Deadline-Aware Task Offloading for Vehicular Edge Computing Networks using Traffic Lights Data

PRATHAM OZA, Virginia Tech, USA
NATHANIEL HUDSON, University of Chicago, USA; Argonne National Laboratory, USA
THIDAPAT CHANTEM, Virginia Tech, USA
HANA KHAMFROUSH, University of Kentucky, USA

As vehicles become increasingly automated, novel vehicular applications emerge to enhance the safety and security of the vehicles and improve user experience. This brings ever-increasing data and resource requirements for timely computation on the vehicle’s on-board computing systems. To alleviate these demands, prior work propose deploying vehicular edge computing (VEC) resources on the road-side units (RSUs) in the traffic infrastructure to which the vehicles can communicate and offload compute intensive tasks. Due to limited communication range of these RSUs, the communication link between the vehicles and the RSUs and therefore the response times of the offloaded applications are significantly impacted by vehicle’s mobility through road traffic. Existing task offloading strategies do not consider the influence of traffic lights on vehicular mobility while offloading workloads on the RSUs, and thereby cause deadline misses and quality-of-service (QoS) reduction for the offloaded tasks. In this paper, we present a novel task model that captures time and location-specific requirements for vehicular applications. We then present a deadline-based strategy that incorporates traffic light data to opportunistically offload tasks. Our approach allows up to 33% more tasks to be offloaded onto the RSUs, compared to existing work, without causing any deadline misses and thereby maximizing the resource utilization on the RSUs.

CCS Concepts: • Networks → Cloud computing; Cyber-physical networks; • Computer systems organization → Real-time system architecture.

Additional Key Words and Phrases: edge computing, connected traffic infrastructure, task offloading

1 INTRODUCTION

The presence of smart vehicles with varying levels of connectivity and autonomy is rapidly becoming prominent on today’s roads. Vehicles are now equipped with advanced sensing, communication, and computation capabilities as they make their transition from being human-driven (Level 0) to fully autonomous (Level 5) as per vehicular automation standards [37]. These capabilities will enable advanced driving assistance (ADAS) applications that will help vehicles make precise driving maneuvers and enhance overall safety, security, and user experience in complex road traffic scenarios [43]. Examples of such ADAS applications include (but are not limited to) lane keeping assist (LKA), lane change assist (LCA), [8] and cooperative advanced cruise control (CACC) [31]. While the vehicles have dedicated on-board sensing and processing to support safety-critical driving applications, it is also necessary to support other data and computationally intensive vehicular applications that enhance the passengers’ experience. Applications involving media-rich infotainment, augmented reality (AR)-based media
streaming, collaborative data aggregation and interactive user applications, etc. are resource-intensive [17] and have timeliness and quality-of-service (QoS) requirements, even though they may not be safety-critical applications.

Cloud servers provide additional storage and computational resources to satisfy these increasing demands, however, the latency of wireless channels for such long-range communications tends to be a bottleneck [24]. The upcoming vehicular edge computing (VEC) paradigm provides an attractive alternative to cloud computing solutions. Under the VEC framework, edge servers are deployed physically nearby to the vehicles, and provide additional compute and storage resources. Additionally, VEC relies on network technologies such as dedicated short-range communication (DSRC), and 5G-based cellular vehicle-to-everything (CV2X) connectivity, etc., that enable high throughput short distance communication between the vehicles and the edge servers deployed along the roadways [1]. The data and connectivity among the entities within the traffic ecosystem such as road-side sensors, vehicles, and traffic lights can now be leveraged to implement novel algorithms that enhance the overall QoS and performance of various automotive applications [16, 47]. Such localized data can be made available on an edge platform and, therefore, VEC is a promising approach to cater to the data and computationally-intensive vehicular tasks with soft real-time requirements.

In a typical VEC network, road-side units (RSUs) (e.g., traffic lights, cameras, detectors) are equipped with additional storage and compute resources. Vehicles can communicate and offload workloads onto the edge-enabled RSUs via V2X communication, with significantly reduced communication latency, when they are within the communication range of a nearby RSU. Recent studies are therefore focused on finding appropriate RSU resources to opportunistically offload tasks to meet the timeliness and resource requirements as the vehicles drive in and out of the RSUs’ communication range [19, 48, 52, 53]. Figure 1 depicts a typical task offloading scenario such that the timeliness requirements of the task are met. As per the figure, as the vehicle travels along a road link, it communicates with the nearby RSUs to upload a task, which is then processed at one of the RSUs and the results are downloaded back to the vehicle. As the vehicle travels, the traffic light status changes which impacts the speed of the vehicle and hence the location at which the task is uploaded to and downloaded from the RSUs. The timeline graph in Figure 1 depicts the total offload time including processing time and communication delays, details of which are discussed in later sections.

A vehicle can successfully communicate, exchange data and offload tasks with an RSU while it is within the communication range of the RSU. In dense traffic networks, the time during which the vehicles can maintain communication with a nearby RSU changes frequently. A vehicle cruising through a green light may spend very little time within the range of one RSU, but can communicate for a significantly longer time with another RSU while waiting at the traffic light if it is red. Therefore, traffic conditions and vehicular mobility significantly impact the communication link between the vehicles and the RSUs, and therefore the choice of RSUs to offload the tasks.

In addition to influencing the communication between the vehicles and the RSUs, vehicular mobility and traffic conditions also affect the requirements of many novel vehicular applications [11, 15]. For example, consider a dynamic routing application on the vehicle that finds an optimized route to a preferred destination based on collaborative data from surrounding vehicles. If an optimal route for the upcoming intersection is to be calculated, and the vehicle is waiting at a red traffic light, the routing task need not be completed until the light turns green and the vehicle starts moving to cross the intersection. Traditional VEC task models do not account for such tasks whose deadlines depend on the traffic flow and traffic lights data when the task arrives. To support emerging applications [47], we propose a comprehensive task model comprising of some tasks that have static deadlines as in traditional task models, while other tasks have location (or distance)-specific deadlines along with the computation requirements. Figure 2 shows an example where the light timings influence the vehicular mobility characteristics, which in turn affects the communication times and tasks with dynamic deadlines. As shown, consider a vehicle traveling through a road link devoid of traffic, controlled by a traffic light. If the traffic light is...
green, it will spend similar amounts of time within the communication range of each RSUs along its way as it drives at a constant speed. However, if the traffic light is red, the same vehicle will spend a much longer time within the range of the RSUs closer to the traffic light as it slows down to come to a stop.

Existing offloading strategies select RSUs based on an underlying goal of minimizing task computation \cite{51}, resource utilization \cite{35}, bandwidth usage \cite{39}, or access costs \cite{27}. As vehicles enter a lane in a sequential manner and RSUs are only accessible once the vehicle is in the communication range, greedy approaches offload the tasks for the earlier arriving vehicles on the RSUs that are located closer to the them when the tasks arrive even if they have a later deadline. This blocks the later arriving tasks (and vehicles) from accessing these resources even if they have earlier deadlines. For example, in Figure 3, an earlier arriving vehicle ($V_i$, marked in red) ends up utilizing the first available resource ($RSU_1$) due to a greedy offloading mechanism even though its task, $\tau_i$, has a later deadline. When a later arriving vehicle ($V_{i+1}$ marked in blue), has a more urgent task ($\tau_{i+1}$) to offload, the resources are blocked by $\tau_i$ leading to $\tau_{i+1}$ missing its deadline. Therefore, existing approaches fail to maximize the number of tasks that can be offloaded onto the edge.

By offloading as many resource intensive tasks with less stringent timing requirements onto the RSUs, the vehicles not only maximize the resource utilization at the RSUs, but also allow the on-board processors to prioritize on time- and safety-critical applications. Since VEC tasks have soft real-time requirements, the system accrues minimal benefit in completing tasks well before the deadline. Missing the deadline however, leads to reduced QoS. By using a deadline-based task offloading strategy, where we offload tasks such that they complete closer to the deadline, we not only ensure that the tasks that are offloaded meet their deadlines, but by using our approach, more tasks can be offloaded onto the edge without blocking resources for later arriving tasks.

To summarize, our contributions are as follows:

- To combat the shortcomings of the existing VEC models that only consider tasks with static deadlines, we present a comprehensive model that consists of tasks with fixed time-based as well as dynamic distance-based deadlines based on traffic flow, traffic lights and the location requirements of the task,
- We incorporate the traffic lights timing data to calculate (i) the mobility characteristics and travel time for the vehicles and (ii) timing requirements for tasks with dynamic distance-based deadlines,
- We propose a task offloading strategy that allocates RSUs to complete tasks as close to the deadlines as possible, thereby maximizing the tasks offloaded without violating the task deadlines.

Fig. 2. Influence of traffic lights on dwell times: Vehicle spends significantly longer time within the range of RSU_m when the light is red (above) in comparison to a situation when the light is green (below). \( \delta_{ij} \) is the dwell time of vehicle \( V_i \) in the communication range of RSU_j.

- We evaluate our proposed approach against existing offloading strategies using VISSIM [10], a microscopic traffic simulator.

### 1.1 Paper Structure

The remainder of this paper is divided into the following sections. Section 2 provides a survey of prior work in task offloading techniques on vehicular edge computing platform. Section 3 presents our system model with details on the traffic infrastructure, task model and assumptions in our communication framework. In Section 4, we provide details on our proposed deadline-based task offloading approach and the algorithm used to find appropriate computing resources to process the tasks. Consequently, in Section 5, we analyze and evaluate the performance of our proposed approach when compared to other existing task offloading algorithms. Finally, Section 6 provides concluding remarks and insights into future work.

### 2 RELATED WORK

With increasing demands of performance, data and compute hungry processes on mobile devices induced breakthroughs in small-cell networks, multi-antenna, and millimeter-wave communications to provide highly reliable and gigabit wireless access to next generation systems [45]. Due to these advancements it was now possible to utilize external network-based storage and compute resources to satisfy the demands. Mobile cloud computing (MCC) was a promising solution for offloading workloads from a plethora of connected devices. Many existing work have focused on tackling various connectivity, resource, mobility, network latency constraints to optimize the utilization of MCC [4, 22, 38].

As connectivity penetrated vehicles and traffic infrastructure, vehicular cloud computing (VCC) became prevalent to enhance safety, security and user experience for vehicular application-specific workloads [26]. Various architectures were proposed to enable cloud-based computation for vehicular applications [3, 20]. Efforts have also been made to make vehicular cloud computing more efficient via optimized resource allocation techniques [49], more secure [41] and more energy-efficient [44] to allow its widespread adoption. However,
many emerging vehicular applications require large amounts of data to be processed with strict response times and network bandwidth constraints. To meet these demands, vehicular edge computing (VEC) paradigm was introduced [42], where storage and compute nodes were brought closer to the proximity of the end user and thereby greatly reducing the network latency [7] and energy consumption [36].

Recent studies have proposed various task offloading mechanisms to offload more tasks from the vehicle onto the edge platform [13, 42]. In addition to offloading tasks, many researchers have also provided resource allocation mechanisms to maximize the utilization of the limited compute resources available at the edge [9, 50]. In [42], authors propose an offloading mechanism for complex vehicular tasks with dependencies. In [27] a matching-based approach was proposed to jointly offload vehicular tasks and allocate resources on the edge. While most of these techniques focus on offloading tasks and utilizing the edge resources deployed along the road-side units of the traffic infrastructure, many researchers also propose a vehicular fog computing (VFC) architecture where the resources available on other vehicles driving in the vicinity or parked vehicles are utilized to meet the demands. Solutions proposed for utilizing the VFC framework are out of scope for our work, interested readers can refer to existing literature [23, 28]. None of these proposed solutions, however, consider the mobility constraints in a vehicular environment.

Vehicular mobility can significantly impact the resource utilization as well as task allocation onto the edge. Novel task offloading and resource allocation mechanisms are proposed that incorporate a driving model for vehicles [2, 5, 53]. However, the driving models used in these work are simplistic and do not accurately represent vehicular mobility in urban driving conditions.

In a highly dynamic urban environment, vehicles' mobility is heavily affected by speed limits, traffic density, traffic lights etc. A holistic solution is required that is applicable in varying traffic conditions by incorporating mobility constraints to accurately offload tasks and allocate resources. To that end, in our proposed work, we overcome the shortcomings of existing task offloading strategies by using a robust driving model that utilizes traffic light timings to estimate vehicle's mobility. Additionally, we propose a task model that closely represents vehicular workloads and provide a task offloading strategy that meets the timeliness constraints of the tasks.

3 SYSTEM MODEL
We now look at the VEC system model and its individual components pertaining to traffic infrastructure, vehicular maneuverability, edge computing tasks, and the VEC communication framework.

3.1 Traffic Infrastructure
We represent a single road link that connects two consecutive intersections as link \( L \). This link has length \( d_L \) with posted speed limit of \( v_{lim} \) for the traffic flowing along link \( L \) to approach an intersection within the urban traffic network. Link \( L \) can have one or more lanes \( \{ l_1, \cdots, l_k \} \), where each lane \( l_i \) is individually controlled using a traffic light \( (s_i) \) located downstream of the traffic flow. The traffic lights follow a cycle of green-yellow-red lights and are enabled with edge connectivity, resembling a traffic light with fixed cycle time or a state-of-the-art traffic controller where timings are known for a fixed prediction horizon [32]. Additionally, the traffic lights have connectivity to broadcast the traffic light data to the connected infrastructure. This traffic lights data consists of its current state \( (\psi_{s_i} = \{ \text{red, green, yellow} \}) \), remaining time in current state \( (t^r_{rem,i}) \), maximum red \( (t^r_{s_i}) \), green \( (t^g_{s_i}) \) and yellow phase time \( (t^y_{s_i}) \), and the cycle time \( (t^{cycle}_{s_i} = t^r_{s_i} + t^g_{s_i} + t^y_{s_i}) \).

3.2 VEC Task Model
As introduced in Section 1, existing task models only consider tasks with fixed time-based deadlines. In this work, in addition to tasks with static deadlines, we also introduce dynamic distance-based tasks that represent novel VEC applications and workloads. Such tasks have dynamic deadlines based on distance-specific requirements
specifying that the tasks need to be completed before the vehicle has travelled a certain distance from the time the task arrives. Based on traffic flow, lights status, and vehicular maneuvering characteristics, the time requirement for such tasks change dynamically.

We therefore consider a task model where a task can be either of the following:

1. **Regular task**: Similar to prior methodologies, such a task has a fixed deadline based on a time before which the task must be executed, or
2. **Distance-based task**: Such a task has a deadline based on a distance before which the task must be executed.

Any task \( t_i \) originating from vehicle \( V_i \) has an arrival time \( (a_i) \), upload size \( (\sigma_{i}^{ul} \text{Mb}) \), computational requirement \( (C_i \text{ clock cycles}) \), download size \( (\sigma_{i}^{dl} \text{Mb}) \) and a deadline \( (d_i) \). Note that \( d_i = d_i^T \) or \( d_i = d_i^D \) depending on whether it is a distance-based task or a regular task, respectively. Therefore, \( t_i \) is represented as a 5-tuple \( t_i = (a_i, \sigma_{i}^{ul}, C_i, \sigma_{i}^{dl}, d_i) \).

A distance-based deadline can be translated into a regular time-based deadline based on the vehicle’s maneuvering capabilities. When the same distance-based task is spawned from two different vehicles, the task spawned from a slower moving vehicle that needs to go through multiple red lights will have a later time-based deadline compared to the task spawned from a faster moving vehicle that always gets a green signal at the traffic lights. Note that our model assumes that all offloaded tasks are independent tasks. In case of tasks with dependencies, our model can be utilized by decomposing inter-dependent tasks into multiple independent sub-tasks with local deadline assignment [30]. Further, the tasks are assumed to have negligible or bounded jitter and additional buffers can be added to task execution times to account for it. However, the analysis of our model with such tasks is out of scope for this discussion and is left for future work.

### 3.3 VEC Infrastructure

A typical VEC framework consists of (i) a **main base station** (MBS), (ii) **edge-enabled road-side units** (RSUs), and (iii) **connected vehicles** requiring the edge resources.

An MBS communicates wirelessly with all vehicles via a high throughput wired connectivity with edge-enabled RSUs, within its communication range. MBS has a larger communication range than other edge-enabled RSUs as they cover multiple intersections in a traffic network. Road-side units (RSUs) then comprise of infrastructure (e.g., traffic lights, sensors, cameras etc.) that are equipped with additional compute, storage, and communication resources. The RSUs communicate with the vehicles within its range via short-range wireless connectivity (for example, 5G-CV2X). All RSUs are connected via a wired backhaul network and are distributed along the road-side such that the entire road link is covered without any blind spots. Additionally, in our model, the RSUs do not share the processing resources and therefore tasks cannot be partially executed on different RSUs. Finally, the vehicles requiring edge resources communicate with the MBS and the RSUs to offload VEC tasks. A detailed description of such a framework can be found in [26].

**Edge Infrastructure**: The road link \( L \) will have RSUs deployed alongside to cover the entirety of the link \( L \), including all of its lanes. Further, the RSUs are located such that there is no overlap in the communication radius of two consequent RSUs. Any \( i^{th} \) RSU along the link \( L \) has a fixed communication radius \( (r_i) \), number of parallel processors \( (p_i) \) and a clock speed \( (z_i) \). The RSUs have strict power consumption requirements as they are deployed at remote locations due to which the transmission power is limited [21]. The communication radius is therefore determined using Friis’ equation [40] to ensure a stable link between the vehicles and the RSU. The number of processors and the clock speeds of each processor at an RSU are also chosen based on the power consumption requirements and the edge resource demands. Optimal placement of RSUs and edge services is an ongoing research problem [18], and is out of scope for this work. In short, our system considers \( m \) RSUs (i.e., \( RSU_1, \cdots, RSU_m \)), that are deployed alongside link \( L \) of length \( d_L \) such that, \( \sum_{i=1}^{m} 2r_i = d_L \) (since no overlap in RSU range) where \( r_i \) is the communication range radius of RSU \( i \).
**Channel Bandwidth and Connectivity:** The VEC framework consists of (i) a wired optical fiber-based backhaul network connecting all the RSUs and the MBS, (ii) low latency short-range wireless network between the vehicles and the RSUs, and (iii) a wireless long-range connectivity between the vehicles and the MBS.

Based on the upload ($\sigma_{ul}^j$) and the download ($\sigma_{dl}^j$) size of the task, there is a corresponding upload ($t_{ul}^j$) and download ($t_{dl}^j$) time based on RSU $j$’s channel characteristics. For simplicity, we consider that all RSUs have similar noise conditions and transmit power, and hence a fixed and equal $\eta$. Additionally, we assume that the transmit power is adjusted such that the SNR remains constant within the short range of communication of an RSU. However, in a more comprehensive setting that is significantly impacted by noise with varying channel conditions, existing formulations [34] can be used to determine the data transmission rate $\eta$ between the communicating entities.

Alternately, the wired backhaul network is a high throughput channel with minimal latency. Due to high bit rates, for smaller workloads, the relay time ($\rho_{jk}$) between two RSUs, $RSU_j$ and $RSU_k$, depends on the distance between the RSUs and can be defined as,

$$\rho_{jk} = \eta_{bh} \cdot \epsilon_{bh}(k-j), \forall j, k \in [1, m], j < k,$$

where $\epsilon_{bh}$ is the minimum transmission relay time between two consecutive RSUs [14] and $\eta_{bh}$ is the data rate for backhaul network.

### 3.4 Vehicle Maneuver Model

We consider that all vehicles within our system are fully connected and have automation capabilities of level 3 and beyond as per automation standards [37]. Such vehicles can drive and perform maneuvers in an automated manner with some (level 3) to no (level 5) intervention from the human driver. Such vehicles have already started penetrating today’s roads [12] and have an increased demand for edge resources as they need the on-board processors for performing real-time driving maneuvers. Such vehicles therefore deploy an advanced driving assistance and maneuvering model to establish real-time control of the vehicle and navigate it through the traffic network. The vehicles also broadcast body safety messages (BSMs) to the MBS and other infrastructure in the vicinity. BSMs are safety messages that the connected vehicles broadcast periodically to inform the infrastructure of its driving characteristics.

We denote the $i^{th}$ vehicle in our system by $V_i$. In our system, $V_i$ will enter a lane within $L$ driving at a speed of $v_i$ and perform one of the following driving maneuvers at any point in time:

1. **Constant speed:** $V_i$ maintains a constant speed of (i) $v_i = v_{lim}$ if there is no leading vehicle in front, (ii) $v_i = v_{i-1}$ in presence of a leading vehicle $V_{i-1}$, or (iii) $v_i = 0$ if the $V_i$ is stationary at the traffic light.
2. **Comfortable acceleration:** $V_i$ may need to accelerate due to a light turning green or a leading vehicle accelerating, and it does so at a constant deceleration rate of $a$.
3. **Comfortable deceleration:** $V_i$ may need to slow down due to a red light or a slowly moving leading vehicle and it does so at a constant deceleration rate of $-b$.

Similar driving maneuvering mechanisms were used in closely related work [27, 53] for task assignments in urban scenario, but without considering traffic lights.

The acceleration and deceleration rates are chosen with the comfort of the passengers within the vehicles as well as the surrounding pedestrians and vehicles in mind [6]. The vehicles also maintain a safe distance of $x_{safe} = h v_i + x_{safe}$ from their leading vehicle using cooperative adaptive cruise control (CACC) techniques where, $h$ denotes a constant time headway, $v_i$ is the driving speed of $V_i$ and $x_{safe}$ is the minimum safety gap at standstill [46]. For consistency, we assume that all vehicles use the same maneuvering model to allow for a realistic simulation setup. However, it should be noted that our proposed offloading strategy is model-agnostic.
Upon entering the link $L$, a vehicle communicates its location and speed via BSMs to the MBS and also requests for RSU allocation to upload, compute and download a task. In our model, we consider worst-case task offloading demands where every vehicle, upon entering a link, will always have a task to offload with varying timing, computation and location requirements.

4 DEADLINE-BASED TASK OFFLOADING

This section describes our proposed task offloading strategy for VEC systems. To do so, we first provide a highlight into the VEC task offloading process.

4.1 VEC Task Offloading

Based on the system model described so far, a typical task offloading process from a vehicle onto the edge is as follows:

1. A vehicle requests the main base station (MBS) to offload and compute any VEC tasks onto the RSU resources.
2. The offloading request also consists of the task requirements such as the arrival time, computation requirement, deadline, upload and download size of the data, etc. which is communicated via the on-board cellular network capabilities.
3. Optionally, the MBS checks if the task can be completed before its deadline if offloaded on the processing units available at the RSUs, in a deadline-aware approach.
4. If the task requirements are feasible, the MBS deploys a task offloading strategy to assign the requesting vehicle with RSUs where the task can be uploaded, computed and downloaded and accordingly reserves processing units to perform the computations.
5. Upon entering the communication range of the assigned RSUs, the vehicle communicates with the RSUs using short-range low latency communication channels to upload, and subsequently download the task and its results back onto the vehicle.

Therefore, the total completion time of a task (as shown in Figure 1) depends on (i) task upload time ($\tau^u_i$), (ii) task relay time ($\rho^r_i$) i.e., the time taken to relay the task data from the RSU where the task was uploaded to the RSU where the task is to be computed, (iii) task compute time ($\tau^c_i$), (iv) task relay time ($\rho^rd_j$) from RSU where the task is computed to the RSU from where the task will be downloaded, (v) wait time ($\tau^w_i$) for the vehicle to enter the communication radius of the RSU where the task is to be downloaded from, and finally, (vi) task download time ($\tau^d_j$). Figure 1 depicts a typical offloading scenario of task $\tau_i$ with an arrival time of $a_i$ and a deadline of $d_i$.

Selection of RSUs and Dwell Time Estimation: The total completion time of the task must not exceed its deadline to ensure that a task meets its deadline. The MBS therefore must select appropriate RSUs to upload ($RSU_u$), compute ($RSU_c$) and download ($RSU_d$) the task, such that the total completion time meets its deadline constraint. To avoid communication interrupts and retries, the vehicle requesting the VEC resource must remain within the range of the RSUs, $RSU_u$ and $RSU_d$, for a duration that is at least equal to the upload and download time of the task. This time duration is known as dwell time that indicates the approximate amount of time that a vehicle spends within the communication range of each RSU. Dwell time for vehicle $V_i$ within the communication range of $RSU_j$ is denoted by $\delta_i$, $\forall j = 1, \ldots, m$. The MBS derives the dwell times based on the vehicle’s mobility characteristics.

We motivate our deadline-based offloading strategy where we offload tasks such that they complete as close to the deadline as possible, using the following example.

4.2 Motivating Example

Consider a VEC system described in Section 3. $m$ RSUs are deployed along the link $L$ of length $d_L$, out of which, $RSU_2, \ldots, RSU_{m-1}$ are occupied by other compute-intensive tasks. The only two resources available for offloading...
4.2.1 Ofloading with Existing Approach. As shown in Figure 3, an existing approach such as [27] offloads tasks to minimize the total completion time but does not account for traffic lights. Since the total completion time for \( t_i \) if uploaded, computed and downloaded at RSU\(_1\) is 3.4 seconds (as per task characteristics) which is less than \( \delta_{i1} \), RSU\(_1\) will be selected for upload, compute and download. When \( t_{i+1} \) arrives, the only available RSU to offload is RSU\(_m\). Offloading \( t_{i+1} \) onto RSU\(_m\) requires \( V_{i+1} \) to travel for 25 seconds which adds to the completion time of the task. This leads to a deadline miss for \( t_{i+1} \). Therefore, with an existing strategy, only \( t_i \) task will be offloaded onto the RSU.

4.2.2 Ofloading with Deadline-based Approach. When \( t_i \) arrives at \( t_0 \), \( \psi_{sa} = red \) and has a remaining time of 45 seconds. Since the travel time from RSU\(_1\) to RSU\(_m\) is 25 seconds, the vehicle \( V_i \) will travel through the link and wait at the traffic light until it turns green. Since \( t_i \) is a distance-based task with the deadline coinciding to when it crosses the link, and the vehicle will take 45 seconds (until the light turns green) to cross the link, the estimated deadline for the task \( t_i \) will be 45 seconds. As per our strategy, the task must be completed closer to the deadline. Therefore, RSU\(_m\) will be selected for offloading the task \( t_i \), even though uploading, computing and downloading \( t_i \) onto RSU\(_m\) incurs a higher completion time (28.4 seconds, including travel time). Note that \( t_i \) still meets its deadline. When \( t_{i+1} \) arrives at \( t_1 \), RSU\(_1\) will be available and hence \( t_{i+1} \) will be uploaded, computed and downloaded on RSU\(_1\). Therefore, with our deadline-based just-in-time task offloading strategy, we (i) maximize the number of tasks that can be offloaded onto the RSUs, and (ii) meet deadlines for all offloaded tasks.

In a typical urban traffic environment with multiple traffic intersections, traffic conditions evolve continuously. In a connected environment, real-time traffic data is made available to monitor and predict as vehicular traffic evolves. We have shown in Section 3 that in vehicular edge computing applications, tasks have a strong dependency...
on vehicle maneuvers. By offloading tasks closer to the deadline, it allows making scheduling decisions based on real-time traffic conditions which minimizes re-computation of tasks in case vehicle’s state changes after uploading the task. Additionally, since most VEC tasks utilize data from vehicles and its surroundings, by delaying its processing closer to the deadline, it enables the utilization of most recent data. Furthermore, prior work that utilize just-in-time scheduling in IoT applications have shown better bandwidth and resource utilization with increased QoS [29]. When network conditions are known, approaches with just-in-time scheduling have shown reduced deadline miss ratio and packet drop ratio [25]. Network routing and vehicular traffic routing draw many similarities in their functions [48], which motivates us to explore our proposed deadline-based task offloading approach.

4.3 Deadline-based Approach

Elaborating on the motivation, we propose a deadline-based task offloading strategy that (i) incorporates traffic lights data to estimate the dwell time for the vehicles within each RSU’s range, (ii) ensures that all task requirements, including distance-based deadlines are met for offloaded tasks, and (iii) offloads tasks such that they complete closer to the deadline and thereby maximizing the number of tasks being offloaded.

We breakdown the task offloading strategy deployed over the main base station (MBS) into three phases. (i) Estimating the mobility and timing characteristics of the vehicles, (ii) Calculating the dwell times based on the mobility characteristics, and (iii) Finding the appropriate RSUs to meet the task deadlines and offload tasks closer to the deadline.

4.3.1 Estimating Mobility Characteristics. We consider that link $L$ consists of a single lane controlled by a traffic light $s$, but our model can be extended to multiple lanes and traffic lights. The MBS acquires the traffic lights data consisting of its current state $(y_s)$, remaining time in current state $(t^{rem}_s)$ and maximum times for green, yellow and red phases, i.e., $t^g, t^y$ and $t^r$, respectively. Based on the driving model and its maneuvers, i.e., (i) constant speed, (ii) accelerating at the rate of $a m/s^2$, or (iii) decelerating at a rate of $-b m/s^2$, the MBS determines the trajectory of vehicle $V_i$ while driving through link $L$ of length $d_L$ with a speed limit of $v_{lim}$. The rate of change in speed is denoted by $\alpha \in \{a, -b\}$ based on the maneuver. To determine the distance and timings within each driving maneuver, the MBS utilizes commonly known Kinematics Equations 2, 3 and 4, where, $v_0, t_0$, and $x_0$ determine the initial speed, time and position at the beginning of a maneuver while $v_f, t_f$, and $x_f$ determine the
speed, time and position at the end of a maneuver.

\[
t_f = \frac{v_f - v_0}{\alpha},
\]

\[
x_f = v_0 t_0 - 0.5at_0^2,
\]

\[
v_f^2 = v_0^2 + 2ax_0
\]

Further, we determine the safe stopping distance \((x_i^{\text{stop}})\) which denotes the distance required for vehicle \(V_i\) to come to a complete stop from driving at the speed limit \(v_{\text{lim}}\) with a deceleration of \(-b\) (Equation 5). \(t_i^{\text{stop}}\) denotes the time spent in traveling \(x_i^{\text{stop}}\) distance (Equation 6).

\[
x_i^{\text{stop}} = \frac{0 - (v_{\text{lim}})^2}{(-2b)},
\]

\[
t_i^{\text{stop}} = \frac{0 - v_{\text{lim}}}{(-b)}.
\]

In a typical traffic infrastructure, the yellow light time \(t_y\) is tuned such that \(t_y = t_i^{\text{stop}}\) that allow vehicles to safely decelerate and come to a halt upon encountering a traffic light. Any vehicle which is more than \(x_i^{\text{stop}}\) distance away from the traffic light will consider the yellow phase as red and would decelerate to come to a stop. Alternately, vehicles which are less than \(x_i^{\text{stop}}\) distance away from the traffic light will perceive the yellow phase as a green light and would continue driving at their current speed, to avoid sudden uncomfortable deceleration and/or acceleration. Therefore, to avoid ambiguity in formulation, a yellow phase of the traffic light is considered a part of the green phase in our formulation.

A vehicle’s mobility in a single lane link will only be influenced by either (i) the traffic light or (ii) another leading vehicle in the lane. To simplify the formulation of the mobility characteristics we first consider the case when a vehicle enters an empty link with no vehicles in front and therefore its maneuvering decision is solely dependent on the traffic light timings.

**Case 1: A vehicle enters an empty link controlled by a traffic light.** Based on the traffic light status and timings acquired by the MBS upon the arrival of the vehicle \(V_i\), we define Property 1.

**Property 1 (Case 1).** A vehicle \(V_i\) entering an empty link \(L\) of length \(d_L\) with no vehicles in front that has a safe stopping distance and corresponding time of \(x_i^{\text{stop}}\) and \(t_i^{\text{stop}}\) respectively, will drive at the speed limit \(v_{\text{lim}}\) until the vehicle is \(x_i^{\text{stop}}\) distance and correspondingly \(t_i^{\text{stop}}\) time away from the traffic light, irrespective of the traffic light state and timings.

Based on Property 1, we calculate the distance \(x_i^{\text{const}}\) and the corresponding time \(t_i^{\text{const}}\) for which the vehicle drives at the speed limit in a link \(L\) of length \(d_L\) with traffic lights located at the end of the link, using Equation 7 and 8.

\[
x_i^{\text{const}} = d_L - x_i^{\text{stop}},
\]

\[
t_i^{\text{const}} = \frac{x_i^{\text{const}}}{v_{\text{lim}}},
\]

Based on the traffic light state and timings, the vehicle will **Case 1(a):** drive through the entire link at the speed limit, **Case 1(b):** drive at the speed limit until it encounters a red traffic light when it decelerates to a halt, or **Case 1(c):** drive at the speed limit until it encounters a red light to decelerate and accelerate again without coming to a halt because of the light changing to green. We now define the conditions for the traffic light timings and the distance traveled based on which the vehicle will acquire one of the driving characteristics using Lemmas 4.1, 4.2 and 4.3. The proof for Lemma 4.1 and 4.2 can be derived in a similar manner as Lemma 4.3.

**Lemma 4.1 (Case 1(A): The vehicle drives through the link at the speed limit).** A vehicle \(V_i\) enters an empty link \(L\) of length \(d_L\) with a traffic light \(s\), and is bound by the acceleration and deceleration rate of \(a\) and \(-b\)
We describe the Algorithm 4.1 that the MBS will utilize to calculate the mobility characteristics of the vehicles (Equation 2). Additionally, the distance travelled during deceleration is given by (Equation 4). Now, the vehicle will continue accelerating until either (i) it reaches maximum speed limit, i.e., for communication radii and therefore, \( t_r \) and \( t_g \) will traverse the entire link \( L \) with a traffic light \( s \), and is bound by the acceleration and deceleration rate of \( a \) and \( -b \) respectively, will come to a complete stop at the traffic light, if one of the following conditions are satisfied: (i) \( \psi_s = \text{red} \) and \( t_g \leq t_r \), or (ii) \( \psi_s = \text{green} \) and \( t_g > t_r \). Here, \( \psi_s \) and \( t_g \) denote the light state and remaining time in current state of traffic light \( s \), and \( t_r \) is the time for which \( V_i \) drives at a constant speed before reaching the safe stopping distance from the traffic light.

**Lemma 4.2** (Case 1(b): the vehicle comes to a halt at the traffic light). A vehicle \( V_i \) enters an empty link \( L \) of length \( d_L \) with a traffic light \( s \), and is bound by the acceleration and deceleration rate of \( a \) and \(-b\) respectively, will come to a complete stop at the traffic light, if one of the following conditions are satisfied: (i) \( \psi_s = \text{red} \) and \( t_g > t_r + t_d \), or (ii) \( \psi_s = \text{green} \) and \( t_g \leq t_r \). Here, \( \psi_s \) and \( t_g \) denote the light state and remaining time in current state of traffic light \( s \), \( t_d \) denotes the duration needed for \( V_i \) to come to a complete stop at the traffic light, and \( t_r \) denotes the time for which \( V_i \) drives at a constant speed before decelerating at the traffic light.

**Lemma 4.3** (Case 1(c): vehicle initially decelerates, then accelerates without coming to a complete stop). A vehicle \( V_i \) enters an empty link \( L \) of length \( d_L \) with a traffic light \( s \), and is bound by the acceleration and deceleration rate of \( a \) and \(-b\) respectively, will initially decelerate for \( t_{i\text{dec}} \) time and then accelerate for \( t_{i\text{acc}} \) time to cross the traffic light without coming to a halt, if \( \psi_s = \text{red} \) and \( t_g < t_r < t_c + t_d \). Here, \( \psi_s \) and \( t_g \) denote the light state and remaining time in current state of traffic light \( s \), \( t_d \) denotes the duration needed for \( V_i \) to come to a complete stop at the traffic light, and \( t_c \) denotes the time for which \( V_i \) drives at a constant speed before decelerating at the traffic light.

**Proof:** Upon \( V_i \)'s arrival, \( \psi_s = \text{red} \), and \( t_g \) is the time after which the light turns green. If \( t_g > t_c + t_d \), then the vehicle starts to decelerate at the red light when it is \( x_{i\text{stop}} \) distance away from the traffic light. However, if \( t_g < t_c + t_d \), then the traffic light will turn green before the vehicle comes to a complete stop and starts accelerating. Therefore, the vehicle only decelerates for \( t_{i\text{dec}} = t_g - t_d \) time. As the light turns green, the vehicle starts accelerating at the rate of \( a \). The speed achieved after deceleration is denoted by \( v_{i\text{dec}} = \frac{v_{i\text{lim}} - v_{i\text{dec}}}{a} \) (Equation 2). Additionally, the distance traversed during deceleration is given by \( x_{i\text{dec}} = \frac{v_{i\text{dec}}^2}{2a} \) (Equation 4). Now, the vehicle will continue accelerating until either (i) it reaches maximum speed limit, i.e., for \( t_{i\text{acc}} = \frac{v_{i\text{lim}}}{a} \) duration or (ii) crosses the traffic light, i.e., upon traveling \( x_{i\text{acc}} = d_L - x_{i\text{dec}} \), achieving a speed of \( v_{i\text{acc}} = \sqrt{v_{i\text{dec}}^2 + 2a} \cdot x_{i\text{acc}} \) (Equation 4) and the corresponding time will be \( t_{i\text{acc}} = \frac{v_{i\text{acc}} - v_{i\text{dec}}}{a} \) (Equation 2), whichever shorter.

Therefore, vehicle \( V_i \) drives at the speed limit for \( t_{i\text{const}} \) time, then decelerating for \( t_{i\text{dec}} = t_g - t_{i\text{const}} \) time followed by acceleration for \( t_{i\text{acc}} = \min \left( \frac{v_{i\text{lim}} - v_{i\text{dec}}}{a}, \frac{v_{i\text{acc}} - v_{i\text{dec}}}{a} \right) \) time.

Before discussing Case 2, where the vehicle’s mobility is also influenced by leading vehicle in front, we discuss the algorithm used to estimate the dwell times in a scenario such as Case 1 where a vehicle enters an empty link. We describe the Algorithm 4.1 that the MBS will utilize to calculate the mobility characteristics of the vehicles and then assign the dwell times for each RSU. Additionally, the MBS also finds the time-based deadline for the traffic dependent tasks with location requirements, based on the derived mobility characteristics of the vehicles.

### 4.3.2 Dwell Time Assignment Algorithm

The MBS takes (i) traffic lights data from traffic light \( s \), i.e., current state (\( \psi_s \)), remaining time in current state (\( t_g \)), maximum red phase time (\( t_r \)), and maximum green phase time (\( t_g \)), (ii) vehicle and link data, i.e., speed limit (\( v_{i\text{lim}} \)) and length (\( d_L \)) of link \( L \), index \( i \) for vehicle \( V_i \) and location-based deadline \( d_t \) depending on the task to be offloaded by \( V_i \), and (iii) RSU information including the number of RSUs (\( m \)) and the communication radii of all \( m \) RSUs (\( R_j, j = 1, \ldots, m \)). Since we consider all RSUs have equal communication radii and therefore, \( R_j = R, \forall j = 1, \ldots, m \). Using Algorithm 4.1, the MBS then calculates the dwell times \( \delta_{ij} \), \( \forall j = 1, \ldots, m \) for vehicle \( V_i \) in all \( m \) RSUs. Additionally, it also finds the time at which the location-based
deadline will be met \( (d_f^i) \), the RSU \( (RSU_{end}) \) where the vehicle will be located when its task meets the deadline, and the time duration \( t_{i_{end}} \) within the range of \( RSU_{end} \) left before the deadline is met.

When vehicle \( V_i \) enters the link \( L \) and requests to offload \( t_j \), the MBS receives the relevant data from the infrastructure \( (\text{Lines 1-3}) \). The MBS is aware of the driving controller on \( V_j \) and hence \( t_{\text{step}} \) which determines the time granularity at which the control decisions are taken by the driving controllers \( (\text{Line 4}) \). The safe stopping distance, the distance driven at speed limit, and their corresponding times are calculated using Equations 5, 7, 6, and 8 \( (\text{Line 6}) \). Then, until the vehicle is estimated to cross the traffic light \( (\text{Line 7}) \), the MBS, using the driving maneuver model decides whether to perform a constant speed, deceleration or acceleration maneuver \( (\text{Lines 8-15}) \), and then calculates the corresponding acceleration, speed, and position for the next time step \( (\text{Line 16}) \). During the entire calculation process, the MBS also accumulates the time \( T_j \), and assigns the dwell time \( \delta_{ij} \) for \( RSU_j \) when the calculated position exceeds the coverage boundary of an RSU, after which \( T_j \) is reset \( (\text{Lines 17-19}) \). Further, during the calculation process, when the MBS encounters that the estimated position equals the distance-based deadline or time duration equals the time-based deadline, the corresponding time \( T_j \) is assigned as the time-based deadline of the task, and the corresponding \( RSU_{end} \) and the duration within the range of \( RSU_{end} \) before the deadline is met are noted for RSU allocation \((\text{Lines 20 and 21})\). As the calculated driving time and distance satisfies Property 1 \((\text{Line 22})\), it checks the traffic light state and timings to find which conditions are met as per Lemmas 4.1, 4.2, and 4.3 \((\text{Lines 23-34})\). Based on the condition, the MBS then selects the appropriate driving maneuver \( (flag_{ij}) \) influenced by the traffic light and also sets the time check points, \( t_{i_{\text{stop}}}, t_{i_{\text{dec}}}, t_{i_{\text{acc}}} \) and \( t_{i_{\text{w}}} \) when the MBS must change the maneuvering calculations \((\text{Lines 35-43})\). The chosen maneuver \( flag_{ij} \) is then selected as an \textit{action} for the next iteration. Calculations for a vehicle \( V_i \) end when its calculated position exceeds the length of the link and the algorithm provides the dwell times of vehicle \( V_i \) in every RSU along the link as an output.

Now, let’s consider a more complex situation where the link is non-empty and there may be leading vehicles that will influence vehicle \( V_i \)’s mobility.

\textbf{Case 2: A vehicle enters a non-empty link with existing vehicle(s) and is controlled by a traffic light.} We now consider a situation where there will be existing vehicles in the link that vehicle \( V_i \) enters. We refer to a vehicle immediately in front of \( V_i \), i.e. \( V_{i-1} \) as \textit{lead vehicle}, and the vehicle under consideration, i.e., \( V_i \) as \textit{ego vehicle}. Based on the location of the lead vehicle \( V_{i-1} \), it may or may not affect the mobility characteristics of the ego vehicle \( V_i \). Further, we highlight that an ego vehicle maintains a safety distance of \( x_{i,\text{safe}} = \text{hv}_i + x_{\text{safe}} \) with its leading vehicle to enable emergency braking and safe traffic movements, where \( v_i \) is the speed of the ego vehicle, \( k \) is a safety constant and \( x_{\text{safe}} \) is the minimum distance between two vehicles at standstill. We therefore define the following Property 2.

\textbf{Property 2 (Case 2).} An \textit{ego vehicle} \( V_i \) \textit{entering a non-empty link} \( L \) \textit{of length} \( d_L \) \textit{has atleast one vehicle} \( V_{i-1} \) \textit{between itself and the traffic lights at the end of the link} \( L \). \( V_i \) \textit{has a safe stopping distance and time of} \( x_{i,\text{stop}} \) \textit{and} \( t_{i,\text{stop}} \), \textit{respectively}, \textit{and a safety gap of} \( x_{i,\text{safe}} = \text{hv}_i + x_{\text{safe}} \) \textit{from the leading vehicle}. In this case, the ego vehicle will drive at a constant speed of \( v_{\text{lim}} \) until either of condition (i) the ego vehicle is \( x_{i,\text{safe}} \) distance behind the lead vehicle \( V_{i-1} \), or (ii) the ego vehicle is within \( x_{i,\text{stop}} \) distance and correspondingly \( t_{i,\text{stop}} \) time away from the traffic light, irrespective of the traffic light state and timings.

While driving, if the ego vehicle satisfies condition (i), it starts following the lead vehicle such that \( x_{i,\text{safe}} \) distance is always maintained. While following the lead vehicle, if condition (ii) is met, and as per Lemmas 4.1, 4.2, and 4.3, the ego vehicle needs to reduce speed, then the traffic lights take a precedence over following the lead vehicle. This behavior is captured in Algorithm 4.2, by extending the conditions in Algorithm 4.1.

As per Algorithm 4.2, in addition to the conditions mentioned in Algorithm 4.1 and therefore the \textit{action} suggested as per the traffic light using \( flag_{ij} \), the following conditions also need to be checked to maintain a safe distance from the lead vehicle \((\text{Line 2})\). To do so, the MBS first acquires the estimated positions of the lead...
Algorithm 4.1: Mobility characteristics estimation and dwell time assignment for vehicles entering an empty link $L$ with no vehicles in front (Case 1)

```
1  Init: \( v_0, t_0, r_0^m, t_0^m, d_0^m \)
2  Vehicle Data: \( J_0^m, d_s, V_s, d_s' \)
3  RSU Data: \( r, m \)
4  Result: \( \delta_{s1}, \ldots, \delta_{sm}, J_m, t_m^m, d_m^m \)
5  Init: \( T_j = 0, X_i = 0, j = 1, T_S = 0, v_i = v_j^m, t_j^m = 1 \)
6  Init: action = flag_{lead}, (Algorithm 4.1) or flag_{act} (Algorithm 4.2)
7  Init: \( x_j^m, t_j^m, x_j^const, t_j^const \)
8  /* \( X_j \): variable to track distance traveled by \( j^{th} \) vehicle, until the vehicle crosses the link \( (d_j) \). */
9  while \( X_j \leq d_j \) do
10     if action = constant speed then
11        \( x_{step} = v_j \times t_{step} \)
12     if action = accelerate then
13        Get \( \nu_{x_{step}} \) and \( x_{step} \)
14        if action = accelerate then
15          Get \( x_{step} \) and \( t_{step} \)
16          \( X_i \leftarrow X_i + x_{step}, T_i = T_i + t_{step}, T_S = T_S + t_{step} \)
17          if \( X \geq 2r \) then // end of range for \( j^{th} \) RSU
18            \( \delta_{ij} \leftarrow T_S, j = j + 1 \)
19            \( T_S = 0 \)
20          if \( X_i = d_j^m \) or \( T_i = d_j^m \) then
21            /* deadline expected to be met */
22            \( d_j^m = T_i, J_m^n = j, t_j^const = T_S \)
23          if \( T_i = t_j^const = T_S \)
24            /* check conditions for Lemma 4.1, 4.2 and 4.3 */
25            if Lemma 4.1 condition (i) or (ii) met then
26              \( flag_j \leftarrow \text{constant speed} \)
27            if Lemma 4.2 condition (i) met then
28              \( \text{Set wait time } t_j^m = t_j^m - t_j^const - t_{step} \)
29              \( flag_j \leftarrow \text{decelerate} \)
30            if Lemma 4.2 condition (ii) met then
31              \( \text{Set wait time } t_j^m = t_j^m - (t_j^const + t_j^step - t_j^m) \)
32              \( flag_j \leftarrow \text{decelerate} \)
33            if Lemma 4.3 condition met then
34              \( \text{Set deceleration time } t_j^m \)
35              \( \text{Set wait time } t_j^m = 0 \)
36              \( flag_j \leftarrow \text{decelerate} \)
37          else /* wait time as RSU \( j \)'s dwell time */
38              \( \delta_{ij} \leftarrow \delta_{ij} + t_j^m \)
39          break
```

vehicle from its previous calculations (Line 3), if the ego vehicle is \( x_j^{safe} \) distance away from the lead vehicle it must continue maintaining its current speed (Lines 4 and 5). If the gap between the ego vehicle and the lead vehicle is less than \( x_j^{safe} \), the ego vehicle must decelerate to increase the gap (Lines 6 and 7). Alternately, if the gap between the ego vehicle and the lead vehicle is more than \( x_j^{safe} \), the lead vehicle does not impact the actions of the ego vehicle and the ego vehicle may accelerate to increase the gap as long as it does not violate the speed limit (Lines 8-12), \( flag_{lead} \) depicts the suggested maneuver based on the leading vehicle’s mobility. \( flag_{act} \) then decides
Deadline-Aware Task Offloading for Vehicular Edge Computing Networks using Traffic Lights Data

Algorithm 4.2: Extension to Algorithm 4.1 to account for the influence of a leading vehicle on the ego vehicle’s maneuvering (Case 2)

1. Init: $h$: time headway, $x_{safe}$: standstill distance
2. $x_{safe}^i \leftarrow hV_i + x_{safe}$
3. $X_{i-1}^T \leftarrow$ estimated position of $V_{i-1}$ at $T_i$ time
   /* monitoring the gap with lead vehicle */
4. if $X_i - X_{i-1}^T = x_{safe}^i$ then
   5. $flag_{lead} \leftarrow$ constant speed
   6. if $X_i - X_{i-1}^T < x_{safe}^i$ then
   7. $flag_{lead} \leftarrow$ decelerate
   8. if $X_i - X_{i-1}^T > x_{safe}^i$ then
   9. if $V_i < V_{lim}$ then
      10. $flag_{lead} \leftarrow$ accelerate
         /* lead vehicle does not influence ego vehicle actions. */
   11. else
      12. $flag_{lead} \leftarrow$ constant speed
         /* action influenced by traffic light or lead vehicle */
   13. if $flag_s = decelerate$ or $flag_{lead} = decelerate$ then
   14. $flag_{act} \leftarrow$ decelerate
   15. else
   16. if $flag_s = constant speed$ or $flag_{lead} = constant speed$ then
   17. $flag_{act} \leftarrow$ constant speed
   18. $action \leftarrow flag_{act}$

whether the traffic light ($flag_s$) or the lead vehicle ($flag_{lead}$) will affect the ego vehicle’s maneuver ($action$) (Lines 13-18).

With the dwell times assigned and mobility estimates known, the MBS now assigns appropriate RSUs for task $\tau_i$ of vehicle $V_i$ based on its deadline $d_i^d$. Note that we also determine the RSU where the vehicle will be, when its task deadlines are met while estimating the mobility characteristics.

4.4 RSU Allocation for Task Offloading

We know that a task $\tau_i$ needs to be completed within $d_i^e$ time of the task’s arrival. We also determined $d_i^d$ corresponding to the distance-based deadline ($d_i^d$) and the RSU$_{end}$ where the vehicle will be when its task meets
the deadline. Therefore, the task has to be uploaded, computed and downloaded within \( RSU_1, \ldots, RSU_{end} \). Further, the duration \( t_i^{end} \) that vehicle \( V_i \) will spend within the range of \( RSU_{end} \) is also known.

**Download RSU:** As explained in the motivating example, offloading tasks such that they complete closer to the deadline improves the resource utilization at the RSUs. To complete \( \tau_i \) closest to the deadline, the most appropriate RSU to download would be \( RSU_{end} \) if the duration for which vehicle \( V_i \) remains in the range of \( RSU_{end} \) is atleast equal to the download time of the task \( \tau_i \). The download time \( t_i^{dl} \) of task \( \tau_i \) from any \( RSU_j \) depends on the download size \( \sigma_i^{dl} \) of the task and the communication data rate \( \eta \) such that \( t_i^{dl} = \eta \cdot \sigma_i^{dl} \). Therefore, if \( t_i^{dl} \leq t_i^{end} \), then the task can be downloaded at \( RSU_d = RSU_{end} \). However, if the duration is not enough to download the task at \( RSU_{end} \) (\( t_i^{dl} > t_i^{end} \)), any \( RSU_j \) closest to \( RSU_{end} \) whose dwell time \( \delta_{id} \geq t_i^{dl} \) is chosen using Equation 9 and its constraints.

\[
\text{maximize } d \quad \text{(9)}
\]

\[
\text{subject to } \delta_{id} \geq t_i^{dl}, \quad 1 \leq d \leq \text{end.} \quad \text{(9a)}
\]

Equation (9) ensures that we maximize the index \( d \) of the RSU and thereby choose an RSU closest to \( RSU_{end} \) where we download the task with the following constraints. **Constraint (9a):** the dwell time of the chosen \( RSU_d \) is atleast equal to the download time \( t_i^{dl} \) of the task, and **Constraint (9b):** the index \( d \) of the chosen RSU must be less than \( \text{end} \) to meet the deadline.

**Compute RSU:** Upon choosing \( RSU_d \), the MBS then finds \( RSU_c \), where the task can be computed. The computation time \( (t_i^c) \) of \( \tau_i \) depends on the compute requirement \( C_i \) of the task and the clock speed \( z_c \) of the processors at \( RSU_c \), such that \( t_i^c = \frac{C_i}{z_c} \) and the processor availability \( p_c \) at \( RSU_c \). Note that the RSUs are connected via a backhaul network and incur an additional delay of \( \rho_{cd} \) (Equation 1) for relaying the data from \( RSU_c \) to \( RSU_d \). Therefore, if there is sufficient time left within the range of \( RSU_d \) to also compute \( \tau_i \), i.e., \( \delta_{id} - t_i^{dl} \geq t_i^c \), then the task is computed at \( RSU_d \) without incurring any additional backhaul network delay. Otherwise, \( RSU_c \) can be found using Equation 10 and its constraints.

\[
\text{maximize } c \quad \text{(10)}
\]

\[
\text{subject to } \sum_{j=1}^{d} \delta_{ij} \geq t_i^c + \rho_{jd} + t_i^{dl}, \quad \text{(10a)}
\]

\[
\rho_{dd} = 0, \quad \text{(10b)}
\]

\[
p_c \geq 1, \quad \text{(10c)}
\]

\[1 \leq c \leq d. \quad \text{(10d)}
\]

Equation 10 maximizes the index \( c \) of the RSU chosen to compute the task so that the task is computed as close as possible to where the task is to be downloaded, with the following constraints. **Constraint(10a):** the sum of the dwell times of all RSUs from the chosen \( RSU_c \) to \( RSU_d \) where the task is to be downloaded is atleast equal to the sum of the compute time \( t_i^c \), relay time \( \rho_{cd} \) of the backhaul network between \( RSU_c \) and \( RSU_d \) and the download time of the task \( t_i^{dl} \). This ensures that the vehicle reaches the range of \( RSU_d \) when the task is ready to download and there is no wait time incurred. **Constraint(10b):** if the dwell time of \( RSU_d \) is sufficient to compute and download the task, then it will not add any additional relay time to the task’s completion time, **Constraint(10c):** there has to be atleast one processor available at \( RSU_c \) to compute task \( \tau_i \), and **Constraint(10d):** the task must be computed before it is to be downloaded at \( RSU_d \).
Upload RSU: The MBS now determines the appropriate $RSU_u$ where the task needs to be uploaded such that the task is computed and downloaded in time of its deadline. The upload time $t_{u}^{ul}$ of task $t_{i}$ from any $RSU_j$ depends on the upload size $\sigma_{i}^{ul}$ of the task and the communication data rate $\eta$ (as per Shannon Hartley theorem [34]) such that $t_{i}^{ul} = \eta \cdot \sigma_{i}^{ul}$. $RSU_u$ is therefore chosen using Equation 9 and its constraints.

\[
\begin{align*}
\text{maximize } & u \\
\text{subject to } & \delta_{iu} \geq t_{i}^{ul} + \rho_{uc}, \\
& \rho_{cc} = 0, \\
& 1 \leq u \leq c.
\end{align*}
\]

Equation 11 maximizes the index $u$ so that the RSU with the highest index is chosen to upload while constrained by the following. Constraint(11a): the dwell time in $RSU_u$ is sufficient to upload the task and relay it to $RSU_c$ for computation, Constraint(11b): if $RSU_u = RSU_c$ no relay time is incurred, and Constraint(11c): The task must be uploaded before it is computed.

By solving Equations 9, 10 and 11 using linear optimization techniques, $RSU_u$, $RSU_j$ and $RSU_d$ can be determined. Once the RSUs are assigned, the vehicle can now upload the task as soon as it is in the range of the assigned $RSU_u$. The MBS then transmits the task to $RSU_c$ if needed, where the task is computed and then relayed to $RSU_d$. Finally, when the vehicle enters the communication range of $RSU_d$ and starts downloading the task. The above formulation ensures that the task is completed as close to the deadline as possible.

**Complexity Analysis:** The dwell time estimation algorithm (Alg. 4.1) has a time complexity of $O(T \times R)$ where $T$ denotes total task offloading requests in queue and $R$ denotes the total RSUs along the link. Once the dwell times are estimated, the RSU assignment is a linear optimization problem with time complexity of $O(R)$.

5 EVALUATION

In this section, we compare our proposed approach with the state-of-the-art task offloading strategies as baselines. Let us describe the simulation parameters used for the evaluation.

5.1 Evaluation Setup

Table 1 mentions the parameters for traffic infrastructure, vehicles, task model, communication model and the VEC framework and also summarizes the notations used in the paper. These parameters closely resemble common VEC workloads, network and traffic infrastructure configurations [27, 53]. We emulate large-scale traffic moving through a link in an urban environment using VISSIM [10], a microscopic traffic simulator. The vehicles arrive at random arrival times as per default distribution in VISSIM. All vehicles that enter the traffic network in VISSIM follow the driving maneuver model and its parameters considered in this paper. An external communication framework emulator based on Python 3 acts as a main base station (MBS) that interacts with VISSIM to acquire vehicle data, calculate dwell times, generate synthetic workloads based on the chosen task parameters, and assign RSUs to the vehicles.

We compare our proposed deadline-based strategy that uses connected traffic lights (referred as traffic lights-aware) with a closely related work [27] (referred as wait time-aware) which considers a time-statistical model to account for the average delays over time due to a traffic light but does not consider connected traffic lights with known traffic light timings. Further, we also compare our work with a mobility-aware task offloading strategy [53] (referred as mobility-aware) that considers mobility without traffic lights to minimize task offloading times. Note that the proposed approach as well as the wait time-aware [27] and the mobility-aware [53] offload tasks onto the RSUs only if the task is expected to be completed before its deadline.
Table 1. Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traffic Infrastructure</strong></td>
<td></td>
</tr>
<tr>
<td>Link length $d_l$</td>
<td>[400,800] m</td>
</tr>
<tr>
<td>Speed limit $v_{lim}$</td>
<td>[25,35] mph</td>
</tr>
<tr>
<td>number of lanes $l$</td>
<td>1</td>
</tr>
<tr>
<td><strong>Traffic Light</strong></td>
<td></td>
</tr>
<tr>
<td>Cycle time $t_s^{cycle}$</td>
<td>60 s</td>
</tr>
<tr>
<td>Max. green time $t_s^g$</td>
<td>26 s</td>
</tr>
<tr>
<td>Yellow time $t_s^y$</td>
<td>4 s</td>
</tr>
<tr>
<td>Max. red time $t_s^r$</td>
<td>30 s</td>
</tr>
<tr>
<td><strong>Vehicles</strong></td>
<td></td>
</tr>
<tr>
<td>Comfortable acceleration</td>
<td>$a$ 1 m/s²</td>
</tr>
<tr>
<td>Comfortable deceleration</td>
<td>$b$ -0.9 m/s²</td>
</tr>
<tr>
<td>Time headway $h$</td>
<td>0.9 s</td>
</tr>
<tr>
<td>Standstill gap $s_{safe}$</td>
<td>1.5 m</td>
</tr>
<tr>
<td><strong>VEC Task Model</strong></td>
<td></td>
</tr>
<tr>
<td>Task type</td>
<td>[time-based, location-based]</td>
</tr>
<tr>
<td>Upload size $a^{ul}$</td>
<td>1 - 100 MB</td>
</tr>
<tr>
<td>Compute requirement $C$</td>
<td>0.24 - 24 x10⁹ clock cycles</td>
</tr>
<tr>
<td>Download size $a^{dl}$</td>
<td>1 - 100 MB</td>
</tr>
<tr>
<td>Time-based deadline $d_t$</td>
<td>100ms - 5s</td>
</tr>
<tr>
<td>Distance-based deadline $d_r$</td>
<td>1m - $d_l$  m</td>
</tr>
<tr>
<td><strong>VEC Communication Framework</strong></td>
<td></td>
</tr>
<tr>
<td>Communication radius $r$</td>
<td>25 m</td>
</tr>
<tr>
<td>Number of processors</td>
<td>[2, 4, 8]</td>
</tr>
<tr>
<td>Clock speed $z$</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td>Wireless data rate $\eta$</td>
<td>500Mbps</td>
</tr>
<tr>
<td>Backhaul data rate $\eta_{bh}$</td>
<td>1Gbps</td>
</tr>
<tr>
<td>Backhaul relay constant $c_{bh}$</td>
<td>10ms</td>
</tr>
</tbody>
</table>

5.2 Performance in an Ideal Setting

We evaluate all the three approaches on the basis of (i) the percentage of tasks, out of the entire taskset, that a strategy deems feasible to offload onto the RSUs based on its estimation of the vehicular mobility and the task deadlines, and (ii) the percentage of tasks that miss their deadlines out of all the tasks offloaded onto the RSUs. Further, in these set of experiments, we assume that the vehicles and the traffic flow behave ideally and follow the maneuvers as planned by the ADAS mobility planner.

**Effect of traffic flow on task offloading (Figure 5):** As the traffic flow increases, the number of vehicles on the link and therefore the number of tasks to be offloaded increase as well. As shown in Figure 5, with traffic flow changing from light to heavy traffic (400 to 1000 vehicles per hour per lane [vhphpl]), the dwell times of the vehicles increase within different RSU ranges due to an increase in vehicle queues at the traffic light. Results show that under heavy traffic the existing approaches fail to adapt to this changing traffic patterns and lead to deadline misses. The wait time-aware approach offloads 47% of all tasks requests and missing the deadlines for 23% of the offloaded tasks, while the mobility-aware approach finds 44% of the tasks feasible to offload and missing the deadlines for 22% of the offloaded tasks. Further, location-based tasks comprise of 53% and 59% of the offloaded tasks that missed their deadlines in time delay-aware and mobility-aware approaches, respectively.
Fig. 5. Effect of varying traffic flow on task offloading. Increasing traffic flow leads to a reduction in the percentage of tasks offloaded out of total task requests (colored). However, the existing approaches cause a significant number of tasks to miss their deadlines (hatched).

Fig. 6. Effect of increasing processor availability on task offloading. With increase in the number of processors all approaches offload more tasks onto the RSUs (colored), however, the wait-time-aware and mobility-aware strategies lead to a significant number of tasks missing their deadlines (hatched).

**Effect of processor availability on task offloading (Figure 6):** As more processors become available at the RSUs, more tasks can be computed in parallel. As shown in Figure 6, with flow fixed to medium traffic, with increase in the number of available processors at each RSU, the number of tasks offloaded for all three approaches increases. It is however important to note that as the number of available processors increase at each RSU, the existing approaches also show an increase in deadline misses (up to 25% deadlines missed with 8 parallel processors). As the number of available processors at each RSU increase, the existing approaches have ample available resources to offload the tasks as soon as they arrive. However, due to the lack of traffic-aware mobility, the vehicles end up leaving the RSU range before the tasks execution completes and thereby leading to an increase in missed deadlines. For our proposed approach, increasing processor availability, increases the number of offloaded tasks with all of them meeting their deadlines and thereby maximizing RSU resource utilization. We skip the discussion on the effect of increasing the link length, as it also leads to an increase in the number of RSUs and processor availability, and similar results were obtained.
Fig. 7. Effect of changing speed limits on task offloading. Increases to speed limit lead to a change in the percentage of tasks offloaded (colored), with a significant increase in the deadline misses (hatched) for the existing strategies.

**Effect of speed limit on task offloading (Figure 7):** We choose two speed limits of 25 mph and 35 mph common for urban traffic lights. With fixed traffic flow (700 vphpl) and number of processors (2 processors per RSU), as the speed increases, the vehicles have lesser dwell time within each RSU, and thereby causing a drop in number of tasks offloaded for our proposed approach. However, with our proposed approach, we still meet the deadlines for all offloaded tasks. On the contrary, the wait time-aware approach sees an increase in the number of offloaded tasks, as higher vehicle speeds causes reduced travel times and wait times at the traffic light. However, increase in speed also causes an increase in the number of deadline misses for the wait time-aware approach, due to increased dwell time estimation errors. Similarly, the mobility-aware offloading too leads to an increased deadline misses as it relies on the average vehicle speeds. Note that with increase in processor availability, a similar trend as in Figure 6 was observed, where at higher speeds, more processor availability meant increase in the number of tasks offloaded but it also lead to an increase in deadline misses.

5.3 Performance under Uncertainty

So far, we have considered that the CAVs within our model perform known driving maneuvers and behavior as the vehicle follow predefined constrained motion such as constant speeds, reducing speeds with known deceleration rates, increasing speed with known acceleration rates. However, in a realistic urban driving environment, the controllers in fully automated vehicles are affected by noise and disturbances which do not allow the vehicles to follow precise mobility constraints and maneuvers. To evaluate the efficacy of our model in a realistic setup with noise and uncertainty, all vehicles in our simulated evaluation show a uniform distribution ranging between $v_{lim} - 5$ to $v_{lim}$ instead of strictly following a target speed of $v_{lim}$. The vehicles, instead of following strict acceleration and deceleration rates, are now modelled with Weidemann 74 car-following model [33]. In this work, Weidemann 74 model is preferred over other car-following models as it accurately captures non-ideal driving conditions where different drivers show varying perception, reaction times and driving behaviors in the same environment. With the car-following model in place, the vehicles show maximum acceleration ranging from 2.5 to $3 \text{ m/s}^2$, and deceleration ranging from $-2.5$ to $-3 \text{ m/s}^2$. These parameter ranges are determined based on driver reaction times and passenger comfort [33].

Now, changing traffic parameters such as traffic speeds and flow rate may lead to vehicles showing increased stop-and-go motion and thereby significantly impact the deviation in modeled vehicle speeds, acceleration and following distances. We therefore investigate the impact of (i) traffic speeds (Fig. 8) and (ii) traffic flow (Fig. 9) on deadline misses in our proposed approach.
Speed limit and deadline misses in non-ideal conditions: As expected (from Fig. 7), increasing speed limits does cause a reduction in the tasks successfully offloaded onto the edge platform. Under realistic conditions however (Fig. 8), about 5% of the offloaded tasks experience deadline misses.

Traffic flow and deadline misses in non-ideal conditions: Increasing traffic flow from low to heavy traffic leads to a reduction in tasks being offloaded onto the RSUs. However, highly varying traffic (very low or very high flow rates) leads to an increased stop-and-go motions, variations in desired speed and acceleration. Therefore, as shown in Fig. 9, under non-ideal conditions, low traffic (where vehicles can travel at higher speeds) or heavy traffic (where vehicles are known to show increased stop-and-go motions) lead to increased (up to 10%) deadline misses. Under medium traffic, all vehicles have a more regularized traffic flow, and therefore experience extremely few (2%) deadline misses.

Effect of model (speed, acceleration and car-following model) uncertainties on deadline misses (Fig. 10): Finally, we investigate the impact on non-ideal conditions in existing approaches when compared to our proposed approach. The uncertainty in speed, acceleration and following distance added in a realistic car-following model are not known apriori and therefore are not considered during dwell time estimation and RSU allocation phase, in our proposed approach as well as other existing state-of-the-art approaches. Due to this, deadline misses are expected due to an ideal model of the system. Figure 10 shows the effect of this uncertainty on deadline misses.
with our proposed approach as well as the existing state-of-the-art approaches under varying traffic flow. Note that as the traffic flow increases from low to heavy traffic, the vehicles show increased stop-and-go motion with larger deviation from the ideal speeds, acceleration and following distances. As shown, our proposed approach, while still suffering from the estimation errors due to non-ideal conditions, causes up to 78% fewer deadline misses than the existing approaches, showing that our proposed dwell time estimation and RSU allocation approach, by capturing traffic flow parameters and mobility behavior of the vehicles, is more resilient to uncertainties and leads to significantly higher utilization of the edge compute platform.

6 CONCLUSION

In this paper, we presented a comprehensive task model to support emerging VEC applications that incorporates tasks with fixed time-based deadline as well as traffic-dependent tasks whose deadlines depend on traffic flow, location and traffic lights. We then incorporated the traffic lights data by leveraging the connected infrastructure to accurately estimate the mobility characteristics and dwell times of the vehicles. We then used the dwell time estimations to allocate appropriate RSUs to upload, compute and download the tasks such that they always meet their deadlines. We also show that by completing tasks closer to their deadlines, we can improve on the resource utilization of the edge resources. Finally, our evaluation using VISSIM, a traffic simulator, showed that our proposed approach outperforms the existing approaches by consistently offloading more tasks onto the RSUs and meeting 100% of the task deadlines. Extending the proposed model for a network-wide edge resource optimization is left for future work.

REFERENCES


24 • Pratham Oza, Nathaniel Hudson, Thidapat Chantem, and Hana Khamfroush


[37] SAE. 2021. SAE J3016 Recommended Practice: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.


