an edge (which we call *vehicle overlaps*). In this work, we first propose an optimization framework for edge resource allocation that minimizes the bandwidth cost of data delivery to connected vehicles while considering the traffic flow and vehicle overlaps. Then, we propose an efficient heuristic to deliver data based on minimizing global edge bandwidth cost gradient under vehicle overlaps. We demonstrate the improvement in resource allocation considering vehicle overlaps. Using real world traffic data, we also demonstrate reduction in data delivery times using the proposed heuristic.**

***Keywords*-Connected Vehicles; Edge Computing; V2I; Data Delivery**

I. Introduction

The advances in sensing, communication and computation have transformed the capabilities of a modern day vehicle and facilitated the rise of a connected vehicle ecosystem. Apart from the above mentioned advances, there have been significant developments in automotive applications that leverage the power of connectivity. Emergence of multiple driver assistance applications have enabled collaboration with infrastructure and other vehicles in the ecosystem in order to enhance the safety and driving experience. Intelligent Transportation System (ITS) allows data dissemination (such as High Definition (HD) Map data [1], software/function update [2], [3], etc.) to vehicles through infrastructure to vehicle (I2V) mode of connectivity. Data can also be transferred from vehicles to the infrastructure using vehicle to infrastructure connectivity (V2I) for applications such as road safety, traffic control, vehicle diagnostics, etc.

With the rise in number of connected vehicles, there has also been an increase in the number of automotive applications leveraging the power of connectivity and requiring large storage [1] or processing [4] capacity. Unfortunately, vehicle platforms have storage and processing capacity constraints making it infeasible to execute applications needing large storage or computation capacity entirely within the vehicle. A previous solution to address the resource constraint problem was to execute the applications partially or entirely in the cloud (known as mobile cloud computing (MCC) [5]). However, it suffered from excessive communication delay and huge bandwidth requirements to transfer data. The above problems led to the paradigm of mobile edge computing (MEC) [6] and its integration with vehicular networks giving rise to *Vehicular Edge Computing* (VEC) [4]. In a VEC system, edge nodes (e.g., Road Side Units (RSUs) or a local server) are located in close proximity of the vehicles, thereby reducing communication delay and backhaul bandwidth issues that a cloud based solution has. Moreover, the edge nodes are equipped with more storage and computation resources compared to the vehicle, making it ideal for automotive applications to exploit.

A. Motivation

One of the primary tasks to realize data delivery is *edge resource allocation*. Based on the application that a vehicle needs to execute, it requires allocation of adequate memory on the edge node as well as a share of the edge node bandwidth. Given that there are many vehicles and edge nodes in a VEC system and constraints on edge resources, the edge resource allocation problem is non trivial. In addition, the mobility of vehicles makes the VEC system highly dynamic. The density of traffic may vary from one edge node to another, which has its influence on resource allocation. The routes of the vehicles also restrict the candidate edge nodes for resource allocation. All the above factors must be taken into consideration while performing resource allocation.

We consider a scenario with multiple edge nodes in the

VEC system and several vehicles moving along their path from a source to destination. A vehicle can request the cloud for data and it is allocated resources on the edges to enable the delivery. For a scenario, where the vehicle routes and travel time schedules are known upfront, it is possible to perform the optimization in the cloud and dispatch the data to the edges before the vehicle reaches the edges.

In this work, our primary focus is to propose a novel edge resource allocation technique for data delivery to connected vehicles considering vehicle flow model and vehicle overlaps in edge coverage regions. *We consider two or more vehicles to be overlapping when they are present in the coverage area of an edge at the same time*. The main contributions of this work are

1) Firstly, we propose the algorithms to determine the overlapping set of vehicles and use the overlapping sets to formulate the edge resource constraints.
2) To the best of our knowledge, our proposed optimization framework is the first of its kind for resource allocation on edge nodes that considers vehicle flow model in conjunction with vehicle overlaps at an edge.
3) Due to the high time complexity of the optimization framework, we propose a fast global bandwidth cost-gradient based heuristic to perform data delivery considering vehicle overlaps.
4) Finally, we conduct few experiments with real data set demonstrating the improvement in resource allocation compared to a recent work [7] and also showing the efficacy of the proposed heuristic approach in reducing run-time complexity.

II. Related Work

In this section, we will primarily discuss prior research done in the related areas of resource allocation techniques in edge computing and data delivery to mobile nodes using edge nodes.

One of the key services provided in MEC is computation offloading as it can reduce the energy required for computation and can speed up the computation process. There are few works which proposed techniques to allocate resources for computation offloading on a single node such that the execution delay is minimized [8], [9]. However, in [9], power consumption is additionally optimized in conjunction with delay. Similarly, there are other works that proposed techniques for resource allocation on multiple edge nodes while minimizing execution delay [10] or a combination of execution delay with power consumption [11]. All the works mentioned above proposed techniques to allocate resources on the edge for computation offloading from a mobile device and do not consider any vehicle mobility, which is essential to incorporate in our scenario of connected vehicles.

The research that is closest to our work are the proposed methods for data delivery from infrastructure to vehicles [12] and online resource allocation for computation offloading [13] from mobile nodes. In [12], a trajectory-based packet forwarding scheme is proposed to deliver data from infrastructure nodes to moving vehicles in vehicular adhoc networks using packet's delivery delay distribution and vehicle's travel delay distribution. An online edge cloud resource allocation algorithm was proposed in [13], which considers arbitrary user movement and resource price variation. 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 transmit data, which has been used in [12]. However, both these works do not consider any vehicle flow model, which characterizes the movement of traffic near the edge or between edges. The work in [7] is the closest to our work for data delivery, but the approach proposed by the authors assumes that all the vehicles with routes through an edge pass the edge at the same time. In this work, we remove this pessimism and propose an improved optimization framework for resource allocation and show the advantages using a real traffic data set. Moreover, the optimization framework in [7] has high run-time complexity. In this work, we propose an fast heuristic algorithm to address the issue of run-time complexity.

III. Problem Formulation

In this section, we present our connected vehicles model consisting of cloud, edges and vehicles. The cloud is connected to all the edges in the network to deliver any requested data by the vehicles. Each edge has a coverage area in which it can serve all the vehicles passing through the edge.

In this work, we are considering that any request by a vehicle is served through the edges only. The data requested by a vehicle can be divided into several chunks and can be delivered to the vehicle by different edges in the route. A request to the cloud by a vehicle also contains the information about the route of the vehicle along with information about the request for data delivery. The cloud maintains the route map of each vehicle. Thus, the cloud can deliver the data chunks to the relevant edges through which the vehicle passes. We model our VEC system in terms of a vehicle model, an edge model and a delay model.

Vehicle Model: A vehicle model is defined by the tuple $\alpha = \langle V_i, x_{i,j}, M_{V_i} \rangle$. All the definitions are as mentioned in Table I.

Edge Model: Each edge is equipped with resources like memory, processing capacity (in terms of VMs) and bandwidth. We model our edge as a tuple given by $\beta = \langle E_j, M_{E_j}, B_j, M_j^{occ} \rangle$. We describe all the variables in Table I.

Table I: Variables and their description

Variable Name	Description
N_v	Number of Vehicles in the VEC system
N_e	Number of Edges in the VEC system
V_i	Vehicle with index i
E_j	Edge with index j
bw_j^{cost}	Bandwidth cost of an edge E_j
$m_{i,j}$	Amount of data that vehicle V_i will receive from edge E_j
$x_{i,j}$	If $x_{i,j}$=1 then edge E_j will fall in the route of vehicle V_i, for other case the value of $x_{i,j}$ is 0
$v_{i,j}$	Velocity of vehicle V_i in the coverage distance of edge E_j
v_j^f	Free flow velocity of the vehicles in the coverage distance of the edge E_j
M_{V_i}	The memory requirement of vehicle V_i for data delivery
M_{E_j},M_j^{occ}	Memory and occupied memory of the edge E_j respectively
B_j	Bandwidth of the edge E_j for communication with the vehicles.
$t\_trv_{i,j}$	Time taken by vehicle V_i to reach to Edge E_j
$t\_comm_{i,j}$	Time taken to send the data chunk $m_{i,j}$ to the edge E_j
n_j	Number of sets of overlapping vehicles at edge E_j
$S_{1,j} \dots S_{n_j,j}$	n_j sets of overlapping vehicles at edge E_j
k_j	Vehicle density in the coverage area of edge E_j
k_j^{jam}	Vehicle density during the jam
$D_{i,j}^{min,data}$	Minimum data that a vehicle can receive from edge E_j in case of data delivery

Delay Model: The delay model is defined by the tuple $\gamma = \langle t\_trv_{i,j}, t\_comm_{i,j} \rangle$. We describe the variables in Table I.

A request for data delivery contains information about the route $x_{i,j}$, the vehicle V_i and the data size required M_{V_i}. Our goal is to find the values of $m_{i,j}$ that will minimize the total bandwidth utilization cost while considering the vehicle overlaps and vehicle flows. We also develop a heuristic that will find the $m_{i,j}$ values and enables significantly faster data delivery.

IV. Proposed Solution

In this section, we present our optimization framework, which consists of the constraints that need to be satisfied for successful data delivery in our VEC system. In addition, we present the algorithms used to formulate some of the constraints discussed. Finally, we present a fast and efficient global edge bandwidth cost gradient-based heuristic to perform resource allocation for data delivery with lesser time complexity.

Now we present the optimization framework with the objective function and the constraints that need to be satisfied for successful data delivery. Given the vehicle model α, edge model β and delay model γ the optimization problem is given by

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

s.t.

$$\begin{aligned} &m_{i,j} = 0, i = 1..N_v, j = 1..N_e : x_{i,j} = 0 \\ &m_{i,j} \geq 0, i = 1..N_v, j = 1..N_e : x_{i,j} = 1 \\ &m_{i,j} \leq M_{V_i}, i = 1..N_v, j = 1..N_e : x_{i,j} = 1 \end{aligned} \tag{2}$$

$$\sum_{j=1}^{N_e} m_{i,j} \times x_{i,j} = M_{V_i}, i = 1..N_v \tag{3}$$

$$\begin{aligned} &m_{i,j} \times t\_comm_{i,j} \leq m_{i,j} \times t\_trv_{i,j}, \\ &i = 1..N_v, j = 1..N_e \end{aligned} \tag{4}$$

$$\begin{aligned} &\max(\sum_{i_1 \in S_{1,j}} m_{i_1,j}, .., \sum_{i_{n_j} \in S_{n_j,j}} m_{i_{n_j},j}) + M_j^{occ} \leq M_{E_j} \\ &, j = 1..N_e, i_1, \dots, i_{n_j} \in \{1 \dots N_v\} \end{aligned} \tag{5}$$

$$m_{i,j} \leq D_{i,j}^{min,data}, i = 1..N_v, j = 1..N_e \tag{6}$$

where

$$bw_j^{cost} = \delta \times (1 + bw_j^{util})^2, \tag{7}$$

$$bw_j^{util} = \frac{v_{i,j}}{L_j \times B_j} \sum_{i=1}^{N} m_{i,j}, \tag{8}$$

$D_{i,j}^{min,data} = \frac{B_j}{k_j^{jam} \times v_{i,j}}$

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 as in [14], which is used to balance bandwidth load across all edges. Here, Eq. (2) represents range constraint in which data chunk $m_{i,j}$ is upper bounded by M_{V_i} and lower bounded by 0. The accumulation constraint shown in Eq. (3) ensures that the sum of data chunks received by a vehicle from all the edges should be equal to the data size M_{V_i}. The constraint in Eq. (4) ensures that the data chunk is transmitted to an edge before the vehicle reaches that edge. The edge resource constraint in Eq. (5) ensures that the total allocated resources (memory) at an edge should be less than or equal to the total available resources. In [7], the above constraint formulation was pessimistic assuming that all vehicles passing through an edge arrive at the edge simultaneously, which is not realistic. It results in over allocation of resources on an edge. In this work, we perform a more accurate allocation by deriving n_j sets of vehicle overlaps at every edge E_j ($j = 1 \dots N_e$) shown as $S_{1,j} \dots S_{n_j,j}$ in Eq. (5). The indices $i_1 \dots i_{n_j}$ represent the indices of the vehicles in the overlapping sets $S_{1,j} \dots S_{n_j,j}$. The algorithms used to find the sets of overlapping vehicles

and the constraint in Eq. (5) are described in Section IV-A. In Eq. (6), bandwidth schedulability constraints is shown, which ensures that 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. The speed of the vehicle in the coverage region of an edge $v_{i,j}$ can be calculated using the linear relationship between speed and density as proposed by Greenshields in [15]. It is given as

$$v_{i,j} = v_j^f(1 - \frac{k_j}{k_j^{jam}}) \tag{9}$$

A. Algorithms to Formulate Edge Resource Constraints under Vehicle Overlaps

The resources of an edge are utilized by the vehicles for data delivery. Given an edge, the total memory usage at any instant should not exceed the total memory capacity of the edge. We determine the memory usage of an edge by considering the memory requirements of a set of vehicles which are present concurrently in the coverage area of the edge at some particular instant of time. We describe this set as an overlapping set of vehicles. We will now describe the algorithm we use to find the overlapping sets of vehicles.

Given the route information, vehicles' speed on the route, there are a number of works that can estimate the time of arrival. One notable approach is to apply a learning based method that can predict ETA based on collected data [16], [17]. However, this approach requires a large number of data points to be continuously collected upon individual drivers' consent. Hence, the performance would be different depending on the amount of data collected by the time the estimation needs to be triggered. In our work, we propose an analytic way to calculate the Earliest Time of Arrival (ETA) and Latest Time of Departure (LTD) based on the distance of the requesting vehicle to the edges and the vehicle density; in particular, we envision the real-time vehicle density can be publicly accessed by the recent U.S. government driven effort [18]. Our work may complement the existing ETA estimation approach to improve the estimation result.

First, using the data from vehicles and edges such as route information, vehicles' speed, distance covered by the vehicles at different sections of route and coverage distance of the edges(L_j), we can find the ETA and LTD of a vehicle in the coverage area of an edge. Towards this, we first calculate the maximum speed $MaxSpeed_{m,n} = v_{m,n}^f \times (1 - \frac{k^{min}}{k_{m,n}^{jam}})$ and minimum speed $MinSpeed_{m,n} = v_{m,n}^f \times (1 - \frac{k^{max}}{k_{m,n}^{jam}})$ of the vehicles in between edges using the minimum (k^{min}) and maximum (k^{max}) vehicle density (vehicle per unit distance). The density values can be obtained by the cloud as a result of traffic monitoring over a period of time. The ETA for vehicle V_i at edge E_j is then calculated using $MaxSpeed_{m,n}$ by considering the times spent in each of the road segments until the edge is reached as given by

$$ETA_{i,j} = \frac{A_i}{MaxSpeed_{m,n}} + \sum_{m,n \in Q_{i,j}} \frac{Dist_{m,n}}{MaxSpeed_{m,n}} + \sum_{k \in \{Q_{i,j} - E_j\}} \frac{L_k}{v_{i,k}} \tag{10}$$

where $Q_{i,j}$ is the set of pair of edges E_j and all the preceding adjacent edges through which vehicle V_i passes.

This may involve length of the coverage regions of each of the preceding edges (L_k), length of the road segments between two adjacent edges ($Dist_{m,n}$) and distance of the initial segment from starting point to the first edge (A_i). Similarly, the LTD is calculated by using $MinSpeed_{m,n}$ and adding the traveling time for the coverage length of the edge under consideration as given by.

$$LTD_{i,j} = \frac{A_i}{MinSpeed_{m,n}} + \sum_{m,n \in Q_{i,j}} \frac{Dist_{m,n}}{MinSpeed_{m,n}} + \sum_{k \in Q_{i,j}} \frac{L_k}{v_{i,k}} \tag{11}$$

After calculating the Earliest Time of Arrival(ETA) and Latest Time of Departure(LTD) of a vehicle passing through the edge E_j, we can find the sets of vehicles which are overlapping while passing through E_j. Algorithm 1 presents the procedure to find the overlapping sets of vehicles. First we sort the list of ETA and LTD in the ascending order of ETA for all the vehicles passing through the edge E_j as shown in lines 1 and 2 into $Sort_{ETA}$ and $Sort_{LTD}$ respectively. If the ETA time from $Sort_{ETA}$ of a vehicle V_i is less than or equal to ETA time from $Sort_{ETA}$ of vehicle V_k and ETA time of vehicle V_k is less than the LTD time from $Sort_{LTD}$ of vehicle V_i, then we add the vehicle V_k in the overlapping set of vehicle V_i. This is shown in lines 11 and 12 of Algorithm 1. We examine the above while iterating over all the vehicle pairs in the order of the sorted ETA list and find all possible overlapping sets of vehicles passing through the edge E_j as shown in step 6 and 10. For example, if the sorted ETA order for vehicles is V_1, V_2 and V_3 for an edge E_j, then the algorithm would first find the overlapping set for V_1 by checking the above comparison of V_1 with V_2 and V_3. Then we would find the overlapping set for V_2 by checking the comparison with V_3, which will give us all the possible overlapping sets for edge E_j .

We can formulate the edge resource constraint for data delivery by considering the overlapping set that consumes the maximum resources and this should not exceed the edge resource capacity. Then, the edge memory constraint for data delivery is given by Eq. (5).

Algorithm 1: Finding the sets of vehicles which are overlapping while passing through an edge.

Input : ETA, LTD of all the vehicles passing through the edge E_j and list of vehicles which passes through edge E_j

Output: Superset S which contains $S_{1,j}$ to $S_{n_j,j}$, the sets of overlapping vehicles which passes through edge E_j

1 $Sort_{ETA} \leftarrow$ Sorted list in ascending order of Earliest Time of Arrivals of all the vehicles passing through edge E_j.
2 $Sort_{LTD} \leftarrow$ Sorted list of Latest Time of Departures in the order of $Sort_{ETA}$ of all the vehicles passing through the edge E_j.
3 $len \leftarrow$ Length of list of vehicles passing through edge E_j
4 $S \leftarrow$ Empty set
5 $i \leftarrow 0$
6 **while** $i < len$ **do**
7 $list \leftarrow$ List of passing vehicles at edge E_j
8 $k \leftarrow i$
9 $ov\_set \leftarrow$ Empty set
10 **while** $k < len$ **do**
11 **if** $Sort_{ETA}[i] \leq Sort_{ETA}[k]$ *and* $Sort_{ETA}[k] < Sort_{LTD}[i]$ **then**
12 $ov\_set.add(list[k])$
13 k++
14 $S.add(ov\_set)$
15 i++

B. Global Cost Gradient Based Method

In this section, we present a fast global edge bandwidth cost gradient-based algorithm to find solution for data delivery while minimizing bandwidth cost. The heuristic algorithm selects the edge that has the least increase in edge bandwidth cost among all the edges for a specific quantum of data chunk. This makes the algorithm cost efficient and fast.

In case of real-time scenarios, several vehicles can request to the cloud for services. The optimization framework may not be fast enough to handle too many requests of the vehicles. The time complexity of the optimization framework grows exponentially with the increase in number of vehicles and edges. Hence, we attempt to address this issue by proposing a global edge bandwidth cost gradient-based heuristic.

The bandwidth cost at an edge E_j for data delivery can be expressed as

$$bw_j^{cost} = \delta \times (1 + \frac{a_j}{k_j})^2 \tag{12}$$

where δ is the bandwidth cost factor, a_j is the total memory used at the edge E_j and $k_j = \frac{v_{i,j}}{L_j \times B_j}$ is the constant for edge E_j as $v_{i,j}, L_j$ and B_j are all constants for a particular edge.

Let us assume that an extra m^* amount of data is used at an edge E_j to serve a vehicle request. Then we can say the increase in total bandwidth cost would be

$$\Delta = \delta \times (1 + \frac{a_j + m^*}{k_j})^2 - \delta \times (1 + \frac{a_j}{k_j})^2 \tag{13}$$

which is

$$\Delta = \frac{\delta \times m^*}{k_j} \times (2 + \frac{2 \times a_j + m^*}{k_j}) \tag{14}$$

We used Eq. (14) in Algorithm 2 to calculate the minimum bandwidth cost incurred for serving vehicles' requests. The magnitude of the quantum m^* is an important factor that will influence the closeness of the total bandwidth cost to optimal value.

Now we present the global edge bandwidth cost gradient-based heuristic for fast data delivery. In Algorithm 2, we calculate the $m_{i,j}$ values for all the vehicles passing through the edges. We first maintain a few lists like list of memory used at all of the edges(A), list of constant $k_j's$ (K), list of available memory for all the edges ($AvlMem$), list of total memory used at an edge E_j by an overlapping set S_k ($Mem_{S_{k,j}}$), list of memory requirement of all the vehicles (M), list of all the edges ($EdgeList$), a list of list which contains all the vehicles passing through each of the edge ($VehicleList$) and an empty matrix to store $m_{i,j}$ values ($MemMatrix$) as shown from lines 1 to 8.

If the list is not empty then we select the first vehicle from the list and check the bandwidth schedulability constraint for that vehicle as shown from lines 21 to 23. After that we check for the edge resource constraint with overlapping set of vehicles from lines 24 to 26. If the value of m^* is greater than the memory requirement of the vehicle, then we set the value of m^* to be exactly equal to the memory requirement of the vehicle as the quantum of chunk needs to be reduced down to the final requirement left of the vehicle as shown in lines 27 and 28. Then we add m^* to the total memory used at edge E_{index} of those overlapping sets in which vehicle v is present as shown in line 29. Further we increase the value of allocated memory to vehicle v at edge E_{index} by m^*, increase the value of used memory at edge E_{index} by m^* and decrease the memory requirement of vehicle v by m^* as shown from lines 30 to 32. If the memory available on edge E_{index} is 0, then we remove this edge from the $EdgeList$ as shown in lines 33 and 34. Then, we check if the memory requirement of vehicle v or the value of m^* becomes 0 then we remove the vehicle v from the $VehicleList$ of the edge E_{index} as shown in line 35 and 36.

Finally, if the $EdgeList$ becomes empty, then it means that there are no more edges left to serve the vehicle requests and we terminate the algorithm as shown in lines

Algorithm 2: Calculating $m_{i,j}$ values for all the vehicles

$A \leftarrow$ List of memory used at all of the edges
$K \leftarrow$ List of $k_j = \frac{v_{i,j}}{L_j \times B_j}$ for each of the edge E_j
$AvlMem \leftarrow$ List of available memory for all of the edges
$Mem_{S_{k,j}} \leftarrow$ Total memory used by an overlapping set S_k at edge E_j
$M \leftarrow$ List of memory requirement of all the vehicles.
$EdgeList \leftarrow$ List of all the edges
$VehicleList \leftarrow$ A list of list which contains the vehicles passing through each edge
$MemMatrix \leftarrow$ Empty matrix of size $N \times M$ for $m_{i,j}$ values
while *true* **do**
  $m^* \leftarrow DATA\_QUANTUM$
  $min_{val} \leftarrow INT\_MAX$
  $index \leftarrow -1$
  for i *in* $EdgeList$ **do**
    $gradient \leftarrow \frac{\delta \times m^*}{K[i]} \times (2 + \frac{2 \times A[i]}{K[i]} + \frac{m^*}{K[i]})$
    if $min_{val} > gradient$ **then**
      $min_{val} = gradient$
      $index = i$
  if $len(VehicleList[index]) == 0$ *and* $index$ *in* $EdgeList$ **then**
    $Edgelist.remove(index)$
    continue
  $v \leftarrow VehicleList[index][0]$
  if $MemMatrix[v][index] + m^* > D^{min,data}_{v,index}$ **then**
    $m^* = D^{min,data}_{v,index} - MemMatrix[v][index]$
  $Maxmem \leftarrow max(Mem_{S_{1,index}}, Mem_{S_{2,index}}, ..., Mem_{S_{n,index}})$
  if $Maxmem + m^* > AvlMem[index]$ **then**
    $m^* = 0$
  if $m^* > M[v]$ **then**
    $m^* = M[v]$
  Add m^* to $Mem_{S1,index}$, $Mem_{S2,index}$, ..., $Mem_{Sn,index}$ if the vehicle v is present in the overlapping sets S_1,S_2, ..., S_n.
  $MemMatrix[v][index] += m^*$
  $A[index] += m^*$
  $M[v] = M[v] - m^*$
  if $AvlMem[index] - (Maxmem + m^*) == 0$ **then**
    $EdgeList.remove(index)$
  if $M[v] == 0$ *or* $m^* == 0$ **then**
    $VehicleList[index].remove(v)$
  if $EdgeList.IsEmpty()$ **then**
    break

37 and 38. This algorithm calculates the value of data chunks $m_{i,j}$ for all the vehicles passing through the edges. 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.

Complexity of the Heuristic: If the average memory requirement of the vehicles is M_i^{avg} then the complexity of the algorithm would be $O(N_v \times N_e \times \frac{M_i^{avg}}{m^*})$. This is because there are N_v vehicles and for each vehicle there would $N_e \times \frac{M_i^{avg}}{m^*}$ iterations to fulfill its demand. Therefore, the total complexity becomes $O(N_v \times N_e \times \frac{M_i^{avg}}{m^*})$.

V. Experimental Setup and Results

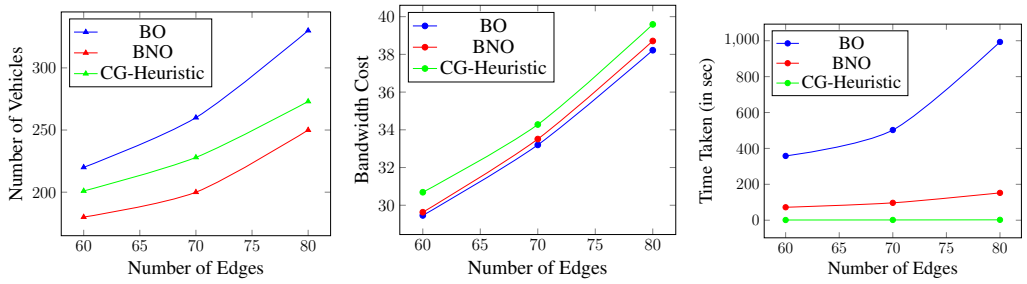
In this section, we present the experimental setup for the optimization framework and the results obtained. First we conducted our experiments on a synthetic dataset and then we used the dataset from a real case scenario of Luxembourg city that was simulated using SUMO in [19]. We conducted our experiments in Matlab 2020a and the optimization solvers were SDPT3 and gurobi. The experiments for run time complexity were conducted on a Intel(R) Xeon(R) CPU E5-2640 based server, which has 20 cores with each running at 2.6GHz. The experimental evaluation will compare the results obtained using 3 different approaches. They are:

- **Base no overlap** (BNO): The base no overlap is the optimal case considering no overlap of vehicles at an edge (proposed in [7]).
- **Base overlap** (BO): The base overlap is the optimal case proposed in this paper considering overlap of vehicles.
- **Cost gradient-based heuristic** (CG-Heuristic): This heuristic is the main proposed technique in this paper (as per Algorithm 2) considering overlaps. For CG-Heuristic we chose the value of $m^* = 0.25$ and our extensive experiments with m* (defined in Section IV equation (13)) confirms the validity of our choice and can be seen with more results in supplementary material.

Hereafter, we refer to the above 3 approaches based on their abbreviations BNO, BO and CG-Heuristic.

A. List of Parameters

1) We used five different values for the number of edges, M. These are 36(6×6), 49(7×7), 64(8×8), 81(9×9) and 100(10×10)
2) Based on the number of edges we varied the number of vehicles N. The number of simulated vehicles were varied from 100 to 350 for different number of edges.
3) The route of the vehicles was generated randomly such that the vehicles passes through different edges.
4) The value of vehicle density at jam(k_j^{jam}) was taken as 50 and the actual vehicle density(k_j) was varied from 25 to 40 in steps of 5 as given in [20].



(a) (b) (c)

Figure 1: Comparison of BNO, BO and CG-Heuristic for Real dataset. (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

5) Memory requirement(M_i) of each vehicle was generated randomly between 60 Mbits to 80 Mbits using uniform distribution.
6) Coverage distance(L_j) of the edges was randomly generated between 0.6 miles to 1.6 miles as provided in [21].
7) Memory capacity(M_j) of the edges was randomly generated between 400 Mbits to 500 Mbits and the occupied memory(M_j^{occ}) was randomly generated between 0 Mbits to 150 Mbits as given in [22].
8) The computation capacity of the edges(P_j in number of VMs) was randomly generated between 24 and 40 and occupied computation capacity was randomly generated between 1 and 3.
9) The vehicle's processing requirement(p_i) was randomly generated between 1 and 10 and t_{pi} was randomly generated between 1 and 10 sec.
10) The data that the vehicle sends(d_i) to the edge and the data that it receives(r_i) from the edge was randomly generated between 1 Mbits to 15 Mbits.
11) The bandwidth of the edges(B_j) was randomly generated between 8 Mbps to 15 Mbps.
12) Free flowing velocity(v_j^f) of the vehicles was randomly generated between 50mph to 70mph as taken from [20].

B. Experiments and results using Synthetic Dataset

In this section, we discuss the experiment and results for the synthetic dataset. First we present the dataset and then we discuss the results of the experiments. We modelled our edge network in square grids of different sizes. Above V-A is the list for values of parameters that we generated synthetically.

We now present the various results that we obtained from the experiments and our inferences from them.

*1) **Maximum number of vehicles served with varying number of edges**:* While comparing the maximum number of vehicles that can be served for a given number of edges by the 3 approaches, a general trend is observed regarding the performance of the CG-Heuristic as can be seen in Fig. 1 (a). The CG-Heuristic is able to serve more vehicles than the BNO, but can serve lesser vehicles than the BO. This is understandable because BO is the optimal approach and CG-Heuristic is the heuristic approach. The difference in the number of vehicles served generally increases with the increase in the number of edges. We observed that the highest percentage increase in the number of vehicles using the BO approach is 63.15% (for data delivery) and 50% (for service delivery) as compared to BNO.

*2) **Total Bandwidth Cost**:* From Fig. 1 (b) it is observable that the CG-Heuristic has a slightly higher bandwidth cost compared to both the BO and BNO (3.1 units more than BO and 1.8 units more than BNO).

*3) **Run Time Complexity**:* A significant decrease in the run time taken for completion of the algorithm can be observed while comparing the CG-Heuristic with BO and BNO as seen in Fig. 1 (c). There is a very small increase in the time taken by the CG-Heuristic on increasing the number of edges. We observed that CG-Heuristic runs on an average 60 times faster than BNO and 270 times faster than BO.

C. Experiments on Real Case Scenario

In this part, we present the experiment and results for our optimization frameworks that we applied on a real case scenario. For the real case scenario we used the dataset of Luxembourg city that was simulated using SUMO in [19]. 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 60 to 80 in steps of 10. The other parameters like the coverage distances(L_j), memory capacity(M_j), processing capacity(P_j), bandwidth(B_j) etc were generated similarly as discussed in subsection V-B. The values fall in the range provided by these works [20], [21] and [22].

We now present the results for the real case scenario.

*1) **Maximum number of vehicles served with varying number of edges**:* For the real case scenario, we can see in Fig. 1(a) that the maximum number of vehicles served are greater for BO than both BNO and CG-Heuristic. This

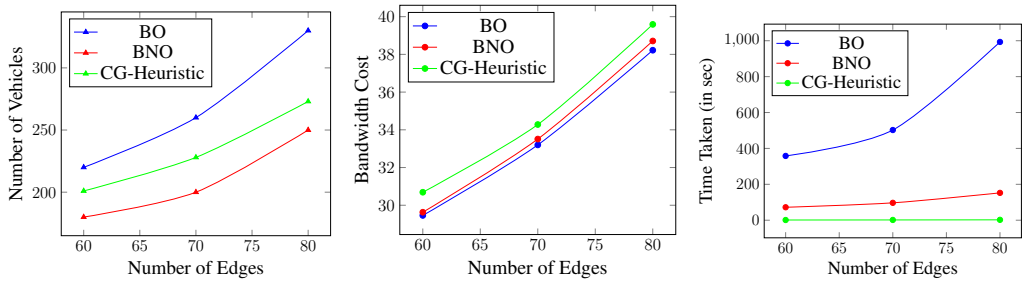


Figure 2: Comparison of BNO, BO and CG-Heuristic for Real dataset. (a) Maximum number of vehicles serviced successfully for a given number of edges, (b) Total bandwidth cost for a given number of edges and (c) Run-Time Complexity

is understandable because BO is our proposed optimization framework for resource allocation compared to CG-Heuristic and considers vehicle overlaps that BNO does not. BO can serve an average of 28.57% more vehicles than BNO and 15.38% more vehicles than CG-Heuristic, which also tells us that CG-Heuristic performs better than the BNO approach by considering vehicle overlaps.

*2) **Total Bandwidth Cost**:* In Fig. 1(b), we can observe that CG-Heuristic has a slightly higher bandwidth cost compared to both BO and BNO (1.37 units more than BO and 1.06 units more than BNO). The vehicle data assignment to nodes using CG-Heuristic is not optimal as it is a fast heuristic, but still not far away from optimal as we use the global cost gradient for allocation

*3) **Run Time Complexity**:* The plot for BO increases rapidly and is higher than BNO and CG-Heuristic as seen in Fig. 1(c). The run time complexity for CG-Heuristic increases at a much lower rate than the other two approaches. We observed that CG-Heuristic runs on an average 65 times faster than BNO and 380 times faster than BO.

Here, the notable point is that the CG-Heuristic can serve more vehicles than BNO in much lesser time with slight increase in bandwidth cost. In addition, it can serve a comparable number of vehicles as BO approach with far lesser run-time complexity and minimal increase in bandwidth cost.

VI. Conclusion

In this work, we proposed an improved resource allocation optimization framework for data delivery to connected vehicles via edge nodes. Specifically, a realistic scenario is assumed where all vehicles passing through an edge may not overlap at the edge. We proposed algorithms to find overlapping sets of vehicles and integrated the overlapping sets into the optimization framework. A global edge bandwidth cost gradient-based heuristic was proposed in order to address the time complexity of the optimization framework. Finally, we conducted extensive experimentation to demonstrate several advantages of our proposed optimization framework and heuristic solution in comparison to an earlier work. In future, we would like to extend this work by considering other factors in the objective function such as delivery time.

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] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *IEEE Globecom Workshops*, 2015, pp. 1–6.

[9] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[10] S. S. Tanzil, O. N. Gharehshiran, and V. Krishnamurthy, "Femto-cloud formation: A coalitional game-theoretic approach," in *IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[11] J. Oueis, E. C. Strinati, and S. Barbarossa, "Small cell clustering for efficient distributed cloud computing," in *25th IEEE International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, 2014, pp. 1474–1479.

[12] 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.

[13] 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 (ICDCS)*, 2017, pp. 1281–1290.

[14] 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.

[15] B. Greenshields, J. Bibbins, W. Channing, and H. Miller, "A study of traffic capacity," vol. 14, 01 1935.

[16] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018, pp. 858–866.

[17] L. H. Vu, B. N. Passow, D. Paluszczyszyn, L. Deka, and E. Goodyer, "Estimation of travel times for minor roads in urban areas using sparse travel time data," *IEEE Intelligent Transportation Systems Magazine*, 2020.

[18] "NYC Open Data: Data Set by Department of Transportation," https://data.cityofnewyork.us/browse?Dataset-Information_Agency=Department+of+Transportation+\%28DOT\%29.

[19] 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.

[20] 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.

[21] P. Salvo, F. Cuomo, A. Baiocchi, and A. Bragagnini, "Road side unit coverage extension for data dissemination in vanets," in *9th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*, 2012, pp. 47–50.

[22] B. KARA and B. ÖZGÖVDE, "Effect of rsu placement on autonomous vehicle v2i scenarios," *Balkan Journal of Electrical and Computer Engineering*, 06 2020.