Deadline-Aware Task Offloading for Vehicular Edge Computing Networks Using Traffic Light Data

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

As vehicles have become increasingly automated, novel vehicular applications have emerged to enhance the safety and security of the vehicles and improve user experience. This brings ever-increasing data and resource requirements for timely computation by the vehicle's on-board computing systems. To meet these demands, prior work proposes deploying vehicular edge computing (VEC) resources in road-side units (RSUs) in the traffic infrastructure with which the vehicles can communicate and offload compute-intensive tasks. Due to the 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 mobility through road traffic. Existing task offloading strategies do not consider the influence of traffic lights on vehicular mobility while offloading workloads onto the RSUs. This causes deadline misses and quality-of-service (QoS) reduction for the offloaded tasks. In this article, 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 RSUs compared with existing work without causing deadline misses, maximizing the resource utilization of 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

ACM Reference format:
Pratham Oza, Nathaniel Hudson, Thidapat Chantem, and Hana Khamfroush. 2024. Deadline-Aware Task Offloading for Vehicular Edge Computing Networks Using Traffic Light Data. *ACM Trans. Embedd. Comput. Syst.* 23, 1, Article 8 (January 2024), 25 pages.
https://doi.org/10.1145/3594541

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 transition from being human driven (Level 0) to fully

Authors' addresses: P. Oza and T. Chantem, Virginia Tech, USA; emails: {prathamo, tchantem}@vt.edu; N. Hudson, University of Chicago, USA and Argonne National Laboratory, USA; email: hudsonn@uchicago.edu; H. Khamfroush, University of Kentucky, USA; email: khamfroush@uky.edu.

1539-9087/2024/01-ART8 $15.00
https://doi.org/10.1145/3594541

autonomous (Level 5) as per vehicular automation standards [36]. 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 [42]. 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) [30]. 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 passenger experience. Applications involving media-rich infotainment, *augmented reality* (AR)–based media streaming, collaborative data aggregation, and interactive user applications are resource intensive [16] 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 [23]. The upcoming *vehicular edge computing (VEC)* paradigm provides an attractive alternative to cloud computing solutions. Under the VEC framework, *edge servers* are deployed near 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 that enable high-throughput short distance communication between vehicles and the edge servers deployed along the roadways [1]. Data and connectivity among the entities within the traffic ecosystem —such as roadside sensors, vehicles, and traffic lights —can now be leveraged to implement novel algorithms that enhance the overall QoS and performance of various automotive applications [15, 46]. Such localized data can be made available on an edge platform. Therefore, VEC is a promising approach to handle 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 communication range of a nearby RSU. Recent studies are 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 RSU communication range [18, 47, 51, 52]. Figure 1 depicts a typical task-offloading scenario such that the timeliness requirements of the task are met. In Figure 1, as the vehicle travels along a road link, it communicates with the nearby RSUs to upload a task. The task 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 where 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 a red traffic light. Therefore, traffic conditions and vehicular mobility significantly impact the communication link between vehicles and RSUs and, therefore, the choice of RSUs to offload the tasks.

In addition to influencing the communication between vehicles and RSUs, vehicular mobility and traffic conditions also affect the requirements of many novel vehicular applications [11, 14]. For example, consider a dynamic routing application on a vehicle that finds an optimized route to a preferred destination based on collaborative data from surrounding vehicles. If an optimal route

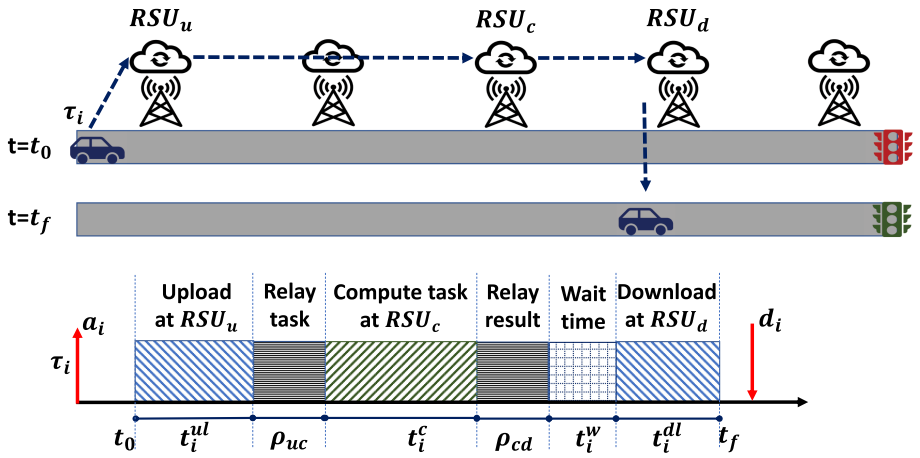

Fig. 1. Offloading scenario of task τ_i uploaded at RSU_u, computed at RSU_c and then downloaded at RSU_d, with completion time of the task as $t_i^{ul} + \rho_{uc} + t_i^c + \rho_{cd} + t_i^w + t_i^{dl}$.

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 traffic flow and traffic light data when the task arrives. To support emerging applications [46], we propose a comprehensive task model comprising 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 in which light timings influence the vehicular mobility characteristics, which, in turn, affect 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 RSU 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 [50], resource utilization [34], bandwidth usage [38], or access costs [26]. As vehicles enter a lane in a sequential manner and RSUs are only accessible once the vehicle is in communication range, greedy approaches offload the tasks for the earlier-arriving vehicles on the RSUs that are located closer to 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, τ_i, has a later deadline. When a later-arriving vehicle (V_{i+1}, marked in blue) has a more urgent task (τ_{i+1}) to offload, the resources are blocked by τ_i, leading to τ_{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 RSUs, vehicles not only maximize resource utilization at RSUs, but also allow 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 our deadline-based task-offloading strategy in which we offload tasks such that they complete closer to the deadline, we ensure that the

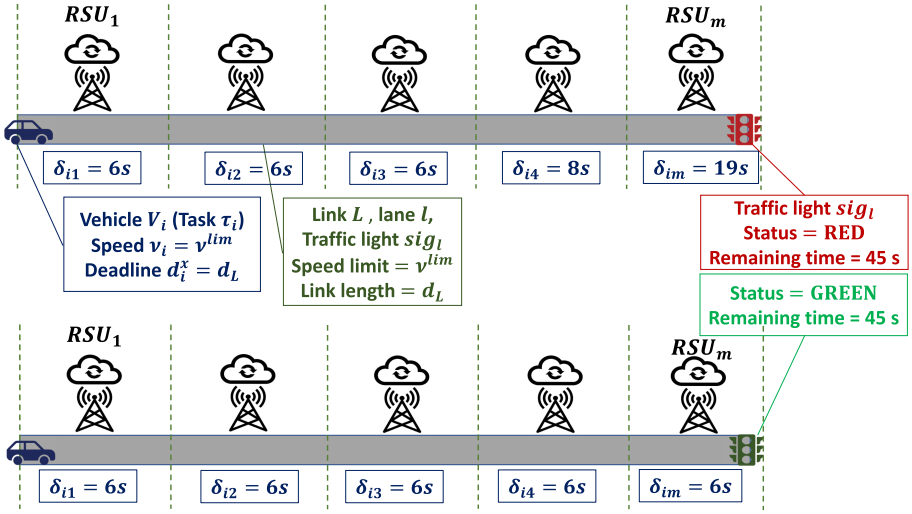

Fig. 2. Influence of traffic lights on dwell times: Vehicle spends significantly longer time within the range of RSU_m when the light is red (top) in comparison with a situation in which the light is green (bottom). δ_{ij} is the dwell time of vehicle V_i in the communication range of RSU_j.

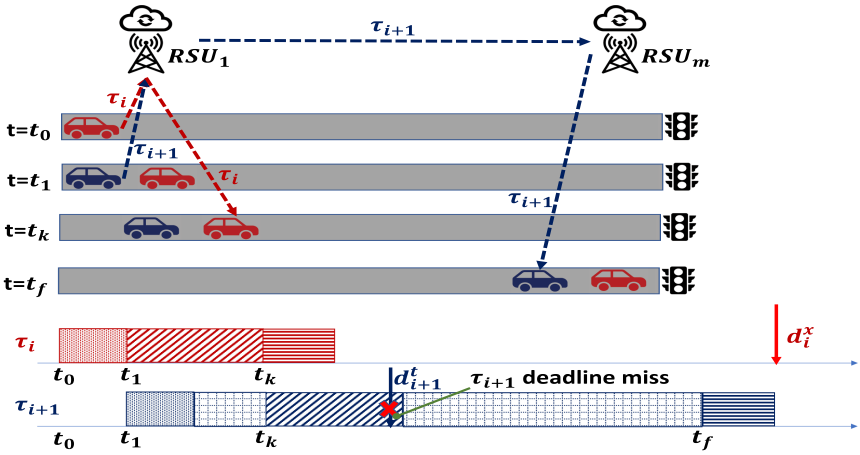

Fig. 3. Offloading tasks without considering the traffic lights affecting vehicular mobility. Task τ_{i+1} would miss its deadline; hence, the existing strategies would drop the task and choose not to be offloaded.

offloaded tasks meet their deadlines and that 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 traffic light timing data to calculate (i) the mobility characteristics and travel time for 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.
- We evaluate our proposed approach against existing offloading strategies using VISSIM [10], a microscopic traffic simulator.

1.1 Article Structure

The remainder of this article is divided into the following sections. Section 2 provides a survey of prior work in task-offloading techniques on the VEC 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. In Section 5, we evaluate the performance of our proposed approach when compared with other existing task-offloading algorithms. Section 6 contains concluding remarks and insights into future work.

2 RELATED WORK

With increasing demands for performance, data- and compute-hungry processes on mobile devices induced breakthroughs in small-cell network, multi-antenna, and millimeter-wave communications to provide highly reliable and gigabit wireless access to next-generation systems [44]. Due to these advancements it became 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 works have focused on tackling various connectivity, resource, mobility, and network latency constraints to optimize the utilization of MCC [4, 21, 37].

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 [25]. Various architectures were proposed to enable cloud-based computation for vehicular applications [3, 19]. Efforts have also been made to make VCC more efficient via optimized resource allocation techniques [48], and more secure [40] and more energy-efficient [43] 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, the vehicular edge computing (VEC) paradigm was introduced [41], in which storage and compute nodes were brought closer to the proximity of the end user, greatly reducing the network latency [7] and energy consumption [35].

Recent studies have proposed various task-offloading mechanisms to offload more tasks from the vehicle onto the edge platform [13, 41]. 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, 49]. The authors of [41] propose an offloading mechanism for complex vehicular tasks with dependencies. In [26], 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 RSUs of the traffic infrastructure, many researchers also propose a vehicular fog computing (VFC) architecture in which the resources available on other vehicles driving in the vicinity or parked vehicles are utilized to meet demands. Solutions proposed for utilizing the VFC framework are out of the scope of our work; interested readers can refer to existing literature [22, 27]. None of these proposed solutions, however, considers the mobility constraints in a vehicular environment.

Vehicular mobility can significantly impact resource utilization as well as task allocation onto the edge. Novel task offloading and resource allocation mechanisms have been proposed that

incorporate a driving model for vehicles [2, 5, 52]. 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, and more. 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 a 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 L has length d_L with a posted speed limit of v^{lim} for traffic flowing along link L to approach an intersection within an urban traffic network. Link L can have one or more lanes $\{l_1, \ldots, l_k\}$, 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 a fixed cycle time or a state-of-the-art traffic controller for which timings are known for a fixed prediction horizon [31]. The traffic lights have connectivity to broadcast traffic light data to the connected infrastructure. This traffic light data consists of its current state ($\psi_{s_i} = \{red, green, yellow\}$), remaining time in current state ($t_{s_i}^{rem}$), maximum red ($t_{s_i}^{r}$), green ($t_{s_i}^{g}$) and yellow phase time ($t_{s_i}^{y}$), and the cycle time ($t_{s_i}^{cycle} = t_{s_i}^{r} + t_{s_i}^{g} + t_{s_i}^{y}$).

3.2 VEC Task Model

As introduced in Section 1, existing task models consider only those tasks with fixed time-based deadlines. In this work, in addition to tasks with static deadlines, we 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 changes dynamically.

We therefore consider a task model in which 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.
(2) *Distance-based task*: Such a task has a deadline based on a distance before which the task must be executed.

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

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 with 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 the case of tasks with dependencies, our model can be utilized by decomposing interdependent tasks into multiple independent subtasks with local deadline assignments [29]. 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 beyond the scope of 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 RSUs, and (iii) connected vehicles requiring edge resources.

An MBS communicates wirelessly with all vehicles via a high-throughput wired connectivity with edge-enabled RSUs, within its communication range. MBSs have a larger communication range than other edge-enabled RSUs, as they cover multiple intersections in a traffic network. RSUs comprise infrastructure (e.g., traffic lights, sensors, cameras) that are equipped with additional compute, storage, and communication resources. RSUs communicate with vehicles within their range via short-range wireless connectivity (fe.g., 5G-CV2X). All RSUs are connected via a wired backhaul network and are distributed along the roadside such that the entire road link is covered without any *blind spots*. Additionally, in our model, the RSUs do not share the processing resources. 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 [25].

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 contiguous 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 [20]. The communication radius is therefore determined using Friis's equation [39] 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 edge resource demands. Optimal placement of RSUs and edge services is an ongoing research problem [17] and is beyond the scope of this work. In short, our system considers m RSUs (i.e., $RSU_1, \ldots, RSU_m$), which 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 of the RSUs and the MBS, (ii) a low-latency short-range wireless network between the vehicles and RSUs, and (iii) a wireless long-range connectivity between the vehicles and MBS.

Based on the upload (σ_i^{ul}) and download (σ_i^{dl}) size of the task, there is a corresponding upload (t_{ij}^{ul}) and download (t_{ij}^{dl}) 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 η. We also assume that the transmit power is adjusted such that the signal-to-noise ratio 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 [33] can be used to determine the data transmission rate η 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 (ρ_{jk}) between two RSUs, RSU_j and RSU_k, depends on the distance between the RSUs. It can be defined as

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

where c_{bh} is the minimum transmission relay time between two consecutive RSUs [13] and η_{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 [36]. 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 operating on 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. These vehicles 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 their 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 the 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 [26, 52] for task assignments in urban scenarios, 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 $x_i^{safe} = hv_i + x_{safe}$ from their leading vehicle using 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 [45]. 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 requests for RSU allocation to upload, compute, and download a task. In our model, we consider worst-case task-offloading demands in which 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. We first provide highlights of 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 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, and upload and download size of the data, which is communicated via the on-board cellular network capabilities.
(3) Optionally, the MBS checks whether the task can be completed before its deadline if offloaded onto 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 to 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 (τ_i^{ul}); (ii) task *relay time* (ρ_{uc}), that is, 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 (τ_i^c); (iv) task *relay time* (ρ_{cd}) from RSU where the task is computed to the RSU from where the task will be downloaded; (v) *wait time* (τ_i^w) 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 (τ_i^{dl}). Figure 1 depicts a typical offloading scenario of task τ_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 a task must not exceed its deadline to ensure that it meets the deadline. Therefore, the MBS must select appropriate RSUs to upload (RSU_u), compute (RSU_c), and download (RSU_d) the task so 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 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*, which 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 $\delta_{ij}, \forall j = 1, \ldots, m$. The MBS derives the dwell times based on the vehicle's mobility characteristics.

Our deadline-based offloading strategy consists of offloading tasks such that they complete as close to the deadline as possible by 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, of which $RSU_2, \ldots, RSU_{m-1}$ are occupied by other compute-intensive tasks. The only two resources available for offloading tasks are RSU_1, located at the entry of link L, and RSU_m, located at the end of link L near the traffic light s_L. Vehicle V_i enters link L at time t_0 and requests to offload τ_i, followed by V_{i+1} entering link L at time $t_1 = t_0 + 1$ requesting to offload τ_{i+1}. τ_i is a distance-based task such that it must complete before V_i crosses the traffic light s_L and, therefore, exits RSU_m's range. Alternately, τ_{i+1} is a time-sensitive task with a deadline of 3 seconds with reference to its arrival time. The characteristics of both tasks are as follows: $t_i^{ul} = 0.2$ s, $t_i^c = 3$ s, and $t_i^{dl} = 0.2$ s, and $t_{i+1}^{ul} = 0.1$ s, $t_{i+1}^c = 2$ s, and $t_{i+1}^{dl} = 0.1$ s. Both vehicles enter the link driving at the speed limit ($v_i = v_{i+1} = v^{lim}$). The travel time between RSU_1 and RSU_m is 25 seconds when driving at the

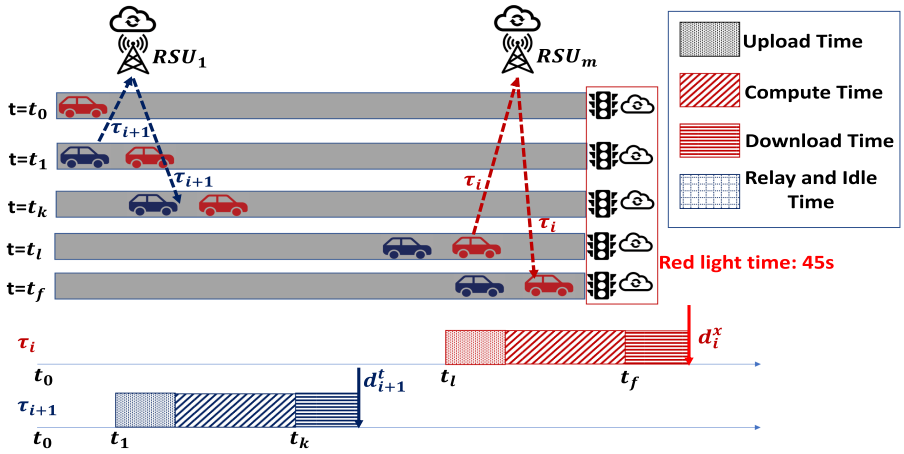

Fig. 4. Task offloading with mobility characteristics using traffic light data. Both tasks meet their respective deadlines.

speed limit. Dwell times for vehicles V_i and V_{i+1} within the range of RSUs 1 and m are $\delta_{i1} = \delta_{im} = \delta_{(i+1)1} = \delta_{(i+1)m} = 3.7$ seconds.

4.2.1 Offloading with Existing Approach. As shown in Figure 3, an existing approach such as that described in [26] offloads tasks to minimize the total completion time but does not account for traffic lights. Since the total completion time for τ_i if uploaded, computed, and downloaded at RSU_1 is 3.4 seconds (as per task characteristics), which is less than δ_{i1}, RSU_1 will be selected for upload, compute, and download. When τ_{i+1} arrives, the only available RSU to offload is RSU_m. Offloading τ_{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 τ_{i+1}. Therefore, with an existing strategy, only τ_i task will be offloaded onto the RSU.

4.2.2 Offloading with Deadline-Based Approach. When τ_i arrives at t_0, $\psi_{s_i} = red$ 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 τ_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 τ_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 τ_i, even though uploading, computing, and downloading τ_i onto RSU_m incurs a higher completion time (28.4 seconds, including travel time). Note that τ_i still meets its deadline. When τ_{i+1} arrives at t_1, RSU_1 will be available. Hence, τ_{i+1} will be uploaded, computed, and downloaded on RSU_1. Therefore, with our deadline-based, just-in-time task-offloading strategy, (i) we maximize the number of tasks that can be offloaded onto the RSUs and (ii) meet deadlines for all offloaded tasks, see Figure 4.

In a typical urban traffic environment with multiple traffic intersections, traffic conditions evolve continuously. In a connected environment, real-time traffic data are made available to monitor and predict as vehicular traffic evolves. We have shown in Section 3 that in VEC 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 recomputation of tasks in case the vehicle's state changes after uploading the task. Additionally, since most VEC tasks utilize data from vehicles and their surroundings delaying processing closer to the

deadline enables utilization of the most recent data. Furthermore, prior works that utilize just-in-time scheduling in IoT applications have shown better bandwidth and resource utilization, with increased QoS [28]. When network conditions are known, approaches with just-in-time scheduling have shown reduced deadline miss ratio and packet drop ratio [24]. Network routing and vehicular traffic routing draw many similarities in their functions [47], 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 light 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, thereby maximizing the number of tasks being offloaded.

We break down the task offloading strategy deployed over the 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. However, our model can be extended to multiple lanes and traffic lights. The MBS acquires the traffic light data consisting of its current state (ψ_s), remaining time in current state (t_s^{rem}), and maximum times for green, yellow, and red phases, that is, t_s^g, t_s^y, and t_s^r, respectively. Based on the driving model and its maneuvers — that is, (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.

$$t_f = (v_f - v_0)/\alpha, \tag{2}$$

$$x_f = v_0 t_0 - 0.5\alpha t_0^2, \tag{3}$$

$$v_f^2 = v_0^2 + 2\alpha x_0 \tag{4}$$

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

$$x_i^{stop} = (0 - (v^{lim})^2)/(-2b), \tag{5}$$

$$t_i^{stop} = (0 - v^{lim})/(-b). \tag{6}$$

In a typical traffic infrastructure, the yellow light time t_s^y is tuned such that $t_s^y = t_i^{stop}$, which allows vehicles to safely decelerate and come to a halt upon encountering a traffic light. Any vehicle that is more than x_i^{stop} distance away from the traffic light will consider the yellow phase as red and would decelerate to come to a stop. Alternately, vehicles less than x_i^{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 be influenced only by the traffic light or another leading vehicle in the lane. To simplify the formulation of the mobility characteristics, we first consider the case in which 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^{stop} and t_i^{stop}, respectively, will drive at the speed limit v^{lim} until the vehicle is x_i^{stop} distance and correspondingly t_i^{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^{const} and the corresponding time t_i^{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 Equations (7) and (8).

$$x_i^{const} = d_L - x_i^{stop}, \tag{7}$$

$$t_i^{const} = x_i^{const}/v^{lim}. \tag{8}$$

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 entering an empty link L of length d_L with a traffic light s and bound by the acceleration and deceleration rate of a and $-b$, respectively, will traverse the entire link L at the speed of v^{lim} if one of the following conditions are satisfied: (i) $\psi_s = red$ and $t_s^{rem} \leq t_i^{const}$ or (ii) $\psi_s = green$ and $t_s^{rem} > t_i^{const}$. Here, ψ_s and t_s^{rem} are the light state and remaining time in current state of traffic light s, and t_i^{const} 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 entering an empty link L of length d_L with a traffic light s and 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) $\psi_s = red$ and $t_s^{rem} > t_i^{const} + t_i^{stop}$ or (ii) $\psi_s = green$ and $t_s^{rem} \leq t_i^{const}$. Here, ψ_s and t_s^{rem} denote the light state and remaining time in current state of traffic light s, t_i^{stop} denotes the duration needed for V_i to come to a complete stop at the traffic light, and t_i^{const} 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 entering an empty link L of length d_L with a traffic light s and bound by the acceleration and deceleration rate of a and $-b$, respectively, will initially decelerate for t_i^{dec} time and then accelerate for t_i^{acc} time to cross the traffic light without coming to a halt if $\psi_s = red$ and $t_i^{const} < t_s^{rem} < t_i^{const} + t_i^{stop}$. Here, ψ_s and t_s^{rem} denote the light state and remaining time in current state of traffic light s, t_i^{stop} denotes the duration needed for V_i to come to a complete stop at*

the traffic light, and t_i^{const} denotes the time for which V_i drives at a constant speed before decelerating at the traffic light.

Proof. Upon V_i's arrival, $\psi_s = red$ and t_s^{rem} is the time after which the light turns green. If $t_s^{rem} > t_i^{const}$, then the vehicle starts to decelerate at the red light when it is x_i^{stop} distance away from the traffic light. However, if $t_s^{rem} < t_i^{const} + t_i^{stop}$, then the traffic light will turn green before the vehicle comes to a complete stop and starts accelerating. Therefore, the vehicle only decelerates for $t_i^{dec} = t_s^{rem} - t_i^{const}$ time. As the light turns green, the vehicle starts accelerating at the rate of a. The speed achieved after deceleration is denoted by $v_i^{dec} = v^{lim} - t_i^{dec} \cdot b$ (Equation (2)). The distance travelled during deceleration is given by $x_i^{dec} = (v_i^{dec})^2 - (v^{lim})^2/(-2b)$ (Equation (4)). Now, the vehicle will continue accelerating until either it reaches maximum speed limit, that is, for $t_i^{acc} = (v^{lim} - v_i^{dec})/a$ duration, or crosses the traffic light, that is, upon traveling $x_i^{acc} = d_L - x_i^{const} - x_i^{dec}$, achieving a speed of $v_i^{acc} = \sqrt{(v_i^{dec})^2 + 2 \cdot a \cdot x_i^{acc}}$ (Equation (4)) and the corresponding time will be $t_i^{acc} = (v_i^{acc} - v_i^{dec})/a$ (Equation (2)), whichever is shorter.

Therefore, vehicle V_i drives at the speed limit for t_i^{const} time, then decelerating for $t_i^{dec} = t_s^{rem} - t_i^{const}$ time followed by acceleration for $t_i^{acc} = \min((v^{lim} - v_i^{dec})/a, (v_i^{acc} - v_i^{dec})/a)$ time. □

Before discussing Case 2, in which the vehicle's mobility is also influenced by the leading vehicle in front, we present the algorithm used to estimate the dwell times in a scenario such as Case 1 in which 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 dwell times for each RSU. 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 light data from traffic light s, that is, current state (ψ_s), remaining time in current state (t_s^{rem}), maximum red phase time (t_s^r), and maximum green phase time (t_s^g); (ii) vehicle and link data, that is, speed limit (v^{lim}) and length (d_L) of link L, index i for vehicle V_i, and location-based deadline d_i^x 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 ($r_j, j = 1, \ldots, m$). We consider all RSUs to have equal communication radii; therefore, $r_j = r, \forall j \in 1, \ldots, m$. 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. It also finds the time at which the location-based deadline will be met (d_i^t), 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 τ_i, the MBS receives the relevant data from the infrastructure *(Lines 1–3)*. The MBS is aware of the driving controller on V_i and hence t_{step}, which determines the time granularity at which the control decisions are taken by the driving controllers *(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) *(Line 6)*. Then, until the vehicle is estimated to cross the traffic light *(Line 7)*, the MBS, using the driving maneuver model, decides whether to perform a constant speed, deceleration, or acceleration maneuver *(Lines 8–15)* and then calculates the corresponding acceleration, speed, and position for the next time step *(Line 16)*. During the entire calculation process, the MBS also accumulates the time (T_δ), and assigns the dwell time δ_{ij} for RSU_j when the calculated position exceeds the coverage boundary of an RSU, after which T_δ is reset *(Lines 17–19)*. 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_i 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 *(Lines 20 and 21)*. As the calculated driving time and distance satisfies Property 1 *(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 *(Lines 23–34)*. Based on the condition, the MBS then selects the appropriate driving maneuver ($flag_s$) influenced by the traffic light s and also sets the time check points, $t_i^{stop}, t_i^{dec}, t_i^{acc}$, and t_i^w when the MBS must change the maneuvering calculations *(Lines 35–43)*. The chosen maneuver $flag_s$ is then selected as an *action* for the next iteration. Calculations for a vehicle V_i end when its calculated position exceeds the length of the link. The algorithm then provides the dwell times of vehicle V_i in every RSU along the link as an output.

Now, let us consider a more complex situation in which the link is non-empty and there may be leading vehicles that will influence vehicle V_i's mobility.

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 in which there will be existing vehicles in the link that vehicle V_i enters. We refer to a vehicle immediately in front of V_i, that is, V_{i-1} as *lead vehicle*, and the vehicle under consideration, that is, V_i as *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. We highlight that an ego vehicle maintains a safety distance of $x_i^{safe} = hv_i + x_{safe}$ 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_{safe} is the minimum distance between two vehicles at standstill. We now define the following Property 2.

Property 2 (Case 2). *An ego vehicle V_i entering a non-empty link L of length d_L has at least one vehicle V_{i-1} between itself and the traffic light s at the end of the link L. V_i has a safe stopping distance and time of x_i^{stop} and t_i^{stop}, respectively, and a safety gap of $x_i^{safe} = hv_i + x_{safe}$ from the leading vehicle. In this case, the ego vehicle will drive at a constant speed of v^{lim} until either of these conditions is met: (i) the ego vehicle is x_i^{safe} distance behind the lead vehicle V_{i-1} or (ii) the ego vehicle is within x_i^{stop} distance and, correspondingly, t_i^{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^{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 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 *action* suggested as per the traffic light using $flag_s$, the following conditions also need to be checked to maintain a safe distance from the lead vehicle *(Line 2)*. The MBS first acquires the estimated positions of the lead vehicle from its previous calculations *(Line 3)*. If the ego vehicle is x_i^{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_i^{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_i^{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 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 τ_i of vehicle V_i based on its deadline d_i^t. Note that we also determine the RSU where the vehicle will be when its task deadlines are met while estimating the mobility characteristics.

ALGORITHM 4.1: Mobility Characteristics Estimation and Dwell Time Assignment for Vehicles Entering an Empty Link L with No Vehicles in Front (Case 1)

Lights Data ψ_s, t_s^{rem}, t_s^r, t_s^g
Vehicle Data v^{lim}, d_L, V_i, d_i^x
RSU Data r, m
Result: $\{\delta_{i1}, \ldots, \delta_{im}\}$, RSU_{end}, t_i^{end}, d_i^t
Init: $T_i \leftarrow 0$, $X_i \leftarrow 0$, $j \leftarrow 1$, $T_\delta \leftarrow 0$, $v_i \leftarrow v^{lim}$, $t_{step} \leftarrow 1$
Init: $action \leftarrow flag_s$ (Algorithm 4.1) or $flag_{act}$ (Algorithm 4.2)
Init: x_i^{stop}, t_i^{stop}, x_i^{const}, t_i^{const}
/* X_i: variable to track distance traveled by i^{th} vehicle, until the vehicle crosses the link (d_L). */
while $X_i \leq d_L$ **do**
  if $action = constant\ speed$ **then**
    $x_{step} = v_i \times t_{step}$
  if $action = decelerate$ **then**
    In Equations (2) and (4), $\alpha \leftarrow -b$, $t_0 \leftarrow t_{step}$
    Get v_{step} and x_{step}
  if $action = accelerate$ **then**
    In Equations (2) and (4), $\alpha \leftarrow a$, $t_0 \leftarrow t_{step}$
    Get v_{step} and x_{step}
  $X_i \leftarrow X_i + x_{step}$, $T_i \leftarrow T_i + t_{step}$, $T_\delta \leftarrow T_\delta + t_{step}$
  if $X \geq 2r$ **then** // end of range for j^{th} RSU
    $\delta_{ij} \leftarrow T_\delta$, $j \leftarrow j + 1$ // assign dwell time
    $T_\delta \leftarrow 0$ // reset counter
  if $X_i = d_i^x$ *or* $T_i = d_i^t$ **then**
    /* deadline expected to be met */
    $d_i^t \leftarrow T_i$, $RSU_{end} \leftarrow j$, $t_i^{end} \leftarrow T_\delta$
  if $T_i = t_i^{const}$ **then**
    /* check conditions for Lemma 4.1, 4.2, and 4.3 */
    if *Lemma 4.1 condition (i) or (ii) met* **then**
      $flag_s \leftarrow constant\ speed$
    if *Lemma 4.2 condition (i) met* **then**
      Set wait time $t_i^w \leftarrow t_s^{rem} - t_i^{const} - t_i^{stop}$
      $flag_s \leftarrow decelerate$
    if *Lemma 4.2 condition (ii) met* **then**
      Set wait time $t_i^w \leftarrow t_s^r - (t_i^{const} + t_i^{stop} - t_s^{rem})$
      $flag_s \leftarrow decelerate$
    if *Lemma 4.3 condition met* **then**
      Set deceleration time t_i^{dec}
      Set wait time $t_i^w \leftarrow 0$
      $flag_s \leftarrow decelerate$
  if $T_i = t_i^{dec}$ **then**
    if $t_i^w = 0$ **then**
      Set acceleration time t_i^{acc}
      $flag_s \leftarrow accelerate$
    else
      /* wait time as RSU j's dwell time */
      $\delta_{ij} \leftarrow \delta_{ij} + t_i^w$
      break
if $T_i = t_i^{acc}$ **then**
  /* speed limit reached */
  $flag_s \leftarrow constant\ speed$
$action \leftarrow flag_s$
/* call Algorithm 4.2 */
Algorithm 4.2()

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)

Init: h : time headway, x_{safe} : standstill distance
$x_i^{safe} \leftarrow hv_i + x_{safe}$
$X_{i-1}^{T_i} \leftarrow$ estimated position of V_{i-1} at T_i time
/* monitoring the gap with lead vehicle */
if $X_i - X_{i-1}^{T_i} = x_i^{safe}$ **then**
  $flag_{lead} \leftarrow constant\ speed$
if $X_i - X_{i-1}^{T_i} < x_i^{safe}$ **then**
  $flag_{lead} \leftarrow decelerate$
if $X_i - X_{i-1}^{T_i} > x_i^{safe}$ **then**
  if $v_i < v^{lim}$ **then**
    $flag_{lead} \leftarrow accelerate$
    /* lead vehicle does not influence ego vehicle actions. */
  else
    $flag_{lead} \leftarrow constant\ speed$
/* $action$ influenced by traffic light or lead vehicle */
if $flag_s = decelerate\ or\ flag_{lead} = decelerate$ **then**
  $flag_{act} \leftarrow decelerate$
else
  if $flag_s = constant\ speed\ or\ flag_{lead} = constant\ speed$ **then**
    $flag_{act} \leftarrow constant\ speed$
$action \leftarrow flag_{act}$

4.4 RSU Allocation for Task Offloading

We know that a task τ_i needs to be completed within d_i^t time of the task's arrival. We also determined d_i^t corresponding to the distance-based deadline (d_i^x) 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 $RSU_1, \ldots, RSU_{end}$. 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 τ_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 at least equal to the download time of the task τ_i. The download time t_i^{dl} of task τ_i from any RSU_j depends on the download size σ_i^{dl} of the task and the communication data rate η 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.

$$\text{maximize } d \tag{9}$$

$$\text{subject to } \delta_{id} \geq t_i^{dl}, \tag{9a}$$

$$1 \leq d \leq \text{end}. \tag{9b}$$

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 at least equal to the download time t_i^{dl} of the task. *Constraint (9b):* The index d of the chosen RSU must be less than *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 τ_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 ρ_{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 τ_i, that is, $\delta_{id} - t_i^{dl} \geq t_i^c$, 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.

$$\text{maximize } c \tag{10}$$

$$\text{subject to } \sum_{j=c}^{d} \delta_{ij} \geq t_i^c + \rho_{jd} + t_i^{dl}, \tag{10a}$$

$$\rho_{dd} = 0, \tag{10b}$$

$$p_c \geq 1, \tag{10c}$$

$$1 \leq c \leq d. \tag{10d}$$

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 at least equal to the sum of the compute time t_i^c, relay time ρ_{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 that 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 at least one processor available at RSU_c to compute task τ_i. *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 to meet its deadline. The upload time t_i^{ul} of task τ_i from any RSU_j depends on the upload size σ_i^{ul} of the task and the communication data rate η (as per the Shannon Hartley theorem [33]) such that $t_i^{ul} = \eta \cdot \sigma_i^{ul}$. Therefore, RSU_u is chosen using Equation (9) and its constraints.

$$\text{maximize } u \tag{11}$$

$$\text{subject to } \delta_{iu} \geq t_i^{ul} + \rho_{uc}, \tag{11a}$$

$$\rho_{cc} = 0, \tag{11b}$$

$$1 \leq u \leq c. \tag{11c}$$

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. *Constraint(11c):* The task must be uploaded before it is computed.

By solving Equations (9) to (11) using linear optimization techniques, RSU_u, RSU_c, and RSU_d can be determined. Once the RSUs are assigned, the vehicle can 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, the vehicle enters the communication range of RSU_d

Table 1. Simulation Parameters

Parameter	Values
Traffic Infrastructure	
Link length d_L	[400,800] m
Speed limit v^{lim}	[25,35] mph
Number of lanes l	1
Traffic Light s	
Cycle time t_s^{cycle}	60 s
Max. green time t_s^g	26 s
Yellow time t_s^y	4 s
Max. red time t_s^r	30 s
Vehicles	
Comfortable acceleration a	1 m/s^2
Comfortable deceleration $-b$	-0.9 m/s^2
Time headway h	0.9 s
Standstill gap x_{safe}	1.5 m
VEC Task Model	
Task type	[time-based, location-based]
Upload size σ^{ul}	1–100 MB
Compute requirement C	0.24–24 $\times$ 10^9 clock cycles
Download size σ^{dl}	1–100 MB
Time-based deadline d^t	100 ms–5s
Distance-based deadline d^x	1 m–d_L m
VEC Communication Framework	
Communication radius r	25 m
Number of processors	[2, 4, 8]
Clock speed z	2.4 GHz
Wireless data rate η	500 Mbps
Backhaul data rate η_{bh}	1 Gbps
Backhaul relay constant c_{bh}	10 ms

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 (Algorithm 4.1) has a time complexity of $O(T \times R)$, where V denotes total task-offloading requests in the 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 a 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 contains the parameters for traffic infrastructure, vehicles, task model, communication model, and the VEC framework, It also summarizes the notations used in this article. These

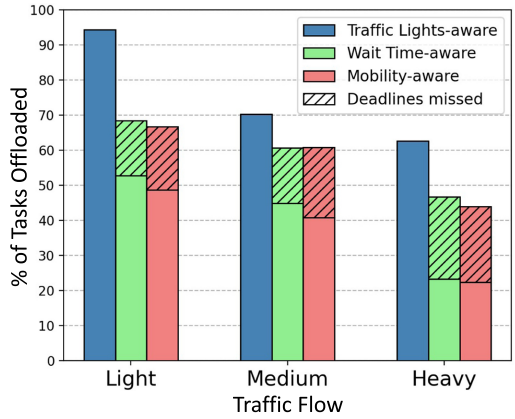

Fig. 5. Effect of varying traffic flow on task offloading. Increasing traffic flow leads to a reduction in the percentage of tasks offloaded out of total task requests (colored). However, the existing approaches cause a significant number of tasks to miss their deadlines (hatched).

parameters closely resemble common VEC workloads and network and traffic infrastructure configurations [26, 52]. 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 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 article. An external communication framework emulator based on Python 3 acts as an 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 [26] (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. We also compare our work with a mobility-aware task-offloading strategy [52] (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 [26] and mobility-aware [52] approaches offload tasks onto the RSUs only if the task is expected to be completed before its deadline.

5.2 Performance in an Ideal Setting

We evaluate all 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 task deadlines, and (ii) the percentage of tasks that miss their deadlines out of all of the tasks offloaded onto the RSUs. In this set of experiments, we assume that the vehicles and 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 (400–1,000 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 these changing traffic patterns and lead to deadline misses. The wait time-aware approach offloads 47% of all task requests and misses the deadlines for 23% of the

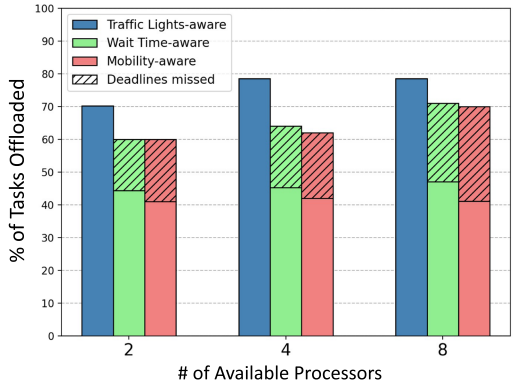

Fig. 6. Effect of increasing processor availability on task offloading. With increase in the number of processors all approaches offload more tasks onto the RSUs (colored), however, the wait time-aware and mobility-aware strategies lead to a significant number of tasks missing their deadlines (hatched).

offloaded tasks, whereas the mobility-aware approach finds 44% of the tasks feasible to offload and misses the deadlines for 22% of the offloaded tasks. Location-based tasks comprise 53% and 59% of the offloaded tasks that missed their deadlines in time delay-aware and mobility-aware approaches, respectively.

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 an increase in the number of available processors at each RSU, the number of tasks offloaded for all three approaches increases. However, it is important to note that as the number of available processors increases at each RSU, the existing approaches also show an increase in deadline misses (up to 25% of deadlines missed with 8 parallel processors). As the number of available processors at each RSU increases, 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 task execution completes, 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, maximizing RSU resource utilization. We are not including a discussion on the effect of increasing link length as it also leads to an increase in the number of RSUs and processor availability, and similar results were obtained.

Effect of speed limit on task offloading (Figure 7): We choose two speed limits common for urban traffic lights: 25 mph and 35 mph. 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, 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. In contrast, 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, an increase in speed also causes an increase in the number of deadline misses for the wait time-aware approach due to an increase in dwell time estimation errors. Similarly, the mobility-aware offloading also leads to increased deadline misses as it relies on the average vehicle speeds. Note that with an increase in processor availability, a similar trend as in Figure 6 was observed, in which at higher speeds, more processor availability meant an increase in the number of tasks offloaded but it also led to an increase in deadline misses.

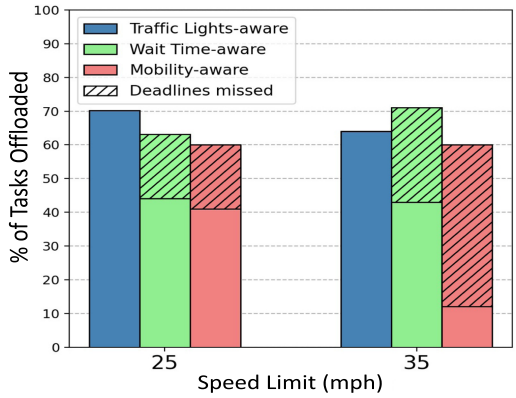

Fig. 7. Effect of changing speed limits on task offloading. Increases to the speed limit lead to a change in the percentage of tasks offloaded (colored), with a significant increase in the deadline misses (hatched) for the existing strategies.

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 follows predefined constrained motion such as constant speeds, reducing speeds with known deceleration rates, and 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 that 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}. Instead of the vehicles following strict acceleration and deceleration rates, they are now modelled after the Weidemann 74 car-following model [32]. In this work, the Weidemann 74 model is preferred over other car-following models as it accurately captures non-ideal driving conditions in which 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 m/s^2, and deceleration ranging from -2.5 to -3 m/s^2. These parameter ranges are determined based on driver reaction times and passenger comfort [32].

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 (Figure 8) and (ii) traffic flow (Figure 9) on deadline misses in our proposed approach.

Speed limit and deadline misses in non-ideal conditions: As expected (from Figure 7), increasing speed limits does cause a reduction in the tasks successfully offloaded onto the edge platform. Under realistic conditions, however (Figure 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 increase in stop-and-go motions and variations in desired speed and acceleration. Therefore, as shown in Figure 9, under non-ideal conditions, low traffic (in which vehicles can travel at higher speeds) or heavy traffic (in which vehicles are known to show increased stop-and-go motions) leads to increased (up to 10%) deadline misses.

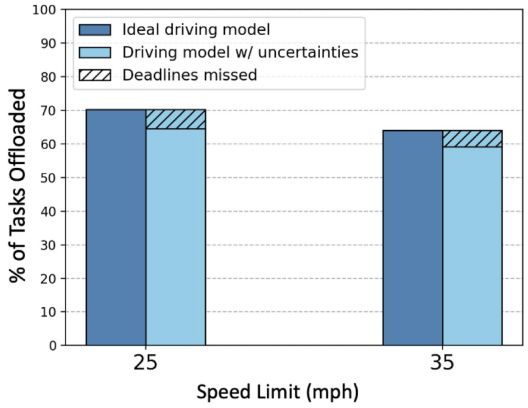

Fig. 8. Effect of speed on deadline misses in our proposed approach.

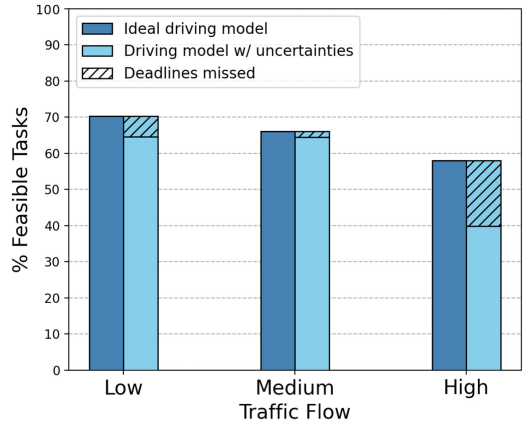

Fig. 9. Effect of traffic flow on deadline misses in our proposed approach.

Under medium traffic, all vehicles have a more regularized traffic flow and, therefore, experience very few (2%) deadline misses.

Effect of model (speed, acceleration, and car-following model) uncertainties on deadline misses (Figure 10): Finally, we investigate the impact on non-ideal conditions in existing approaches when compared with our proposed approach. The uncertainty in speed, acceleration, and following distance added in a realistic car-following model are not known *a priori*. Therefore, they 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. Because of this, deadline misses are expected due to the model being an ideal version 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 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.

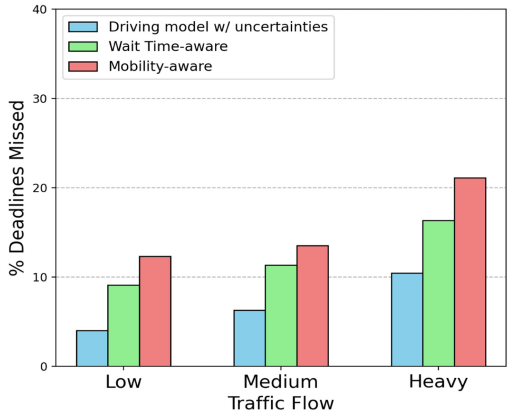

Fig. 10. Effect of model uncertainties on deadline misses.

6 CONCLUSION

In this article, we presented a comprehensive task model to support emerging VEC applications that incorporates tasks with fixed time-based deadlines as well as traffic-dependent tasks whose deadlines depend on traffic flow, location, and traffic lights. We then incorporated the traffic light data by leveraging the connected infrastructure to accurately estimate the mobility characteristics and dwell times of the vehicles. We then used the dwell time estimates 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

[1] Shimaa A. Abdel Hakeem, Anar A. Hady, and HyungWon Kim. 2020. 5G-V2X: Standardization, architecture, use cases, network-slicing, and edge-computing. *Wireless Networks* (2020).

[2] Ta Huu Binh, Hiep Khac Vo, Binh Minh Nguyen, Huynh Thi Thanh Binh, Shui Yu, et al. 2022. Value-based reinforcement learning approaches for task offloading in delay constrained vehicular edge computing. *Engineering Applications of Artificial Intelligence* 113 (2022), 104898.

[3] Azzedine Boukerche and E. Robson. 2018. Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks* 135 (2018), 171–189.

[4] Gaofeng Cui, Xiaoyao Li, Lexi Xu, and Weidong Wang. 2020. Latency and energy optimization for MEC enhanced SAT-IoT networks. *IEEE Access* 8 (2020), 55915–55926.

[5] Yueyue Dai and Yan Zhang. 2022. Adaptive digital twin for vehicular edge computing and networks. *Journal of Communications and Information Networks* 7, 1 (2022), 48–59.

[6] Stavroula Panagiota Deligianni, Mohammed Quddus, Andrew Morris, Aaron Anvuur, and Steven Reed. 2017. Analyzing and modeling drivers' deceleration behavior from normal driving. *Transportation Research Record* (2017).

[7] Jianbo Du, Liqiang Zhao, Xiaoli Chu, F. Richard Yu, Jie Feng, and I. Chih-Lin. 2018. Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems. *IEEE Transactions on Vehicular Technology* 68, 2 (2018), 1757–1771.

[8] Angela H. Eichelberger and Anne T. McCartt. 2016. Toyota drivers' experiences with dynamic radar cruise control, pre-collision system, and lane-keeping assist. *Journal of Safety Research* (2016).

[9] Wenhao Fan, Yi Su, Jie Liu, Shenmeng Li, Wei Huang, Fan Wu, and Yuan'an Liu. 2023. Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes. *IEEE Transactions on Intelligent Transportation Systems* (2023).

[10] Martin Fellendorf and Peter Vortisch. 2010. Microscopic traffic flow simulator VISSIM. In *Fundamentals of Traffic Simulation*. Springer.
[11] James Fowe. 2021. Method, Apparatus, and Computer Program Product for Lane-level Route Guidance. US Patent 11,022,457.
[12] Mercedes-Benz Group. 2021. First Internationally Valid System Approval for Conditionally Automated Driving: Mercedes-Benz Group. Retrieved May 1, 2023 from https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/system-approval-for-conditionally-automated-driving.html.
[13] Hongzhi Guo, Jiajia Liu, Ju Ren, and Yanning Zhang. 2020. Intelligent task offloading in vehicular edge computing networks. *IEEE Wireless Communications* 27, 4 (2020), 126–132.
[14] Takamasa Higuchi, Seyhan Ucar, and Onur Altintas. 2021. Cooperative Parking Space Search by a Vehicular Micro Cloud. US Patent 10,896,609.
[15] Baik Hoh, Seyhan Ucar, Pratham Oza, Chinmaya Patnayak, and Kentaro Oguchi. 2020. CORR: Collaborative on-road reputation. In *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS'20)*.
[16] Xumin Huang, Rong Yu, Dongdong Ye, Lei Shu, and Shengli Xie. 2021. Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing. *IEEE Transactions on Vehicular Technology* (2021).
[17] Nathaniel Hudson, Hana Khamfroush, and Daniel E. Lucani. 2021. QoS-aware placement of deep learning services on the edge with multiple service implementations. In *2021 International Conference on Computer Communications and Networks (ICCCN'21)*. IEEE.
[18] Nathaniel Hudson, Pratham Oza, Hana Khamfroush, and Thidapat Chantem. 2022. Smart edge-enabled traffic light control: Improving reward-communication trade-offs with federated reinforcement learning. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP'22)*. 40–47.
[19] Hasan Ali Khattak, Haleem Farman, Bilal Jan, and Ikram Ud Din. 2019. Toward integrating vehicular clouds with IoT for smart city services. *IEEE Network* 33, 2 (2019), 65–71.
[20] Yu-Jen Ku and Sujit Dey. 2019. Sustainable vehicular edge computing using local and solar-powered roadside unit resources. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE.
[21] Thomas Lagkas, Dimitrios Klonidis, Panagiotis Sarigiannidis, and Ioannis Tomkos. 2021. Optimized joint allocation of radio, optical, and MEC resources for the 5G and beyond fronthaul. *IEEE Transactions on Network and Service Management* 18, 4 (2021), 4639–4653.
[22] Yiran Li, Hongwei Li, Guowen Xu, Tao Xiang, and Rongxing Lu. 2022. Practical privacy-preserving federated learning in vehicular fog computing. *IEEE Transactions on Vehicular Technology* 71, 5 (2022), 4692–4705.
[23] Bing Lin, Fangning Zhu, Jianshan Zhang, Jiaqing Chen, Xing Chen, Naixue N. Xiong, and Jaime Lloret Mauri. 2019. A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. *IEEE Transactions on Industrial Informatics* (2019).
[24] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang. 2006. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *4th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*. 5 pp.–46.
[25] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. 2021. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications* (2021).
[26] Pengju Liu, Junluo Li, and Zhongwei Sun. 2019. Matching-based task offloading for vehicular edge computing. *IEEE Access* (2019).
[27] Wencan Mao, Ozgur Umut Akgul, Abbas Mehrabi, Byungjin Cho, Yu Xiao, and Antti Ylä-Jääski. 2022. Data-driven capacity planning for vehicular fog computing. *IEEE Internet of Things Journal* 9, 15 (2022), 13179–13194.
[28] John L. Messenger. 2018. Time-sensitive networking: An introduction. *IEEE Communications Standards Magazine* 2, 2 (2018), 29–33.
[29] Di Natale et al. 1994. Dynamic end-to-end guarantees in distributed real time systems. In *1994 Proceedings of Real-Time Systems Symposium*. IEEE, 216–227.
[30] Gerrit J. L. Naus, Rene P. A. Vugts, Jeroen Ploeg, Marinus J. G. van De Molengraft, and Maarten Steinbuch. 2010. String-stable CACC design and experimental validation: A frequency-domain approach. *IEEE Transactions on Vehicular Technology* (2010).
[31] Pratham Oza, Thidapat Chantem, and Pamela Murray-Tuite. 2020. A coordinated spillback-aware traffic optimization and recovery at multiple intersections. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'20)*.
[32] Sangjun Park and Hesham Rakha. 2010. Continuous flow intersections: A safety and environmental perspective. In *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 85–90.
[33] Eric Price and David P. Woodruff. 2012. Applications of the Shannon-Hartley theorem to data streams and sparse recovery. In *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE.

[34] Guanhua Qiao, Supeng Leng, Ke Zhang, and Yejun He. 2018. Collaborative task offloading in vehicular edge multi-access networks. *IEEE Communications Magazine* (2018).
[35] Peng Qin, Yang Fu, Guoming Tang, Xiongwen Zhao, and Suiyan Geng. 2022. Learning based energy efficient task offloading for vehicular collaborative edge computing. *IEEE Transactions on Vehicular Technology* 71, 8 (2022), 8398–8413.
[36] SAE International. 2021. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_202104. Retrieved May 1, 2023 from https://www.sae.org/standards/content/j3016_202104/.
[37] Syed Danial Ali Shah, Mark A. Gregory, Shuo Li, Ramon dos Reis Fontes, and Ling Hou. 2022. SDN-based service mobility management in MEC-enabled 5G and beyond vehicular networks. *IEEE Internet of Things Journal* 9, 15 (2022), 13425–13442.
[38] Bodong Shang, Lingjia Liu, and Zhi Tian. 2021. Deep learning-assisted energy-efficient task offloading in vehicular edge computing systems. *IEEE Transactions on Vehicular Technology* (2021).
[39] Joseph A. Shaw. 2013. Radiometry and the Friis transmission equation. *American Journal of Physics* (2013).
[40] Muhammad Sameer Sheikh, Jun Liang, and Wensong Wang. 2020. Security and privacy in vehicular ad hoc network and vehicle cloud computing: A survey. *Wireless Communications and Mobile Computing* 2020 (2020), 1–25.
[41] Qiaoqiao Shen, Bin-Jie Hu, and Enjun Xia. 2022. Dependency-aware task offloading and service caching in vehicular edge computing. *IEEE Transactions on Vehicular Technology* 71, 12 (2022), 13182–13197.
[42] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. 2020. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology* (2020).
[43] Mohammad Shojafar, Nicola Cordeschi, and Enzo Baccarelli. 2016. Energy-efficient adaptive resource management for real-time vehicular cloud services. *IEEE Transactions on Cloud Computing* 7, 1 (2016), 196–209.
[44] Yushan Siriwardhana, Pawani Porambage, Madhusanka Liyanage, and Mika Ylianttila. 2021. A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1160–1192.
[45] Darbha Swaroop and K. R. Rajagopal. 2001. A review of constant time headway policy for automatic vehicle following. In *Proceedings of 2001 IEEE Intelligent Transportation Systems (ITSC'01) (Cat. No. 01TH8585)*. IEEE.
[46] Seyhan Ucar, Chinmaya Patnayak, Pratham Oza, Baik Hoh, and Kentaro Oguchi. 2019. Management of anomalous driving behavior. In *2019 IEEE Vehicular Networking Conference (VNC'19)*. IEEE.
[47] Zhen Wang, Sifa Zheng, Qiang Ge, and Keqiang Li. 2020. Online offloading scheduling and resource allocation algorithms for vehicular edge computing system. *IEEE Access* (2020).
[48] Wenting Wei, Ruying Yang, Huaxi Gu, Weike Zhao, Chen Chen, and Shaohua Wan. 2021. Multi-objective optimization for resource allocation in vehicular cloud computing networks. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (2021), 25536–25545.
[49] Chao Yang, Baichuan Liu, Haoyu Li, Bo Li, Kan Xie, and Shengli Xie. 2022. Learning based channel allocation and task offloading in temporary UAV-assisted vehicular edge computing networks. *IEEE Transactions on Vehicular Technology* 71, 9 (2022), 9884–9895.
[50] Chao Yang, Yi Liu, Xin Chen, Weifeng Zhong, and Shengli Xie. 2019. Efficient mobility-aware task offloading for vehicular edge computing networks. *IEEE Access* (2019).
[51] Jie Zhang, Hongzhi Guo, Jiajia Liu, and Yanning Zhang. 2019. Task offloading in vehicular edge computing networks: A load-balancing solution. *IEEE Transactions on Vehicular Technology* (2019).
[52] Yifan Zhang, Xiaoqi Qin, and Xianxin Song. 2020. Mobility-aware cooperative task offloading and resource allocation in vehicular edge computing. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW'20)*. IEEE.

Received 9 November 2022; revised 1 March 2023; accepted 19 April 2023