Assisting Vehicles Automated Locomotion via ML-Aided Edge-Twin Paradigm

Tianzi Yang

Department of Computer Science
McGill University
Montréal, Québec, Canada

August 15, 2023

A thesis presented for the degree of Masters of Computer Science

Abstract

In response to escalating traffic demands, the innovative centralized Edge Twin (ET) paradigm offers a significant shift from traditional traffic management systems. This research explores the ET's capacity for real-time cooperative driving decisions and advanced traffic oversight from an external perspective. By developing a novel graph-based data architecture specifically designed for our scheduling platform, we've broadened the scope of the ET model to encompass a variety of traffic scenarios, including highways and non-signalized intersections. Using the computational power of edge servers and a vast traffic data repository, our focus is on understanding unique traffic patterns to improve decision-making and traffic flow. For non-signalized intersections, we introduce a groundbreaking Machine Learning (ML)-aided priority policy that accounts for immediate and anticipated traffic conditions, significantly boosting traffic efficiency. On the highway front, our model's core lies in data analytics. We construct a step-by-step predictive model that anticipates the behavior of non-connected, traditional human-driven vehicles. This model is crucial for encouraging cooperative driving between Connected and Autonomous

Vehicles (CAVs) and their non-connected vehicles. Lastly, validation through the CARLA simulator underscores these advancements, revealing up to 16% enhancement in traffic flow on congested highways compared to conventional methods and up to 8.5% improvement at non-signalized intersections relative to prior research.

Abrégé

En réponse à l'escalade des exigences de trafic, le modèle innovant Centralized Edge Twin (ET) propose un changement significatif par rapport aux systèmes traditionnels de gestion du trafic. Cette recherche explore la capacité de l'ET à prendre des décisions de conduite coopérative en temps réel et à exercer une supervision avancée du trafic depuis une perspective externe. En développant une architecture de données basée sur des graphes spécialement conçue pour notre plateforme de planification, nous avons élargi la portée du modèle ET pour englober une variété de scénarios de trafic, y compris les autoroutes et les intersections non signalisées. En utilisant la puissance de calcul des serveurs Edge et un vaste répertoire de données sur le trafic, notre objectif est de comprendre les modèles de trafic uniques pour améliorer la prise de décision et le flux de trafic. Pour les intersections non signalisées, nous introduisons une politique de priorité assistée par Machine Learning (ML) révolutionnaire qui prend en compte les conditions de trafic immédiates et anticipées, augmentant significativement l'efficacité du trafic. Sur le front des autoroutes, le cœur de notre modèle réside dans l'analyse de données. Nous construisons un modèle prédictif étape par étape qui anticipe le comportement des véhicules traditionnels non connectés. Ce modèle est crucial pour encourager la conduite coopérative entre les Véhicules Connectés et Autonomes (CAVs) et leurs homologues non connectés. Enfin, la validation à travers le simulateur CARLA souligne ces avancées, révélant jusqu'à 16 % d'amélioration du flux de trafic sur les autoroutes congestionnées par rapport aux méthodes conventionnelles et jusqu'à 8,5 % d'amélioration aux intersections non signalisées par rapport aux recherches précédentes.

Acknowledgements

As I reflect on the journey of crafting this dissertation, I am deeply grateful for the extensive support and assistance I've received. My foremost gratitude goes to my supervisor, Professor Muthucumaru Maheswaran. His expertise was invaluable in shaping the research questions and methodology. His insightful feedback was pivotal in refining my thought process and elevating the quality of my work.

I am also thankful for the collaboration and encouragement from my colleagues and friends. In particular, I want to highlight the sincere and supportive contributions of Ibrahim Sorkhoh and Xiru Zhu, whose suggestions and help were invaluable.

Moreover, a special thanks goes to my parents. Their wise advice and empathetic support have been a constant source of strength and comfort. Their presence has been an anchor through this academic voyage.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advancements in the realm of autonomous driving have been significant, attributed largely to the substantial improvements in artificial intelligence, wireless communication and computing technologies. Pioneers in the industry, like Tesla and Google, along with academic researchers, have significantly advanced the essential technologies such as computer vision, networking, signal processing and machine learning within this field [1]. These technological breakthroughs promise to deliver safer, more friendly, and efficient automated vehicles. Such advancement should help tackling tangible problems including traffic congestion, energy consumption and car accidents. However, a critical hurdle remains unresolved: autonomous vehicles must coexist with traditional, human-driven vehicles on public roads for a considerable time before a full transition to autonomous driving is feasible.

The coexistence of fully-automated, semi-automated and human-driven cars increases the failure risk, mainly because human-driven vehicles lack effective communication methods and a plausible response time. To mitigate the consequences of such coexistence, Edge Computing (EC) can provide a suite of solutions that assist the three kinds of cars to cooperatively accomplish on-the-road tasks safely and effectively. An EC agent can orchestrates the real-time cars interaction by providing a broader perspective for the vehicles and a long-term sequence of decisions that will make the car-to-car interplay more agile and robust.

Our research introduces a centralized Edge Twin (ET) which replicates real-world traffic scenarios within roadside edge servers. Our ET agent is equipped with ML predictors and path planning procedures, enabling it to simulate future traffic. By leveraging these simulations, the ET framework aims to provide real-time cooperative driving decisions, enhanced data processing, and advanced traffic surveillance. In earlier works, digital twin technology has shown significant potential in improving intelligent traffic management and Connected and Automated Vehicle (CAV) research, as supported by various studies such as those by Kumar (2018) and Deren (2021).

To develop this centralized ET paradigm, we propose a novel integrated data platform called Topology Graph (TG), for aggregating diverse traffic data. This platform encompasses road configurations and real-time vehicle movements and is tailored for diverse environments including non-signalized intersections and highways. our ET agent also leverages external sensors to provide perspectives distinct from in-car sensors, often yielding higher clarity.

This is particularly effective when integrated with AI-enhanced traffic cameras, as noted by Muthu [2]. These edge computing tools continuously monitor drivers and pedestrians, thus enriching the traffic data available for ET.

For future traffic simulation, we focus on two fronts. For human-driven vehicles, we employ ML predictors to anticipate driver intentions and project future vehicle locations within TG, capitalizing on the data amassed from external sensors. Conversely, for autonomous vehicles, we utilize reservation-based path planning procedures, which leverage TG to navigate free space and prevent collisions.

In our study, we assess the performance of a centralized ET paradigm in scenarios such as a non-signalized intersection and a highway, utilizing the CARLA simulator [3] for our assessment. We demonstrate that this system can improve the traffic flow at non-signalized intersections and foster cooperative driving among CAVs and human-driven vehicles on highways.

1.1 Thesis contributions

The main contribution of this work is the following:

- We introduce the Topology Map, denoted as TG, a graph-based data structure tailored for our scheduling platform. Drawing from unique geographical features and diverse traffic regulations, the TG offers adaptability across multiple traffic environments.

- We propose an ML-based policy that enhances traffic flow efficiency at a non-signalized intersection by analyzing historical traffic data patterns on each entry lane. With the predictive insights gained, we have devised a sophisticated priority strategy for vehicle scheduling. This approach strategically prioritizes vehicles by carefully evaluating their immediate and future traffic impact. Our evaluations underscore its superior performance, particularly at non-signalized intersections, accommodating both symmetrical and asymmetrical traffic flows. The simulation was conducted through the CARLA simulator.

- We provide a novel prediction model combined with TG to assist cooperative driving on highways by facilitating interaction between CAVs and conventional human-driven vehicles. By utilizing a sophisticated step-by-step prediction model, we can anticipate the positions of human-driven vehicles and integrate them into the TG. Then, employing a time-based path-searching algorithm, we efficiently allocate space for CAVs. Our approach markedly surpasses conventional traffic management methods in improving traffic efficiency. We conducted the experiment through the CARLA simulator.

1.1.1 Thesis organization

Chapter 2 provides background information on edge computing-based cooperative traffic scheduling and future traffic predictions. Chapter 3 introduces the overall architecture of the

Edge-Twin(ET) paradigm. Chapter 4 introduces the essential module in the ET paradigm, the path planning procedure, and the priority policy. Chapter 5 covers the fundamental architectures of deep-learning models we used and how they have been trained. Chapter 6 discusses how to implement the entire ET paradigm inside the CARLA simulator. Chapter 7 presented simulated experimental results at non-signalized intersections and highways. Chapter 8 presents related research of edge computing on traffic scheduling at non-signalized intersections and highways. Chapter 9 concludes the thesis with a discussion of future research for extending the presented work.

Chapter 2

Background

2.1 Edge computing based cooperative traffic scheduling

In modern traffic management, a centralized traffic scheduling system is enhanced by leveraging edge computing capabilities, particularly through Vehicle-to-Everything (V2X) communication. Typically, such systems operate on a reservation-driven mechanism. Within this framework, vehicles dynamically communicate their intent to traverse a specific region at particular times using V2X. The edge devices, placed strategically near intersections or critical regions, quickly process these requests due to their proximity and computational power. A central authority, aided by these edge devices or functioning as an overarching management system, evaluates and prioritizes these requests. This orchestration of movement schedules, powered by the swift processing capabilities of edge computing, ensures that safety and throughput are optimized.

2.1.1 Reservation-based scheduling algorithm

Ensuring collision-free movement is paramount for traffic scheduling, particularly within networks of autonomous vehicles. This can be achieved through the following four components:

1. **Vehicles as Agents**: Treat each vehicle as an agent with a distinct start and end point. In our research, this includes both CAVs and non-connected human-driven vehicles.

2. **Road Network as Grid**: Consider the road network as a grid, with waypoints as nodes. The Grid Box method divides an area into a grid of smaller boxes or cells. Each box represents a portion of the intersection that a vehicle might occupy at any given moment. In essence, the region becomes a two-dimensional grid, where each cell or box has unique coordinates.

3. **Time Expansion**: Integrate time into the grid box above, transforming it into a space-time grid.

4. **Avoiding Collisions**: Ensure that no two vehicles occupy the same node at the same time in the space-time grid.

The A* algorithm is a widely used pathfinding and graph traversal method known for its efficiency and accuracy [4]. It combines features of the best-first search and Dijkstra's algorithm, prioritizing nodes based on a cost function that includes both the distance from the start node and an estimated distance to the goal. This heuristic approach allows A* to efficiently find the shortest path in various settings, from game development to robotics. However, when traditional A* search is applied to determine routes for multiple connected agents, it often overlooks the intentions of other vehicles, treating them as static obstructions. This limitation emerges because A* focuses on finding a single optimal path from a start point to an endpoint without considering dynamic interactions between multiple agents. Consequently, when a collision occurs (i.e., the computed minimal distance between any two agents falls below the safety margin), it necessitates a rerun of the path calculation. In situations with a high number of units or limited roadway space, the algorithm may enter a repetitive cycle of collisions and subsequent recalculations, highlighting the need for an enhanced or alternative approach for cooperative pathfinding in complex environments.

The Cooperative Path Search algorithm as shown in *Algorithm 1*, provides an elegant solution to multi-agent pathfinding problems [5]. Its primary objective is to identify paths for multiple agents, ensuring they reach their destinations without colliding with one another. The relevance of this algorithm becomes evident in vehicle scheduling, where ensuring non-colliding paths for vehicles is paramount. It addresses this issue by decoupling path computations into a series of path searches and sharing the units' intentions by storing computed paths in a reservation table.

Algorithm 1: Cooperative path search for vehicle scheduling [5]

```
Input: $Vehicles, RoadNetwork$
1  $SpaceTimeGrid \leftarrow$ InitializeSpaceTime($RoadNetwork$)
2  $Paths \leftarrow$ EmptyList()
3  foreach $vehicle$ $in$ $Vehicles$ do
4      $start \leftarrow$ vehicle.startPosition
5      $goal \leftarrow$ vehicle.endPosition
6      $path \leftarrow$ FindPath($start, goal, SpaceTimeGrid$)
7      if $path$ $is$ $not$ $null$ then
8          $Paths$.append($path$)
9          UpdateSpaceTimeGrid($SpaceTimeGrid, path$)
10     else
11         return No valid path found for a vehicle
12 return $Paths$
```

In conclusion, by adapting the Cooperative Path Search algorithm, we can achieve efficient vehicle scheduling that guarantees non-colliding paths for vehicles, ensuring safer and smoother vehicular movement.

2.2 Future traffic predictors

Traffic predictions have always been pivotal for traffic management. With advancements in computational power and data acquisition, traffic predictions have evolved significantly. They now range from predicting broader traffic flows in urban environments to the precise motion of individual vehicles.

2.2.1 Traffic flow prediction

Traffic flow prediction involves forecasting the quantity and movement patterns of vehicles in a given area or along a particular route. Traditional methods, such as time series analysis (e.g., ARIMA), have been employed for years to model and predict traffic flows. However, such models have been restricted to linearity assumption, stationary requirement on data statistical properties, and accurate only on short-term forecasting.

With the surge in available traffic data and the power of neural networks, deep learning has made its mark on traffic flow prediction. Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, are particularly apt for sequence prediction tasks. LSTM consists of a cell state and three gates: Forget Gate, Input Gate, and Output Gate. These gates control the flow of information into, within, and out of the cell. For a given input x_t at time t and the previous LSTM state h_{t-1}:

1. **Forget Gate**:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

   This gate decides which information from the cell state should be thrown away or kept. It outputs a number between 0 (completely forget) and 1 (completely remember).

2. **Input Gate**:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

   The input gate updates the cell state. It first decides which values will be updated using i_t. Then, a tanh layer creates a vector of new candidate values.

3. **Update of Cell State**:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

   The previous state C_{t-1} is updated to the new cell state C_t. It's a combination of the old state (as controlled by the forget gate) and the newly proposed state from the input gate.

4. **Output Gate**:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(C_t)$$

It decides what information from the cell state is sent as output.

Where:

- σ is the sigmoid function.
- W and b are the weights and biases for each gate, respectively.
- $[h_{t-1}, x_t]$ denotes the concatenation of h_{t-1} and x_t.

LSTMs are designed to recognize patterns over time intervals and are intrinsically suited for sequence patterns for the following reasons:

1. **Cell State**: The cell state acts as a ”memory” of the LSTM, storing relevant historical information. The controlled addition or removal of information through gates allows LSTMs to keep long-term dependencies.

2. **Gating Mechanism**: The gates (specifically the forget and input gates) control when information is ”remembered” or ”forgotten.” This means LSTMs can learn to recognize patterns over varying time intervals.

3. **Vanishing Gradient Problem**: Traditional RNNs suffer from the vanishing (or exploding) gradient problem, making them forget long-term dependencies. LSTMs, due to their architecture, can mitigate this problem. The cell state’s design, with

multiplicative gate values, ensures that the gradient can pass through many time steps without vanishing or exploding as quickly as in basic RNNs.

In conclusion, LSTMs are adept at recognizing and 'remembering' temporal patterns in sequences, a capability that proves particularly valuable in traffic prediction tasks involving long-term dependencies. To enhance this ability, the attention mechanism is often integrated with LSTM-based models. This mechanism, initially introduced by Vaswani et al. [6], fundamentally alters the way in which the model processes input sequences. Unlike traditional LSTM models that treat all parts of the input sequence equally, an attention-augmented LSTM can 'focus' on specific segments of the input more intensely at different stages of processing. This is achieved by assigning varying attention weights to different elements of the input sequence.

These attention weights play a crucial role in determining the influence of each input element on the output at any given time. Essentially, the model learns to pay more 'attention' to certain parts of the input that are more relevant for making predictions. The computation of attention weights is typically a multi-step process: it involves scoring each input element based on its importance to the current output context, normalizing these scores to form a probability distribution (using functions like softmax), and then applying these weights to create a weighted combination of the input features. This mechanism allows the LSTM to dynamically adjust its focus on different parts of the input sequence throughout the prediction process, leading to improved handling of long sequences and complex patterns.

In conclusion, LSTMs are designed to both recognize and ”remember” temporal patterns in sequences, making them especially powerful for traffic prediction tasks with long-term dependencies.

In addition, the attention mechanism was introduced to help the LSTM-based model learn better on long sequences. It provides a way for the model to ”focus” on different parts of the input sequence dynamically while producing an output. Given an input sequence, attention allows the model to assign different attention weights to each input element. These weights dictate how much focus the model should give to each input while generating the output. Mathematically, the attention weights are often computed using the following steps:

1. Calculate the alignment scores between the current decoder state (in seq2seq models) or the current state in attention-based LSTMs and each encoder state.

$$e_t = \text{align}(h_t, h_s)$$

where h_t is the current state and h_s are all the states in the sequence.

2. Compute the attention weights using a softmax:

$$\alpha_t = \text{softmax}(e_t)$$

3. The context vector is computed as a weighted sum of the input states:

$$c_t = \sum_s \alpha_{t,s} \cdot h_s$$

This context vector c_t is then used along with the original state h_t to make decisions. In an attention-based LSTM, instead of just using the LSTM's hidden state for decisions, the context vector (from the attention mechanism) is also used. This allows the LSTM to focus on different parts of the input sequence based on the attention weights, effectively giving it a dynamic view of the input data. In essence, the attention mechanism augments the capabilities of LSTMs, enabling them to handle longer sequences and providing a form of interpretability to their predictions.

2.2.2 Vehicle motion prediction

Vehicle motion prediction plays a pivotal role in a centralized scheduling system, helping the system anticipate the behavior of entities and make plans accordingly. Historically, vehicle motion prediction relied on mathematical models based on physics and kinematics. One most common models is the Constant Velocity Model (CVM) with the following formula:

$$x_{t+1} = x_t + v_t \Delta t$$

where x_t is the position at time t and v_t is the velocity and it assumes that the vehicle maintains a constant velocity. Leveraging its computational efficiency, we will employ the CVM to estimate the positions of connected vehicles as they traverse a specified path.

Despite their real-time prediction advantages, these models often falter in dynamic environments, lacking adaptability in complex traffic scenarios. Sequence-to-sequence models heralded a paradigm shift in vehicle motion prediction. Leveraging deep learning architectures, especially attention-based LSTM mentioned above, they can recognize temporal patterns in data sequences. By treating past trajectories as input sequences and predicting future trajectories as output sequences, the encoder encapsulates the context, while the decoder projects the future path. Yet, even with their adaptability, Seq2Seq models can sometimes overlook intricate interactions between vehicles in dense traffic.

Building upon the foundations set by prior methods, Graph Neural Networks (GNNs) offer a more nuanced approach. They envisage a traffic scene as a graph: nodes symbolize vehicles, and edges denote spatial inter-relationships and interactions.

The core idea behind GNNs can be distilled into iterative node updates based on their neighbors, which allows the model to capture intricate traffic scene dynamics. Mathematically, the GNN update rule is often formulated as:

$$h_v^{(l+1)} = \sigma\left(W \cdot \text{AGGREGATE}^{(l)}\left(\{h_u^{(l)}, \forall u \in \text{Neighbors}(v)\}\right)\right)$$

Where:

- $h_v^{(l)}$ is the feature vector of node v at layer l.
- σ represents a non-linear activation function.
- W is a weight matrix.
- AGGREGATE is an aggregation function, like mean, sum, or max, that compiles information from neighboring nodes.

In our study, we utilize a Graph Neural Network (GNN) to serve as an input encoder, which captures the spatial relationships between vehicles. This encoded information is then fed into an attention-based LSTM, allowing us to accurately understand temporal dependencies between sequential data points.

Lastly, we integrate the aforementioned model using a step-by-step prediction methodology. Unlike Seq2Seq models that predict the entire future trajectory in one go, the Step-by-Step prediction method, as the name suggests, forecasts one step at a time. After predicting the immediate next step, the model uses this prediction as part of the input for forecasting the subsequent step, and this process continues iteratively.

Chapter 3

System architecture

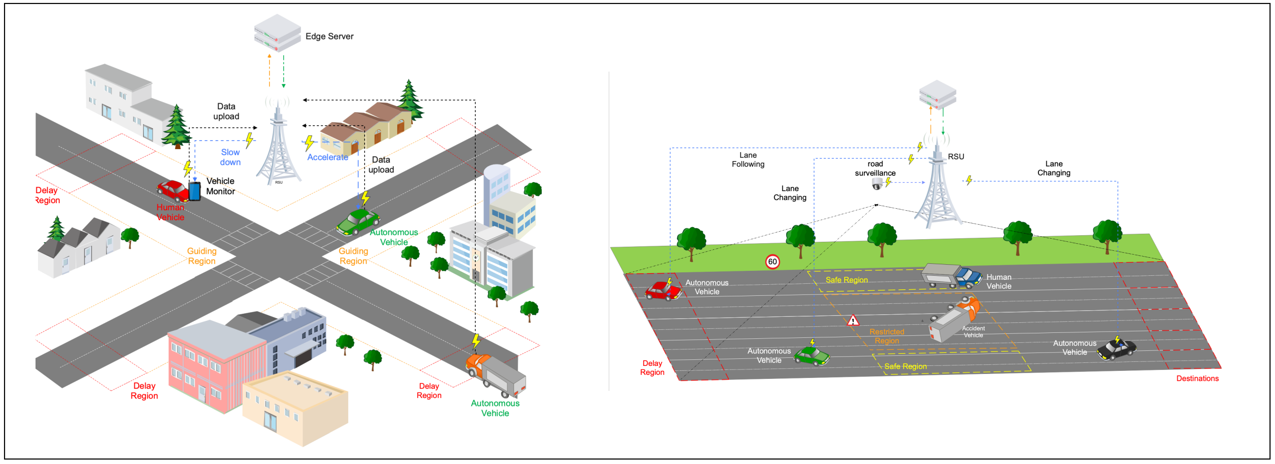

Figure 3.1: Abstract graph of ET paradigm

In this work, illustrated in Figure 3.1, we delve into two specific traffic scenarios. In both scenarios, we assume the availability of Roadside Units (RSUs) and enhanced edge computing capabilities. We operate under the premise that the edge server can efficiently handle real-time data aggregation and decision-making for its designated region, subsequently communicating instructions to vehicles. As corroborated by Ebrahim et al. [7], this setup excels in supporting cooperative driving, ensuring optimal Age of Information (AoI). For our experimental procedures, we break down the scheduling into individual time units, denoted by $t \in T$.

In the non-signalized intersection scenario, we assume all vehicles $v \in V$ are connected, periodically relaying their information (location (v_l), speed (v_s), driving intent (v_i), and surrounding information (v_e)) to the local RSU. Autonomous vehicles natively interface with the RSU through V2I, while we assume human-driven vehicles also have integrated network systems, receiving our system's guidance via onboard displays or smartphone apps [8]. These vehicles can input the driver's intended destination via an onboard interface, allowing our system to provide tailored maneuvering instructions.

In contrast in the highway scenario, we do not assume all vehicles are connected to the system. Instead, our system gleans their statuses via road surveillance, as shown in Figure 3.1. The main challenge is predicting non-connected vehicles' behaviors, such as lane changing. We've integrated a highway merging into an intersection, prompting many vehicles to seek lane change. An introduced obstruction, like an accident, further encourages additional lane changes. Against this setting, we deploy two deep learning models: the first forecasts non-connected vehicles' ultimate lane choices using past traffic data, and the second predicts their immediate movements based on the first model's output.

The edge server processes incoming data, updating the data platform with the current and predicted statuses of all agents using ML models. It then identifies vehicles requiring path planning and ranks them based on a given policy, which encompasses both registered vehicles deviating from or lacking assigned paths and new requests. The server then sequentially plans paths for these vehicles and dispatches decisions to them.

The primary goal of this system is to manage vehicle speeds effectively. This aspect of speed management is crucial for two reasons. Firstly, by controlling speeds, we aim to maximize the throughput of the intersection – that is, to increase the number of vehicles passing through the intersection in a given time frame. Secondly, efficient speed management minimizes the overall time vehicles spend at the intersection, thereby reducing congestion and improving traffic flow. In high-traffic scenarios, particularly where intersections lack signals, the challenge is to allocate the limited space resourcefully. Our system prioritizes vehicles based on various factors, such as their arrival time at the intersection and urgency, to optimize the flow and ensure safe, efficient travel for all road users. This approach not only enhances traffic efficiency but also plays a significant role in reducing the likelihood of conflicts at these intersections.

3.1 Time-based topology graph(TG)

Our foremost challenge is efficiently representing diverse traffic scenarios in a digital twin. Previous studies [8–10] often adopted the grid box method, segmenting each lane into squares. Yet, these methods don't account for lane changes; vehicles are presumed to strictly follow their lanes, including at intersections. The TG method offers several advantages over the grid box approach:

1. **Rich Data Storage**: Unlike the grid box method that primarily captures occupancy status, the TG serves as a comprehensive data platform for the road's Digital Twin. Owing to its structure, it can house a wealth of real-world information. Each TG_t reflects the scene at time step t, with nodes representing real-world waypoints and edges indicating possible movements. Specifically, edges within a lane indicate lane-following, while those between lanes imply lane changes. Moreover, each TG edge can encompass additional information such as geometric data, speed limits, driving directions, and future reservations. This rich data infrastructure aids rapid data access and processing in subsequent stages.

2. **Efficiency in Machine Learning**: When it comes to machine learning, particularly with Graph Neural Network (GNN) based motion prediction, working with TG is more streamlined. Given that GNNs inherently work with graph structures, utilizing data from nodes in TG_t and adjacent edge data as node features for input becomes seamless. Additionally, we can easily formulate adjacency matrices based on distances between connected nodes. There's a notable efficiency in data conversion and pre-processing when using TG compared to the grid box method.

3. **Balancing Accuracy and Computation**: The grid box method struggles with a trade-off between computational efficiency and the precision of the space representation. Denser grids capture detailed events, like lane drifting, but demands more computational power both for occupancy determination and subsequent processing. On the other hand, coarser grids might oversimplify the occupied space, leading to unnecessary blockages. As depicted in Figure 3.2, consider a vehicle slightly deviating at A2. For grid box method, both A2 and B2 get occupied, potentially causing unnecessary traffic impediments. However, the TG method employs edges to signify traffic availability. When matched for equivalent node/grid box count, TG is more precise. In the earlier example, only Node N2 and its proximate edges, like N2-N3 and N1-M4, get occupied, preserving the M2 to M3 route for other vehicles.

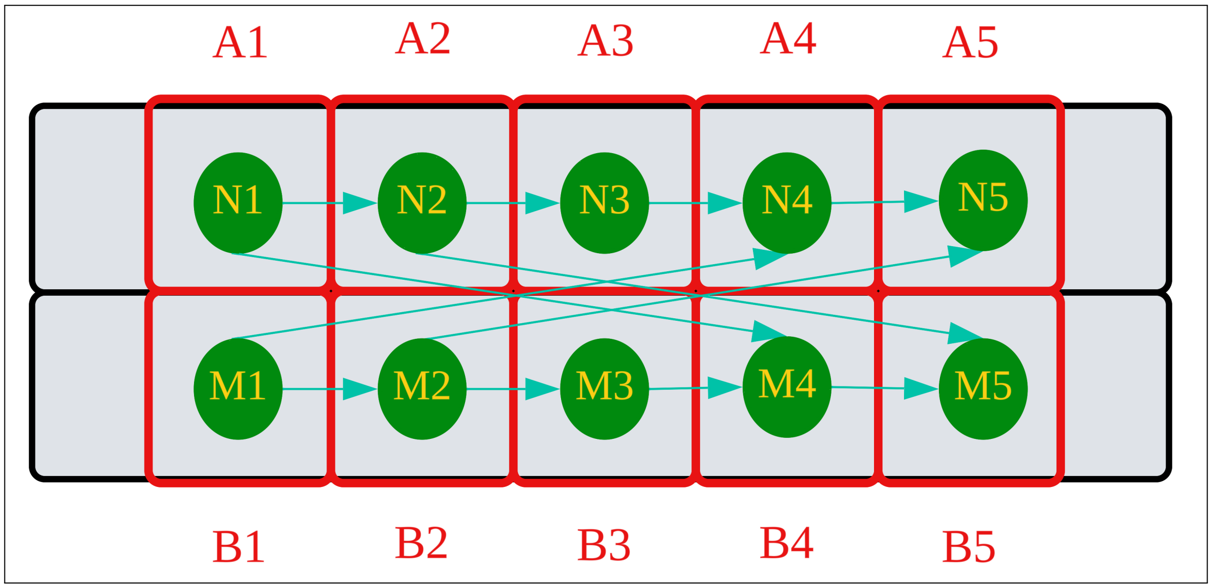

Figure 3.2: At a sample two-lane highway TG, the red squares ranging from A1 to A5 and B1 to B5 represent traditional grid boxes. In contrast, the green nodes and edges from N1 to N5 and M1 to M5 showcase our TG approach.

Next, to accurately pinpoint a vehicle's location on our TG, we adopt 2D Rotation Transformations. These transformations help to derive the exact coordinates of the four corners of the vehicle, providing a more precise representation of its presence on the TG. For example, the front-left corner of the vehicle can be determined using:

$$x_1 = x + \frac{w}{2}\cos(\theta) - \frac{h}{2}\sin(\theta) \tag{3.1}$$

$$y_1 = y + \frac{w}{2}\sin(\theta) + \frac{h}{2}\cos(\theta) \tag{3.2}$$

Here, (x, y) stands for the center coordinates of the vehicle. The parameters w and h denote the vehicle's width and height, respectively, and θ is the vehicle's yaw angle, or its rotation relative to a reference direction.

After determining the four corner coordinates, the next step is to align our reference frame with the vehicle's orientation. This is done by rotating our coordinate system around the vehicle's center by the inverse of its yaw angle, $-\theta$. With this adjusted frame of reference, it becomes straightforward to determine which *nodes* and *edges* of the TG the vehicle's rectangle intersects with.

While this approach is adept at capturing the real-time location of any vehicle on the TG_t, projecting future positions introduces further complexity. For vehicles that strictly follow their lanes, predicting the future center position is relatively simple and relies on initial coordinates, speed, and a predictive model we'll delve into subsequently.

Yet, predicting the yaw angle, θ, for lane-changing vehicles presents a tougher challenge. A vehicle's trajectory during a lane change can vary based on factors such as speed, vehicle mass, steering input, and the state of the road, which makes determining every possible trajectory in our TG a Herculean task. As a pragmatic solution, we've chosen a standardized lane change trajectory that applies to all vehicles and is represented as an edge on the TG. This edge incorporates a hyper-parameter, dictating the overall distance of the lane change, ensuring that despite its standardized nature, there's still a degree of flexibility in its representation.

By simplifying the lane-changing trajectory to an edge between an entry and an exit node on the TG, estimating the yaw angle, θ, becomes a straightforward geometric problem. Given two points, namely the entry and exit nodes on our edge, we can calculate the difference in their x and y coordinates as:

$$\Delta x = \text{exit.x} - \text{entry.x} \tag{3.3}$$

$$\Delta y = \text{exit.y} - \text{entry.y} \tag{3.4}$$

These differences, Δx and Δy, effectively represent the horizontal and vertical changes, respectively, between the two points. The yaw angle, θ, represented by the angle between the vehicle's heading and the x-axis, can be derived by calculating the arctangent of the ratio of these differences. The arctangent function (often denoted as arctan or $\tan^{-1}$) gives the angle whose tangent is the quotient of the two specified numbers. The equation to determine θ is:

$$\theta = \arctan\left(\frac{\Delta y}{\Delta x}\right) \times \frac{180}{\pi} \tag{3.5}$$

With this geometric approach, predicting θ becomes uncomplicated and precise, eliminating the need for complicated dynamic modeling for lane-changing maneuvers.

3.2 System workflow

Figure 3.3 demonstrates the flow for a vehicle request. The edge servers capture requests, holding them in a $waitList$ until a dynamically determined $waitingperiod$ elapses. In this interval, vehicles use autopilots for motion management, while the server updates the $\boldsymbol{TG}$ with incoming vehicle data.

The system continuously tracks connected vehicles via RSU. For non-connected vehicles, strategically placed sensors, like surveillance cameras, detect their status. These cameras use deep learning to extract vehicle data from video streams. When cameras detect connected vehicles, this serves to cross-validate their reported data.

In path planning, movements of non-connected vehicles are predicted and slots are reserved in $TG_{i+1:i+n}$, where n is the maximum future reservation span. Next, connected vehicles are prioritized and assigned paths within this span. If a path is found, vehicles are alerted. Ongoing monitoring ensures path adherence. If a vehicle strays notably from its path, it's placed back in the $waitList$ with an updated starting point. Meanwhile, those without a path are advised to maintain a consistently lower speed and added to the $waitlist$. In the next $waitingperiod$, the system reattempts path allocation, giving these vehicles higher priority.

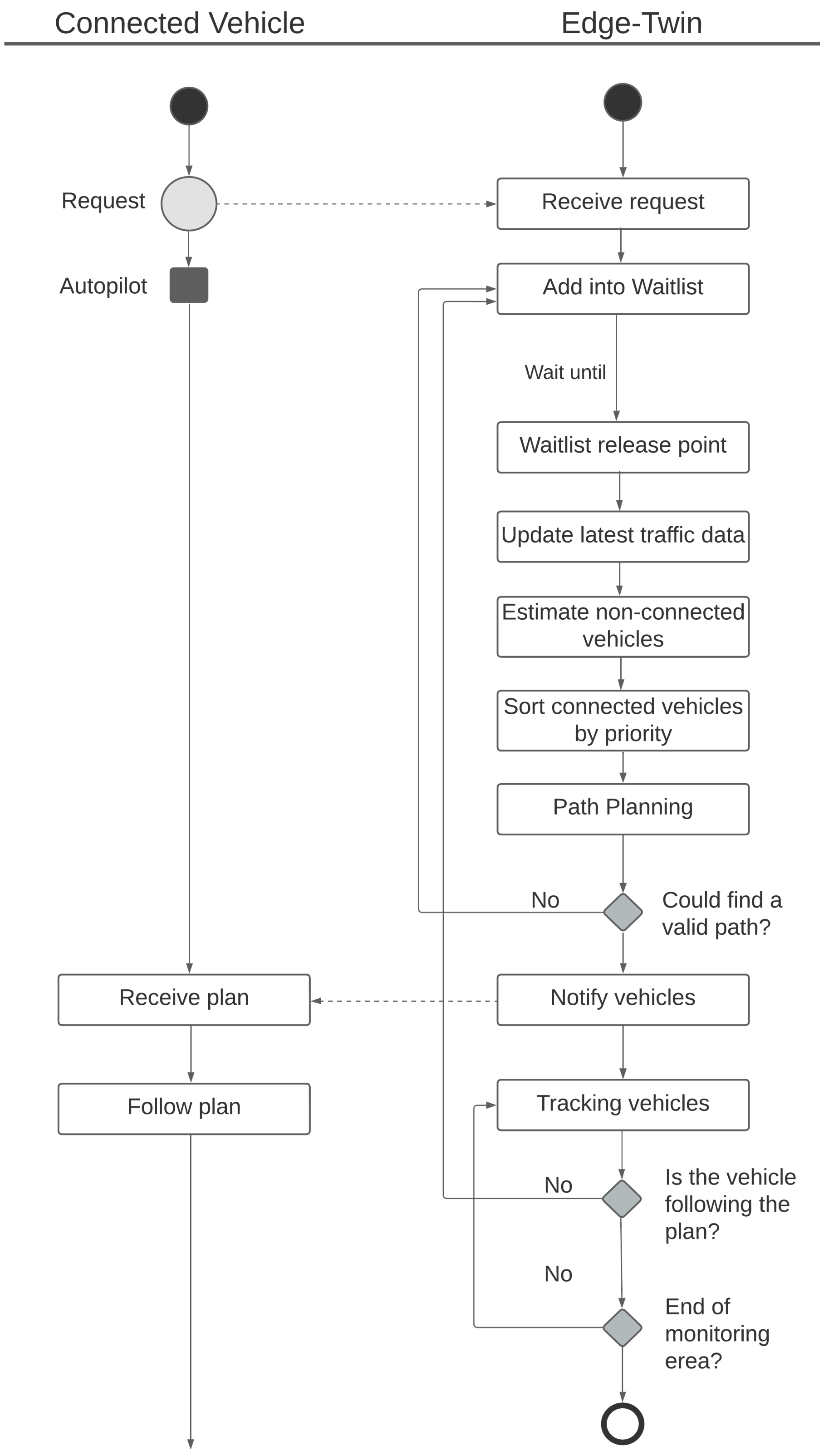

Figure 3.3: System workflow

Chapter 4

Path planning

The primary goal of path planning is to determine a reliable route for each connected vehicle to adhere to, extending from its initial location to its targeted destination. This path is represented by a series of interconnected nodes from the TG. Concurrently, the system should recommend appropriate maneuvers for each vehicle at each node, ensuring that the vehicle can reach the anticipated location within the scheduled timeframe.

4.1 Time-based search algorithm

Our principal algorithm draws from the Cooperative A* algorithm [5], a highly reputed method for tackling cooperative path planning dilemmas, enabling the calculation of collision-free routes for multiple connected agents. In our path planning algorithm, this strategy is employed with the TG serving as the reservation table. Furthermore, we have developed a unique time-based path search algorithm that not only identifies a sequence of nodes as a path but also suggests suitable maneuvers at each node. Individual path computations for each autonomous vehicle are conducted using customized strategies that consider the surrounding traffic environment.

Utilizing the traditional A* search algorithm, we manage four core structures: *openSet*, a priority queue storing nodes prioritized by their proximity to the end goal; *cameFrom*, a map detailing the optimal predecessor of each node, aiding in tracing the shortest path; *gScore*, a map denoting the start node's journey cost to each subsequent node; and *fScore*, estimating the complete journey cost from start to end, passing through each node. The algorithm kicks off by initializing the start node's *gScore* and *fScore*, adding it to the *openSet*. It then iteratively selects the node with the minimal *fScore*, evaluating its neighbors. The algorithm gauges a provisional *gScore* for each neighbor, contemplating reaching it from the current node. Should this *new_gScore* undercut the existing *gScore* or if the node remains unexplored, we recalibrate *cameFrom*, *gScore*, and *fScore* for that neighbor, introducing it to *openList* if absent. This cycle persists till the *openList* is drained or the goal is achieved. An empty *openList* implies all paths are exhausted with no viable route to the goal, thus returning *Failure*. Reaching the goal enables the retracing of the optimal path using the *cameFrom* linkage.

Algorithm 2: Time-based search algorithm

Input: Time-based Topology Graph TG, Ego vehicle i's localized starting node $Start$, i's longitudinal speed v_i, localized destination node $Goal$, road speed limit l_{max} and l_{min}, starting time t

Output: Return a list of connected waypoints from $Start$ to $Goal$, each waypoints with a suggest speed value v or $Failure$ if valid path couldn't be found

```
1  $historySpeed \leftarrow$ set with $v_i$
2  $openSet \leftarrow$ set with $(Start,\ t,\ historySpeed)$
3  $cameFrom \leftarrow \emptyset$
4  $gScore[(Start, t)] \leftarrow 0$
5  $fScore[(Start, t)] \leftarrow$ heuristic_cost$(Start, Goal)$
6  $fScore[(Start, t)] \leftarrow$ heuristic_cost$(Start, Goal)$
7  while $openSet \neq \emptyset$ do
8      $current, currentTime, historySpeed \leftarrow$
         pop the node in $openSet$ with the lowest $fScore$ value
9      $currentSpeed \leftarrow$ get the last value from $historySpeed$
10     if $current = Goal$ then
11         return reconstruct_path_with_speed()
12     foreach $neighbor$ of $current$ do
13         $Edge \leftarrow T.get\_edge(current, neighbor)$
14         $availableSpeed \leftarrow$ available_speeds$(historySpeed, l_{max}, l_{min})$
15         foreach $newSpeed \in availableSpeeds$ do
16             $newTime \leftarrow$ estimate_arrival_time()
17             if $\forall t \in [currentTime, newTime],\quad t \notin T.get\_egde\_occupid\_time(Edge)$ then
18                 if $newTime \notin openSet$ or $new\_gScore < gScore[(neighbor, newTime)]$ then
19                     $cameFrom[(neighbor, newTime)] \leftarrow (current, currentTime, historySpeed)$
20                     $gScore[(neighbor, newTime)] \leftarrow new\_gScore$
21                     $fScore[(neighbor, newTime)] \leftarrow new\_gScore + heuristic\_cost(neighbor, Goal)$
22                     if $(neighbor, newTime) \notin openSet$ then
23                         $openSet.add((neighbor, newTime))$
24 return $Failure$
```

Differing from the traditional A*, our refined version incorporates anticipated arrival times at nodes based on agent speed. As shown in *Algorithm 2*, our elements, like *openSet* and *gScore*, use both the node and expected arrival time as distinct identifiers. We employ an enhanced *gScore* to consider the agent's speed from the start, resulting in a final path composed of $(node, speed)$ pairs.

Two critical functions support our approach. $available_speeds$ identifies viable speeds between linked nodes using past data and speed limits, l_{min} and l_{max}. Next, $estimate_arrival_time$ forecasts the travel time between nodes based on speeds at each end. Assuming constant acceleration changes, we determine the travel time between nodes with $estimate_arrival_time$, as given by:

$$t_i = \frac{2s}{v_i + v_{i+1}} \tag{4.1}$$

Here, d_i represents the distance between nodes for vehicle i, while v_i and u_i signify initial and final speeds. We utilize the Euclidean Distance as our heuristic, incorporating a penalty for lane deviations to promote timely lane changes.

4.2 Priority-based planning strategy

Using our aforementioned search algorithm, each vehicle iteratively reserves its spot on the reservation table TG. While many existing solutions [8–10] for non-signalized intersections process vehicles based on their arrival time, we argue their methods tend to fall short in complex, real-world scenarios. For instance, during peak hours, intersections can witness unbalanced traffic, with one direction bearing a heavier load. To navigate this, we introduce an approach that prioritizes vehicle servicing, optimizing space allocation in high-demand situations, and ensuring smoother traffic flow.

We propose the Estimated Priority-Based (E-PB) method to effectively rank vehicles considering the traffic density of their entry lanes. In essence, vehicles from busier lanes get precedence. Imagine a situation where two vehicles, A and B, approach an intersection from a vertical direction, with B trailing A. Concurrently, vehicle C approaches from a horizontal direction. In an arrival time-based priority system, both A and C would be prioritized, possibly sidelining B. This could cause B to slow down, ensuring C's uninterrupted passage. Yet, in congested lanes like B's, vehicles tend to be closely spaced. So, by decelerating B, we risk impeding a chain of vehicles behind it, which might outnumber those affected by slowing down C. This can hamper the intersection's overall efficiency.

Algorithm 3: Priority algorithm

```
Input: $waitList$
Output: $pathReserve$, $newWaitList$
1  $newWaitList \leftarrow \emptyset$
2  for $vehicle \in waitList$ do
3      $frontVehicles \leftarrow$ get_front_vehicles($vehicle$)
4      $backVehicles \leftarrow$ predict the number of vehicles that will come after $vehicle$
5      $weight \leftarrow frontVehicles + backVehicles$
6  $waitList \leftarrow$ sort $waitList$ by the $weight$
7  for $vehicle \in waitList$ do
8      $path \leftarrow$ apply Algorithm 1 on $vehicle$
9      if $path$ is a valid $path$ then
10         $pathReserve[vehicle] \leftarrow path$
11         $T.update(vehicle, path)$
12     else
13         $newWaitList.add(vehicle)$
14 return $pathReserve, newWaitList$
```

We present the implementation of this method in *Algorithm 3*. This approach accepts a '*waitlist*' as input, which stores all the vehicles that have applied since the last waiting period as well as those that have yet to be assigned a valid path. To prioritize a target vehicle, we take into account the traffic conditions both in front and behind it. For assessing the front traffic of the target vehicle, we employ the function *get_front_vehicles*. This function quantifies the vehicles within a 15-meter radius in front of the target vehicle that has not yet approached the intersection zone. For the rear traffic, we forecast the number of vehicles that will enter the same lane as our target vehicle in the near future. Subsequently, we adjust the order of the *waitlist* by summing up the traffic data from the front and rear. Vehicles with higher combined traffic are prioritized. Subsequently, *Algorithm 2* is applied to the vehicles in this new order, and their paths are preserved if a valid one is found. If a valid path cannot be found, the vehicles are returned to the *waitlist* and instructed to maintain a consistently slower speed.

Chapter 5

Future traffic prediction

A standout feature of our Edge Twin is its capacity to forecast traffic conditions, enhancing subsequent path-planning algorithms. While prediction priorities differ across traffic settings, they generally fall into two categories: single-vehicle prediction and overall traffic flow estimation. Here, we delve into the prediction models tailored for non-signalized intersections and highways.

5.1 Traffic flow predictors for non-signalized intersection

Building on our previous chapter, our planning strategy requires predicting future traffic following a target vehicle. Specifically, we aim to anticipate how many vehicles will soon join the same lane. For this, we've developed a deep learning model that predicts the vehicle count set to traverse a lane's delay zone at the next time point $i + 1$, based on history data from times $i - 9$ to i. Notably, the time gap between i and $i + 1$ is adjustable; in our tests, it's fixed at 30 seconds.

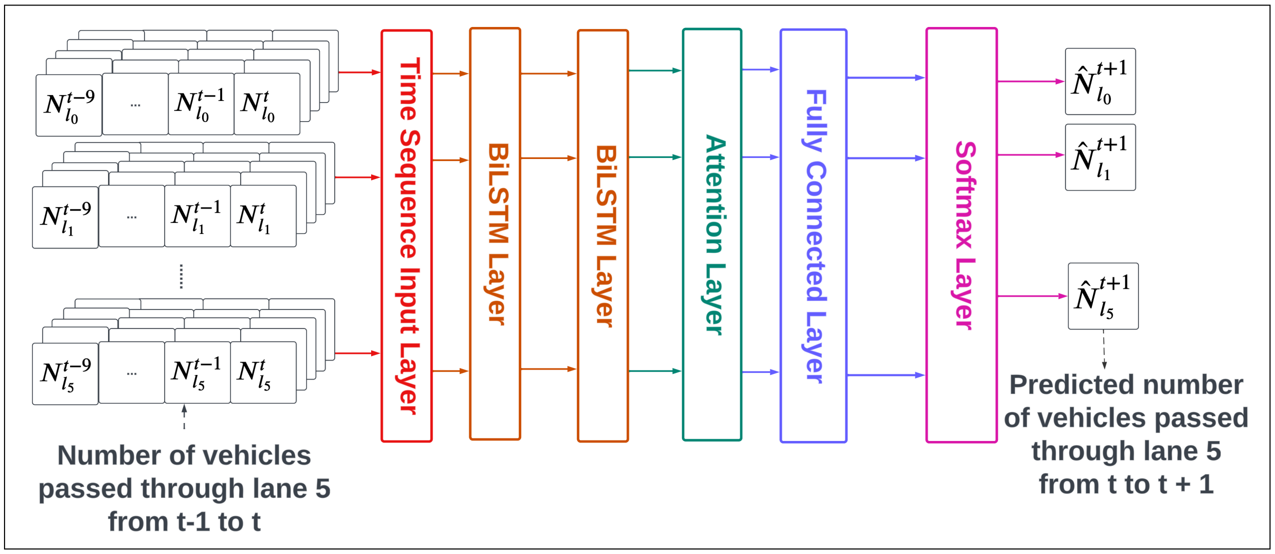
$N_{l_0}^{t-9}$ ... $N_{l_0}^{t-1}$ $N_{l_0}^{t}$
$N_{l_1}^{t-9}$ ... $N_{l_1}^{t-1}$ $N_{l_1}^{t}$
$N_{l_5}^{t-9}$ ... $N_{l_5}^{t-1}$ $N_{l_5}^{t}$
Number of vehicles passed through lane 5 from t-1 to t
Time Sequence Input Layer
BiLSTM Layer
BiLSTM Layer
Attention Layer
Fully Connected Layer
Softmax Layer
$\hat{N}_{l_0}^{t+1}$
$\hat{N}_{l_1}^{t+1}$
$\hat{N}_{l_5}^{t+1}$
Predicted number of vehicles passed through lane 5 from t to t + 1

Figure 5.1: Traffic flow prediction model for non-signalized intersection

Figure 5.1 illustrates the architecture of our model. In addressing the challenge of managing long-term dependencies in time series data, we opted for an LSTM-based approach, a method well-established for its effectiveness in time series traffic forecasting [11–13]. This choice was influenced by the superior ability of LSTMs in capturing temporal dependencies and handling variable-length input sequences, compared to traditional methods such as ARIMA. While ARIMA is a powerful tool for linear time series forecasting, it often falls short in traffic prediction scenarios that exhibit non-linear patterns and require the processing of large datasets with complex temporal dynamics.

To further refine our time series predictions, we incorporated attention-based biLSTMs [14]. This enhancement allows our model to not only capture the sequential data efficiently but also to focus on the most relevant parts of the input sequence, a feature not available in ARIMA or similar linear models.The input layer takes a series represented as $(N_{l_i}^{t-9}, ..., N_{l_i}^{t-1}, N_{l_i}^{t})$, where each component indicates the vehicle count in lane l_i's delay zone from time $t-1$ to t. With two bidirectional LSTM layers, the model captures rich historical and anticipated context. An attention layer then assigns weights to these LSTM outputs, emphasizing crucial parts of the sequence. The weighted results are processed by subsequent fully connected layers to yield the prediction $(\hat{N}_{l_i}^{t+1})$.

We sourced our dataset from a CARLA simulator's six-lane intersection spanning an hour. Traffic flow variations arise from different traffic types created by an exponential random generator with scale values between 50 to 300, each lasting 2 to 5 minutes. The time gap between t and $t+1$ is 100 simulator time units. The dataset is split 70:30 for training and validation. Refer to Table 5.3 for model specifics and outcomes.

5.2 Intention and motion prediction for highway

Highways differ from intersections due to the presence of non-connected vehicles that our system can't directly communicate with. Therefore, we must predict their driving intentions, including final lane choice and real-time positioning. Our approach is twofold: first, we identify which lane the vehicle will occupy by the end of the highway segment; next, we estimate its position at every time step.

5.2.1 Destination predictor

In the initial phase, non-connected vehicles are classified into one of five destination points. Considering our earlier assumption of the experimental highway's tendency for rightward lane shifts, we need a model to discern this pattern from historical highway traffic data.

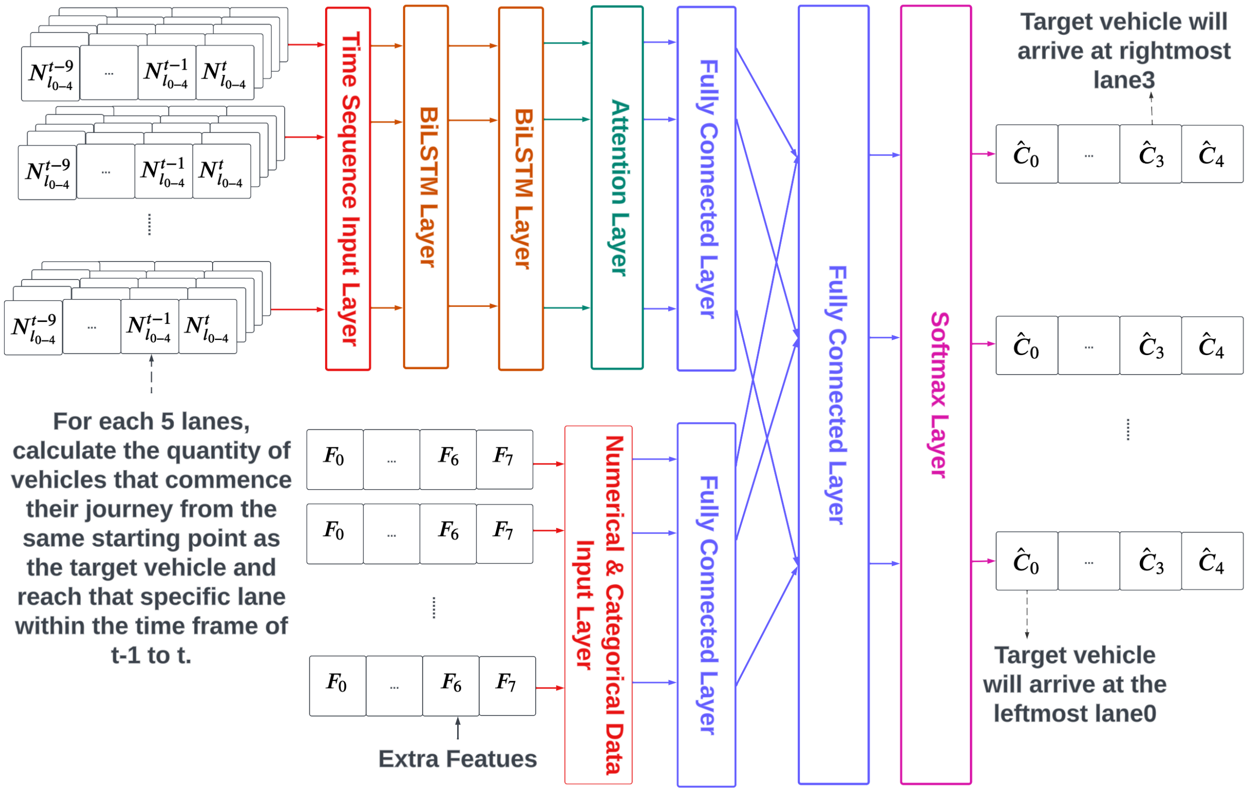
$N_{l_{0-4}}^{t-9}$ ... $N_{l_{0-4}}^{t-1}$ $N_{l_{0-4}}^{t}$
For each 5 lanes, calculate the quantity of vehicles that commence their journey from the same starting point as the target vehicle and reach that specific lane within the time frame of t-1 to t.
Time Sequence Input Layer
BiLSTM Layer
BiLSTM Layer
Attention Layer
Fully Connected Layer
F_0 ... F_6 F_7
Extra Featues
Numerical & Categorical Data Input Layer
Fully Connected Layer
Fully Connected Layer
Softmax Layer
$\hat{C}_0$ ... $\hat{C}_3$ $\hat{C}_4$
Target vehicle will arrive at rightmost lane3
Target vehicle will arrive at the leftmost lane0

Figure 5.2: Destination prediction model for highway

The model architecture, shown in Figure 5.2, adopts a dual-input system. The first input is a time-series data stream, $(N_{l_{0-5}}^{t-9}, ..., N_{l_{0-5}}^{t})$, capturing vehicle counts from similar start points to the target vehicle reaching all lanes between l_0 to l_4 within the interval $t-1$ to t. This input uses a bidirectional LSTM structure, attention mechanisms, and a connected layer, mirroring our prior model but excluding the final output layer. The second input includes the target vehicle's observed state, represented as $(F_0, ..., F_7)$ and detailed in Table 5.1.

Feature Description	**Unit**	**Feature Type**
Instantaneous longitudinal velocity	m/s	numerical
Instantaneous longitudinal acceleration	m/s^2	numerical
Instantaneous lateral acceleration	m/s^2	numerical
Yaw angle	rad	numerical
Entry lane index	None	categorical
Vehicle at the front	0/1	categorical
Vehicle at the front left lane	0/1	categorical
Vehicle at the front right lane	0/1	categorical

Table 5.1: Categorical and numerical features for destination prediction

The resultant outputs from the primary and secondary inputs are subsequently integrated and channeled into a fully connected layer, and a softmax layer determines the vehicle's category, ranging from $(\hat{C}_0, ..., \hat{C}_4)$.

The training data, sourced from a CARLA-simulated hour-long highway scenario with five parallel lanes, establishes a 30-second interval between t and $t+1$. We split the dataset into training and validation sets at a 70:30 ratio. Refer to Table 5.3 for model specifics and outcomes.

5.2.2 Location predictor for non-connected vehicles

In the next phase, we aim to determine the non-connected vehicle's position and, when needed, identify its lane change point. While trajectory forecasting usually favors sequence-to-sequence (seq2seq) models [15–18], we sidestep them due to their constraints. Our system requires predicting a vehicle's entire trajectory, often surpassing a seq2seq model's maximum time frame. Misjudgments in distant predictions could misalign with our Traffic Grid (TG).

Instead, we adopt a step-by-step prediction utilizing a Multi-Task Learning Network. Tailored for the highway's three possible forward moves (lane changes left or right and lane following), our model predicts the vehicle's next node and arrival time. The classification sub-network results in three outcomes, while a regression sub-network estimates the arrival time.

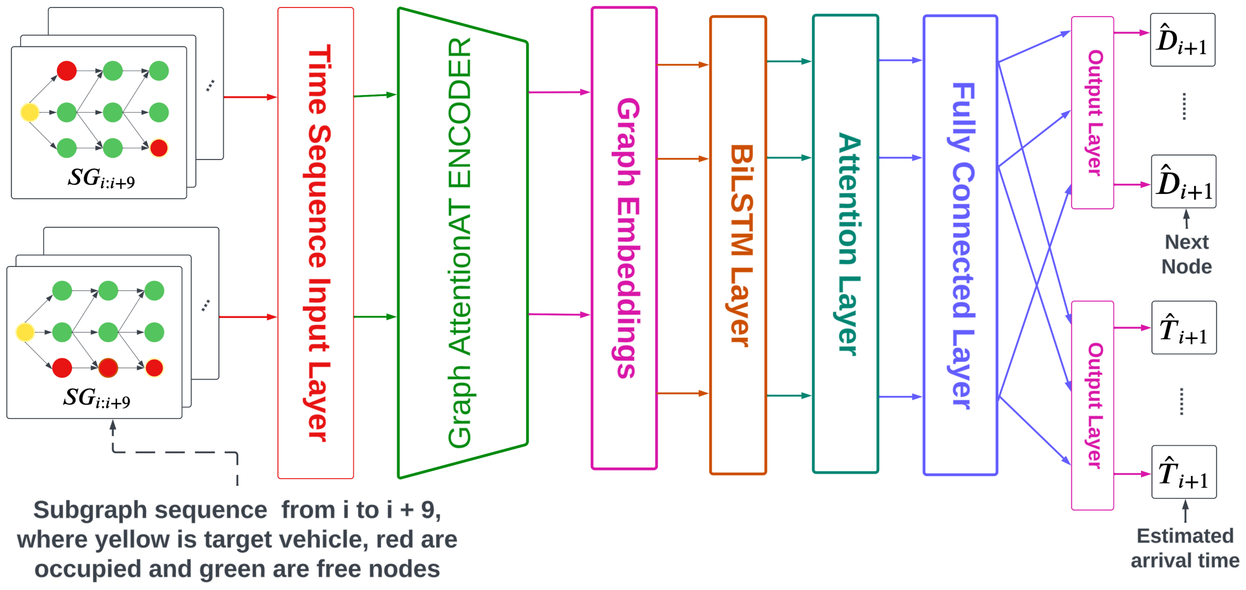

Figure 5.3: Location predictor for highway

Figure 5.3 showcases our model, grounded on the interplay between a vehicle's movements and the conditions of the vehicles ahead. Within this context, our topology map $TG_{i:i+9}$ incorporates reservations from the previous timestep.

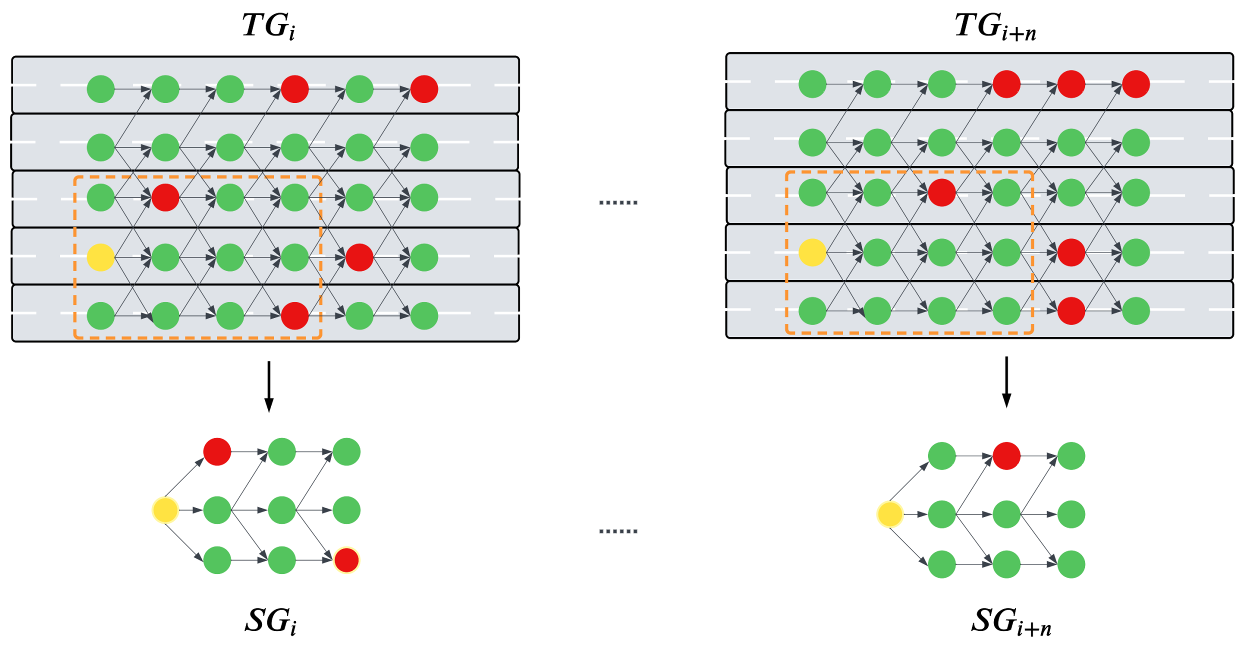

Figure 5.4: A representative sub-graphs $SG_{i:i+9}$ as the graph attention network input

Using $TG_{i:i+9}$'s graph structure, we extract a local subgraph from it as sequence-type input, each focusing on the target vehicle's front nodes and edges up to a depth of 10, as illustrated in Figure 5.4. To capture the spatial and temporal nuances in $TG_{i:i+9}$, we employ a Graph Attention Network (GAT) paired with LSTM, an approach validated by previous traffic studies [11, 13].

To feed subgraphs $TG_{i:i+9}$ into the model, we extract data from each node and nearby edge data as node features for the GAT, then construct adjacency matrices from the TG

structure. These matrices record distances between connected nodes. Relevant node features are detailed in 5.2. Subsequently, a graph attention network is deployed as an encoder to convert each subgraph into an embedding vector. LSTM further processes these vectors and is trained as a multi-task learning model to concurrently predict the sequence of the next three arriving nodes $\hat{D}_{i+1}$and their anticipated arrival times $\hat{T}_{i+1}$.

Feature Description	**Optimizer**	**Feature Type**
Occupancy state	None	categorical
Instantaneous longitudinal velocity	m/s	numerical
Location predictor	None	categorical

Table 5.2: Categorical and numerical features for location estimation

The model is trained with vehicle travel data from CARLA simulation, where each vehicle's records are transformed into adjacency and feature matrices from $TG_{i:i+9}$. With the data spread across an hour, we split it 70:30 for training and validation. Using this model, we predict the next node a non-connected vehicle will inhabit, reserving space in TG_{i+1}. By iterating with subsequent TG datasets, we trace the vehicle's trajectory to its endpoint. Refer to Table 5.3 for model specifics and outcomes.

Model Description	**Loss Function**	**Optimizer**	**Evaluation Function**	**Validation Results**
Traffic Flow predictor	Mean square error	RMSprop	Mean square error	0.5574
Destination predictor	Categorical Cross-entropy	Adam	Accuracy	74.5%
Location predictor	Categorical Cross-entropy & Mean Square Error	Adam & RMSprop	Accuracy & Mean square error	89.2% & 0.127

Table 5.3: Settings for data pre-processing & training & Results

Chapter 6

Simulation

6.1 The CARLA environment

CARLA (Car Learning to Act) [3] is an open-source simulator built specifically for autonomous driving research. It is a highly flexible and realistic simulator that allows researchers to test their self-driving algorithms in a wide range of scenarios. Meanwhile, CARLA is designed to simulate real-world driving conditions with high fidelity. It provides a rich set of sensors including cameras, LiDARs, radars, and GPS, which are essential tools for both autonomous driving and smart traffic. Lastly, CARLA allows researchers to add various dynamic objects into the simulation, such as other vehicles, pedestrians, and traffic lights. Overall, CARLA provides us with an ideal and customizable environment that can be used to test and develop our Edge Twin systems with autonomous driving.

6.2 PID based vehicle

CARLA operates with an in-built autopilot system that permits vehicles to adhere to predefined paths. This autopilot system relies on the principles of a PID controller, an acronym that encapsulates Proportional, Integral, and Derivative. The PID controller is a prevalent feedback controller used across numerous control systems. Within CARLA's environment, a PID controller regulates various attributes of a vehicle, such as velocity or steering angle. The desired setpoint could be a particular speed or a designated waypoint on the roadway, with the controller manipulating the vehicle's throttle, brake, and steering to minimize the difference between the current state and the desired state.

Simultaneously, CARLA also furnishes a Traffic Manager (TM), a more sophisticated interface for researchers to supervise vehicles in autopilot mode. This Manager also incorporates an in-built collision detection mechanism, so vehicles operating in autopilot mode actively circumvent vehicle collisions even when the suggested path entails conflict. Furthermore, the Traffic Manager imprints some human-like behaviour onto the autopilot system - for instance, vehicles invariably decelerate slightly before navigating an intersection. These behaviours allow us to create complex, realistic scenarios to test the robustness of our system.

6.3 Non-signalized intersection experiment

In this research, we employ 'map03', a large town with features of a downtown urban area. The map exhibits intriguing elements of a road network such as a rotary intersection, underpasses, and flyovers. The town also incorporates a raised metro track and a considerable construction site. We chose one of its cross-intersections for our non-signalized experimental environment. This intersection sees traffic from both horizontal and vertical directions. It accommodates six entry lanes, four vertical and two horizontal. Similarly, there are six exit lanes, with a vehicle regarded as leaving the monitor region once it enters any of these lanes. We have excluded any built-in traffic lights within this zone. Figure 6.1 illustrates the intersection we chose.

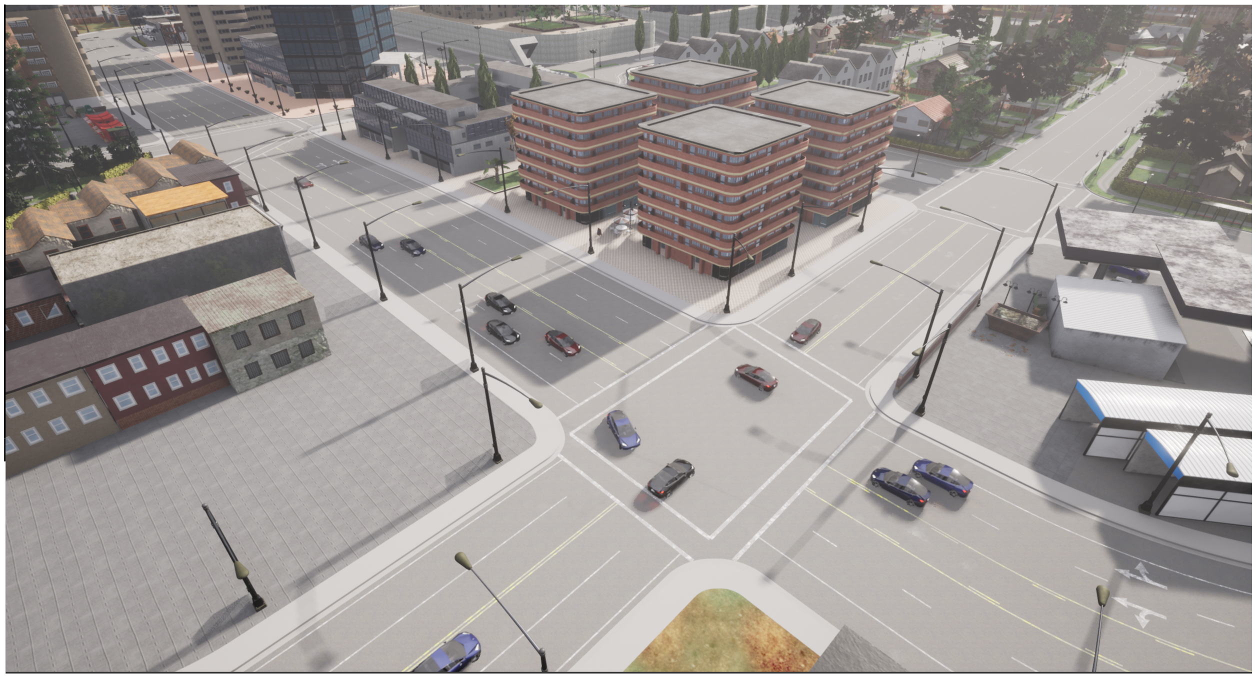

Figure 6.1: Non-signalized intersection at Map03

6.4 Highway experiment with blockage

We opt for 'map06', a sparse town nestled in a pine-laden environment, featuring an array of wide 4-6 lane roads and unique junctions like the Michigan Left. This map is used to simulate traffic congestion on highways. For this purpose, we select a segment of a five-lane road with an intersection at its terminus for our highway simulation. We then designate five endpoints, one at the end of each lane, considering a vehicle to have exited the monitored region once it surpasses these points. Assigning destinations for vehicles within the highway segment enables us to observe more instances of lane changing, thereby increasing the overall complexity of the system. To introduce a temporary obstruction, we place a stationary, overturned truck in the middle of the road segment, effectively blocking the central three lanes and leaving only the outermost lanes open. Figure 6.2 illustrates the highway we chose.

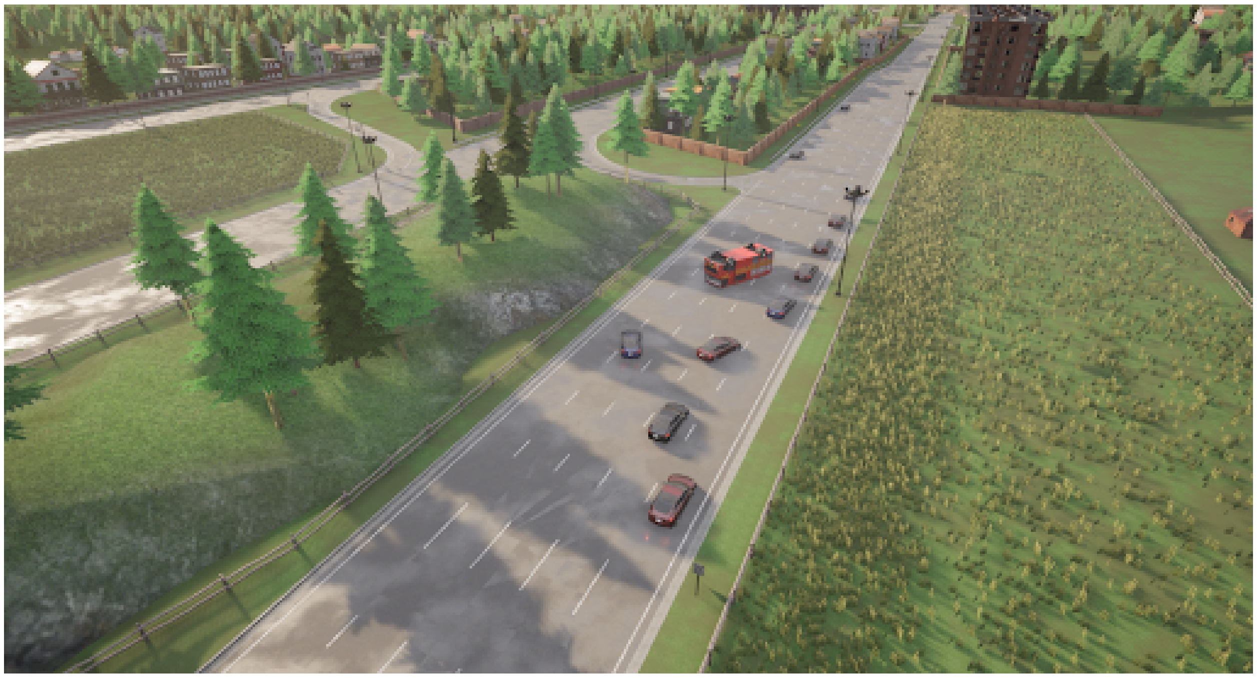

Figure 6.2: Non-signalized intersection at Map03

6.5 Experiment set-up in CARLA simulator

Within CARLA simulator, we model traffic data collection, flow prediction, motion prediction, and path planning for each connected vehicle. The experimental settings in CARLA are detailed in 6.1.

All experiments spanned 36,000 CARLA time units, roughly equating to 30 real-world minutes, and were replicated on 30 occasions. We adopted an exponential random generator for a randomized vehicle spawning approach. This generator excels at crafting numbers fitting the exponential distribution, described mathematically as:

Setting Description	**Map03**	**Map06**
Location	Non-signalized Intersection	Highway
Number of of entry lanes	6	5
Entry lanes direction	2 horizontal & 4 vertical	5 vertical
Speed limit	30 - 50 km/h	50 - 70 km/h

Table 6.1: Settings for CARLA simulator

$$f(x|\lambda) = \lambda e^{-\lambda x}, \quad \text{for } x \geq 0, \quad \text{and 0 otherwise,} \tag{6.1}$$

where λ acts as the rate parameter while β performs the role of the scale parameter (essentially the inverse of the rate). Here, β determines the decay rate of the probability of an event as the inter-event time increases. Specifically, a larger β implies that longer inter-event times are more probable, resulting in a slower decay of the distribution, and vice versa. This choice is motivated by its common use in scenarios where the time between occurrences of successive events follows an exponential distribution. This strategy is a reasonable approximation of real-world traffic patterns, where vehicle arrivals are often ”bursty”. Lastly, we evaluate on different scale values to generate varied forms of traffic flows.

The CARLA simulator provides real-time data on all active vehicles, including locations, speed, deviation, etc. As mentioned in the previous chapter, we employ 2D Rotation Transformations to determine the vehicle's four corner coordinates. Then, we rotate the coordinate system around the rectangle's center by the negative of the vehicle's yaw. Subsequently, we identify all *nodes* and *edges* from the *TG* that overlap with this rotated rectangle. Figure 6.3 illustrates a vehicle traveling at the non-signalized intersection and its reserved occupancy space.

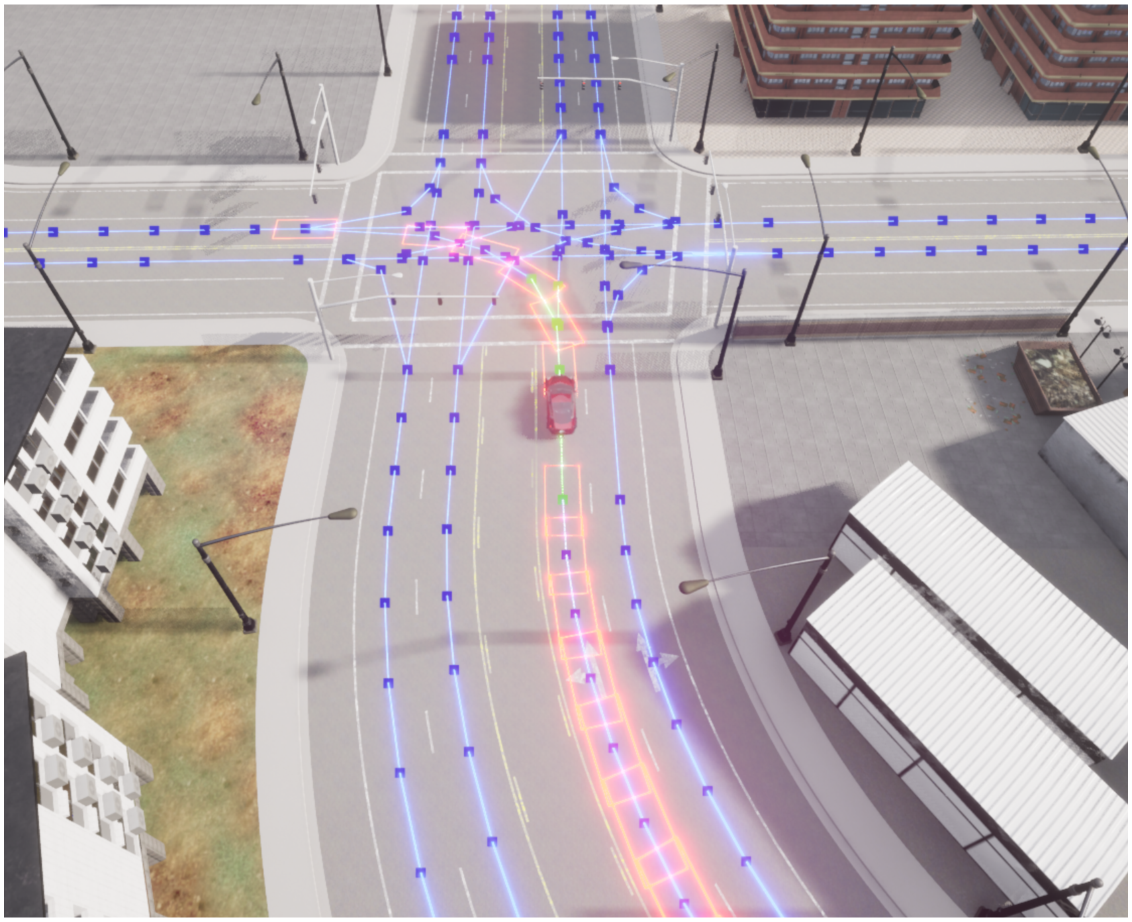

Figure 6.3: This image depicts a connected vehicle navigating through a non-signalized intersection. The red geometrical figures demarcate the vehicle's space estimated by our system, whereas the green lines delineate the area specifically reserved for it.

Chapter 7

Results

In this section, we'll discuss simulating the Edge-Twin on two CARLA simulator maps. We assess its performance using Average Travel Time and Average Traffic Ratio metrics from 30 repeated experiments.

1. **Average Travel Time**: This metric measures the average time taken for a vehicle, whether connected or not, to traverse from entry to exit of the monitoring zone.

2. **Traffic Ratio**: This calculates the percentage of vehicles that passed through the monitoring area over a specified duration. It's normalized against the total vehicles spawned to account for the generator's uneven vehicle creation in different runs.

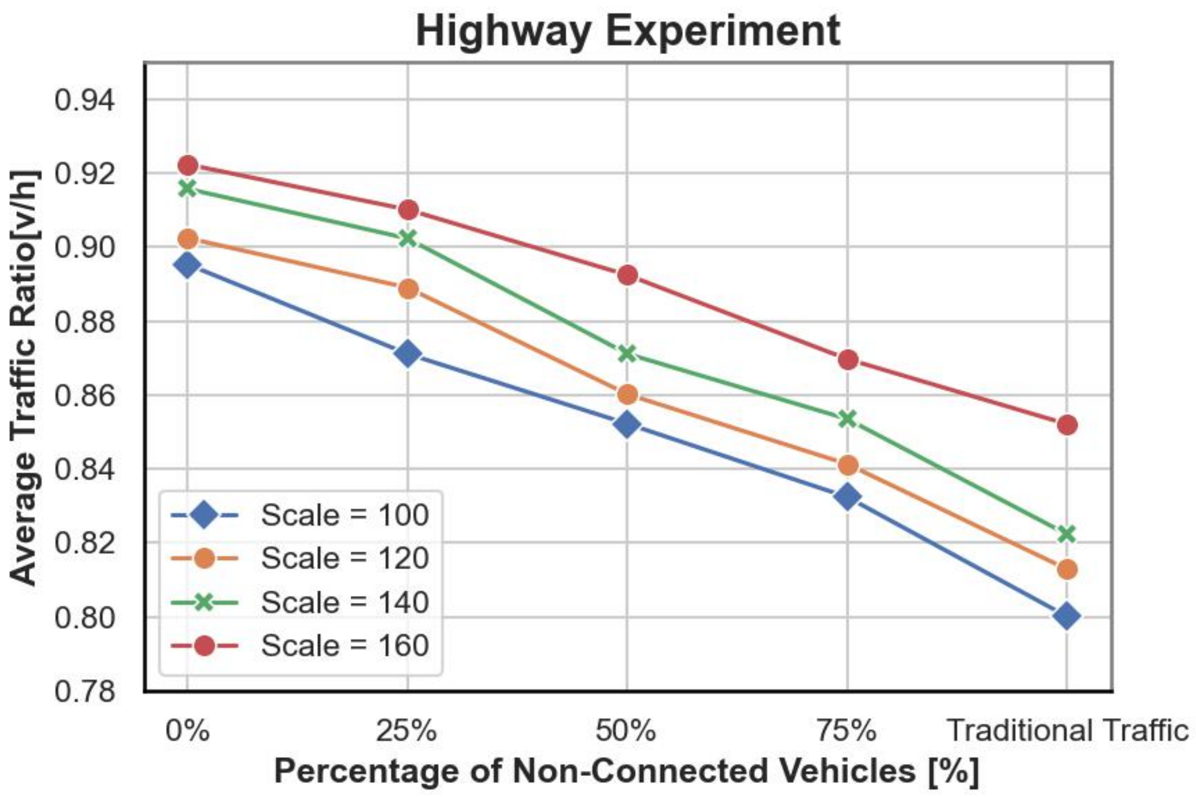

Figure 7.1: Arrive ratio for different magnitudes of traffic flows at highway

7.0.1 Results on high-way

Figure 7.1 and Figure 7.2 showcase the average traffic ratio and travel time on a highway with free flow, under various traffic demands. These demands are adjusted using the scale value, with choices of 100, 120, 140, and 160; here, 100 indicates the maximum traffic demand. The Y-axis distinguishes results based on the share of non-connected vehicles, with the far right denoting an all-non-connected vehicle scenario.

As observed from Figure 7.1, the traffic ratio drops with an increase in non-connected

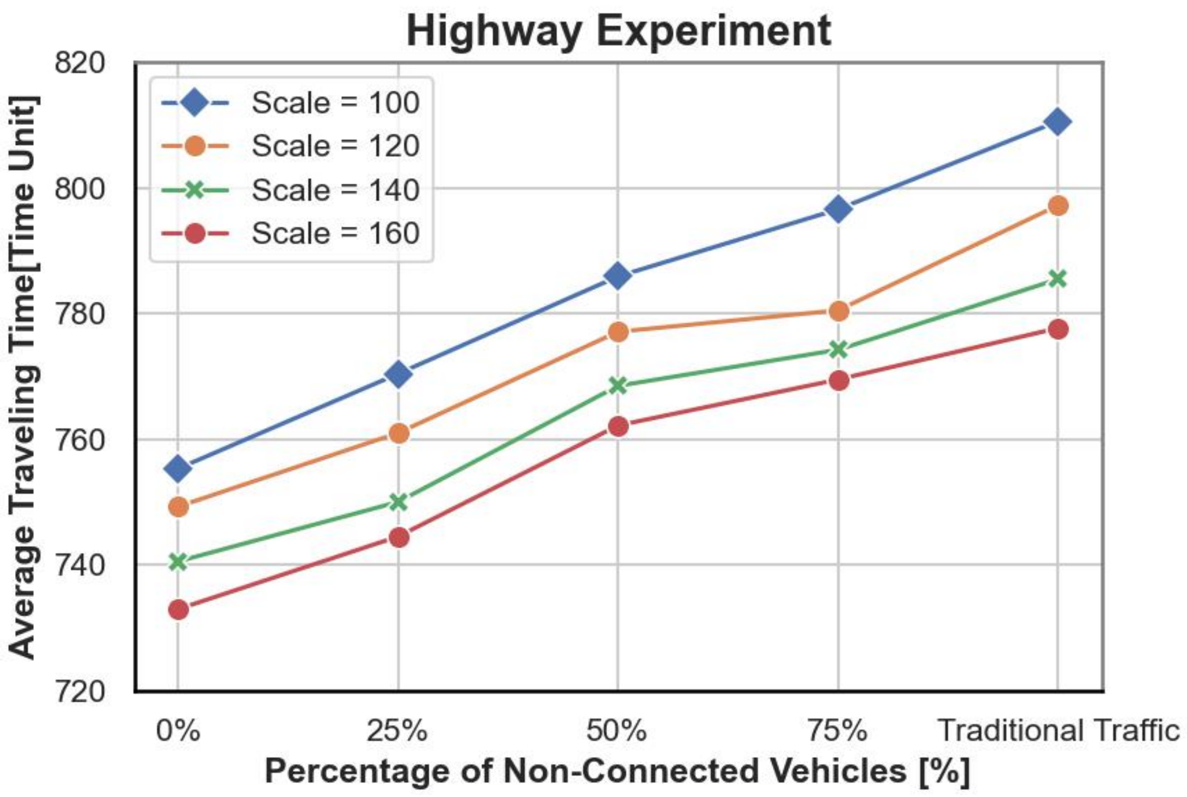

Figure 7.2: Average traveling time for different magnitudes of traffic flows at highway

vehicles, irrespective of traffic demand. A greater number of non-connected vehicles tends to amplify prediction errors about future traffic states, leading to unexpected traffic clashes and slowing down the flow. Overall, this approach outperforms traditional scenarios with only non-connected vehicles by about 7% to 10% across all demands. Similarly, Figure 7.2 shows that travel times increase with a rise in non-connected vehicles.

Figures 7.3 and 7.4 reflect the metrics at the same highway spot but with a traffic hindrance from an accident. Although the trends align with the clear highway scenario, both the traffic ratio and travel times fare worse, given the reduced lane availability due

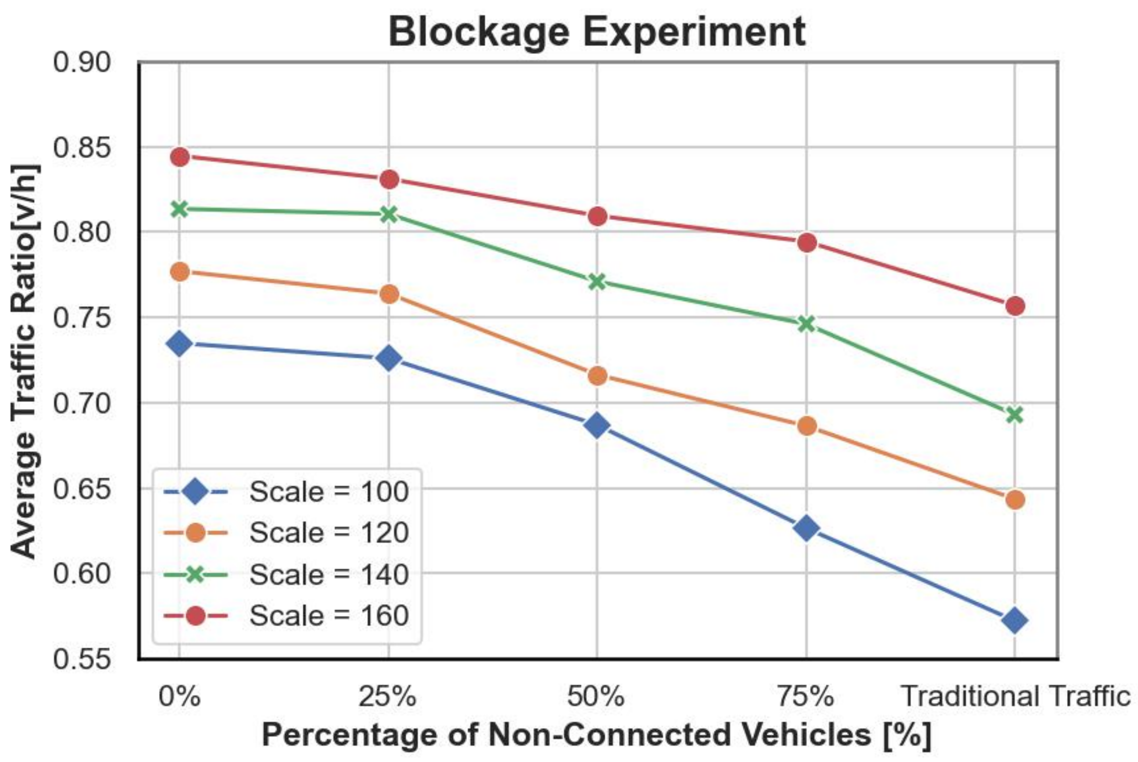

Figure 7.3: Arrive ratio for different magnitudes of traffic flows at blocked highway

to the accident. Moreover, the traffic ratio differences between varying percentages of non-connected vehicles are more pronounced (10% - 15%) than in the free-flowing scenario (7% - 10%). This hints at our system's heightened efficacy as traffic density nears road capacity.

7.0.2 Results on non-signalized intersection

In our examination of non-signalized intersections, we assess our system under various traffic flow conditions, including balanced and unbalanced inflows between horizontal and vertical paths. For consistent inflows, a uniform scale value is used. Unbalanced flows start with

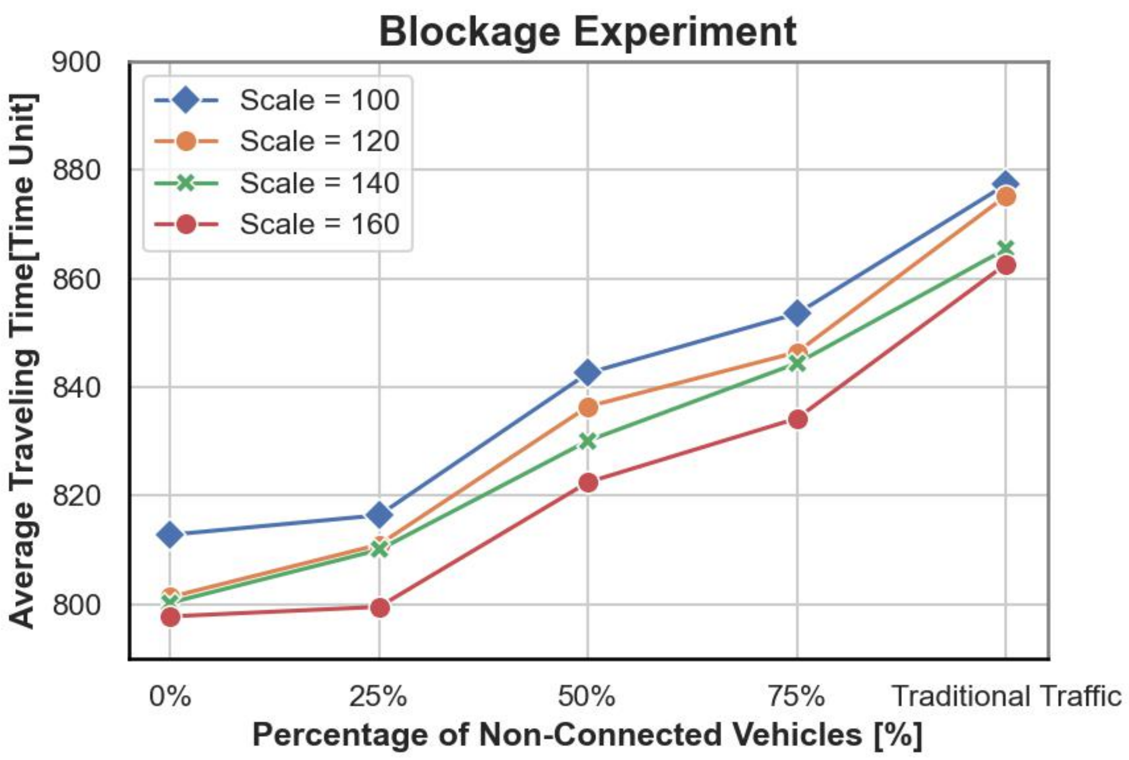

Figure 7.4: Average travelling time for different magnitudes of traffic flows at blocked highway

equal scale values for both directions, then diverge by +/- 40. Every 2000 time units, the horizontal scale value shifts by 20 if within the acceptable range, followed by a similar change in the vertical scale. If no change occurs, the process repeats after 2000 units; otherwise, a 6000-unit pause ensues.

Our analysis of the non-signalized intersection hinges on three primary elements:

1. Three distinct sorting strategies: First Come First Serve (FCFS) [9], Time-Based Priority (T-PB) [8], Collision Potential-Based Priority (C-PB) [10],, and our method,

Estimation-Based Priority (E-PB).

2. E-PB's performance across four traffic flow intensities, defined by scale values of 100, 120, 140, and 160.

3. The efficacy of E-PB when factoring in varying ratios of human drivers: 0%, 25%, 50%, and 75%.

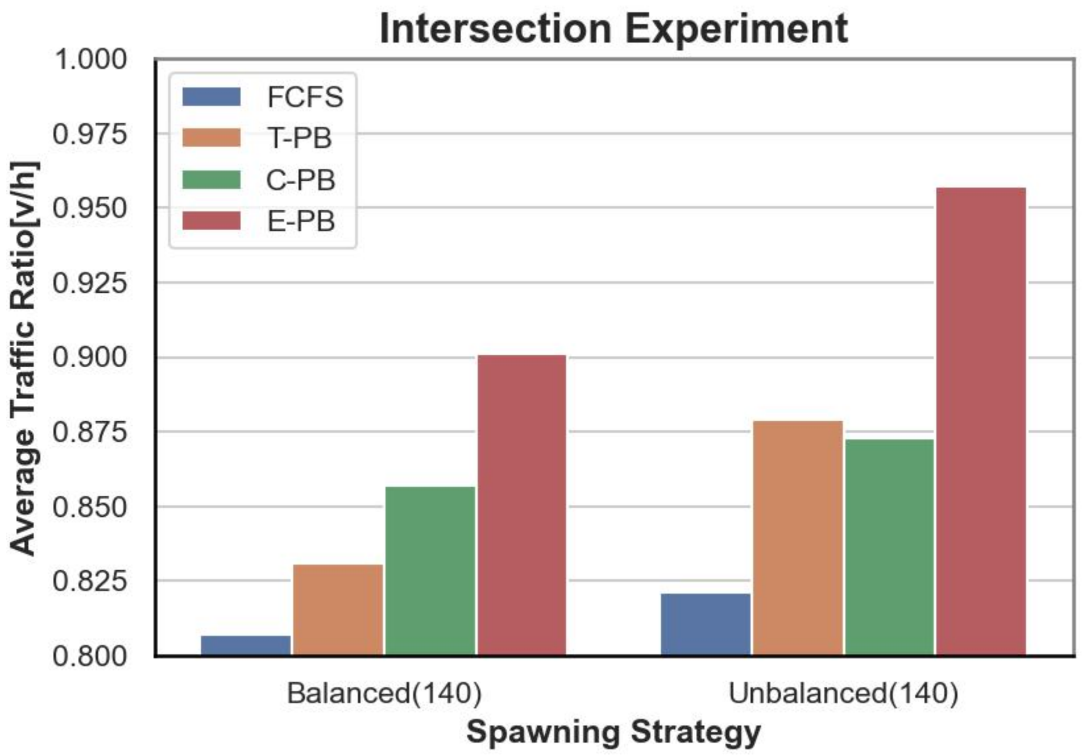

Figure 7.5: Traffic ratio for different strategies at non-signalized intersection

Figure 7.5 and Figure 7.6 display the average traffic ratio and travel time at a non-signalized intersection for different priority strategies. All methods were assessed under the

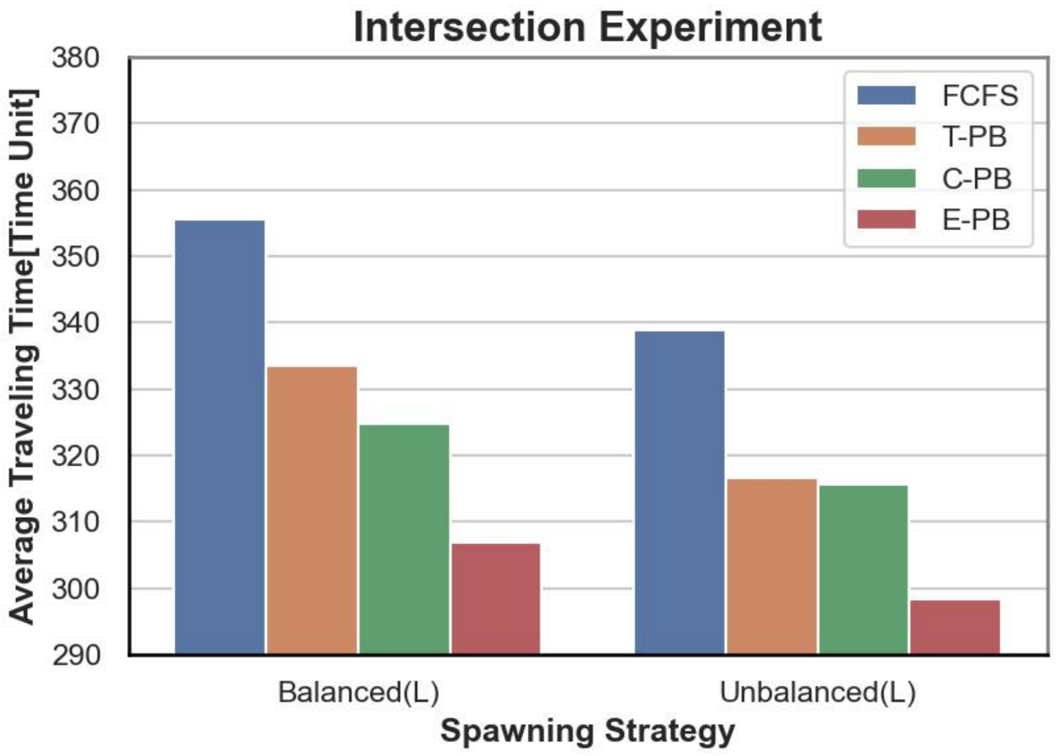

Figure 7.6: Average travelling time for different strategies at non-signalized intersection

same traffic volume with a scale value of 140. The Y-axis differentiates between balanced and unbalanced traffic conditions. Clearly, E-PB consistently leads, with its advantage being more evident in unbalanced traffic. Using FCFS as a benchmark, it consistently trails, with C-PB surpassing T-PB in balanced conditions but losing this advantage in unbalanced situations. This is consistent with our anticipation since earlier methods were gauged in balanced conditions, while our method is designed for unbalanced situations. The trends in travel time (Figure 7.6) reflect these findings.

We assessed the traffic ratio and travel time using the E-PB method across different

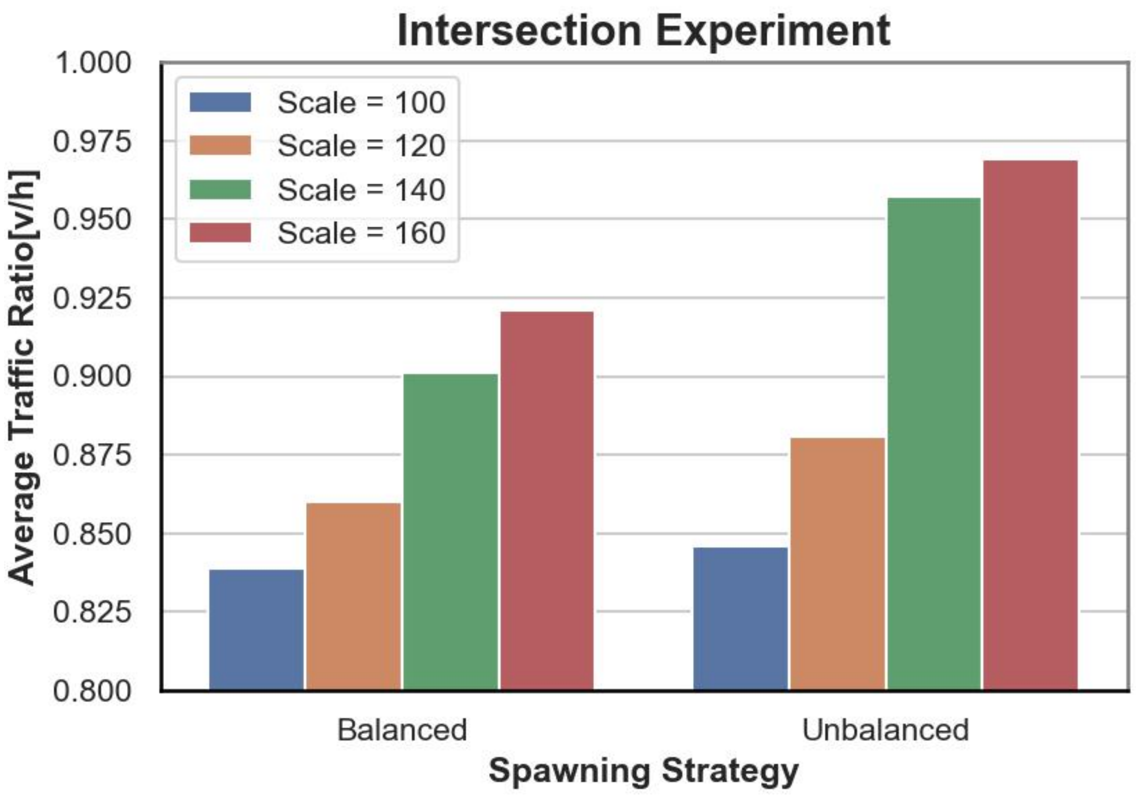

Figure 7.7: Arrive ratio for different magnitudes of traffic flows at non-signalized intersection

traffic volumes, as shown in Figure 7.7 and Figure 7.8. This involved varying the scale values from 100 to 160. Generally, a drop in traffic volume leads to an increase in the traffic ratio. This parallels the highway scenario where fewer vehicles mean less conflict, allowing more seamless intersection passage. Notably, the difference between scale values of 120 and 140 is more distinct than others, indicating the traffic ratio improvements might taper off near the intersection's limit or when vehicle count isn't high enough to cause conflicts.

Lastly, We probed the E-PB method's performance considering varying proportions of

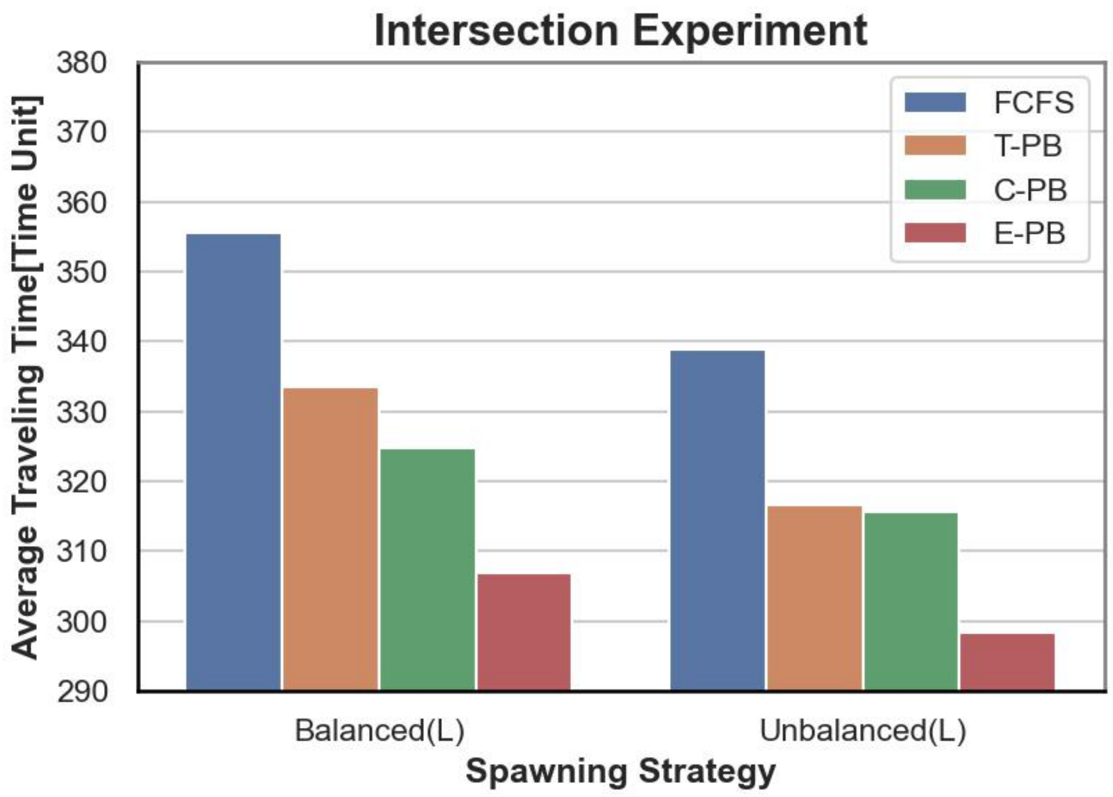

Figure 7.8: Average travelling time for different magnitudes of traffic flows at non-signalized intersection

human-driven vehicles. While these vehicles are connected through onboard systems, their behavior deviates from autonomous vehicles. Specifically, human drivers have longer reaction times to system directives, mainly because of inherent human response delays and multitasking. Within the CARLA simulator, we emulated this by adding a reaction delay of 0.5ms to 1s for human-driven vehicles after they receive system directions. Additionally, human-driven vehicles might not strictly follow system-recommended speeds. We simulated this by introducing a random speed deviation of 0% to 20% for these vehicles.

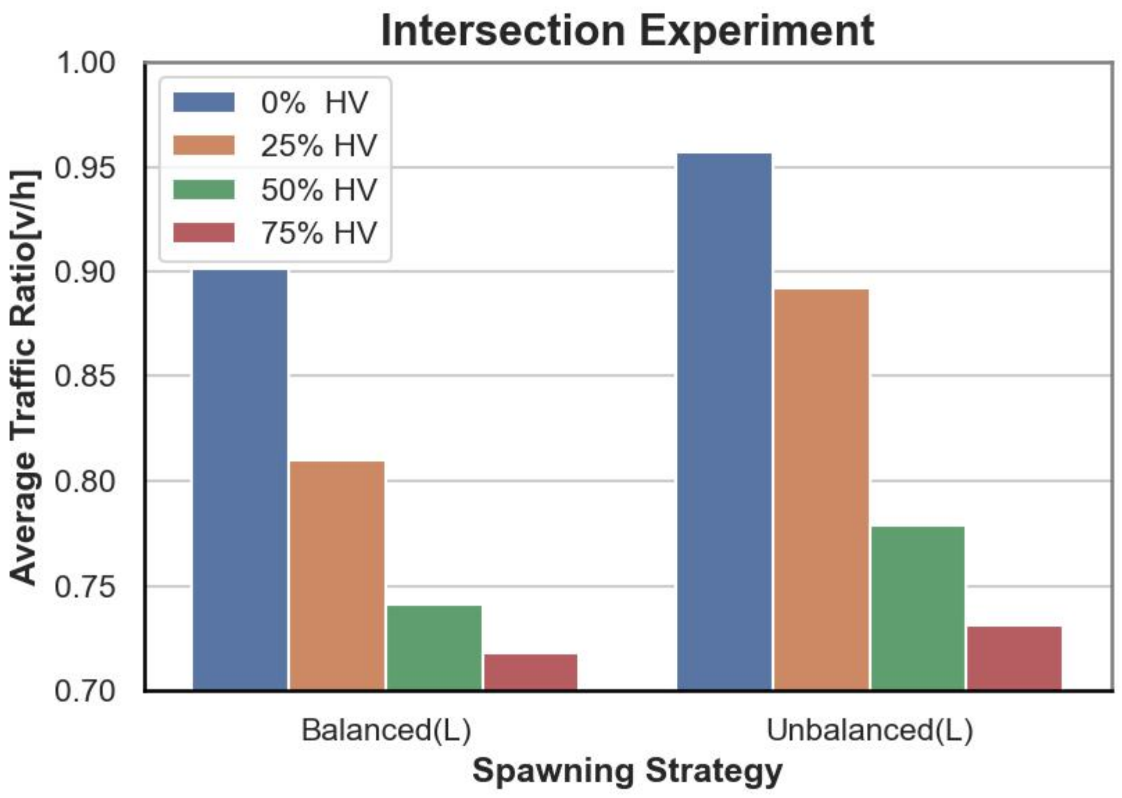

Figure 7.9: Arrive ratio for varying proportion of human-driven vehicles at non-signalized intersection

Figure 7.9 and Figure 7.10 show that as human-driven vehicles (HV) increase, performance wanes in both balanced and unbalanced traffic conditions. Yet, this drop in performance seems to stabilize once HV cross the 50% mark, implying the system's adaptability may peak when a majority of vehicles don't strictly adhere to directions. Interestingly, comparing 0% to 25% HV reveals that the system is more resilient in unbalanced traffic situations.

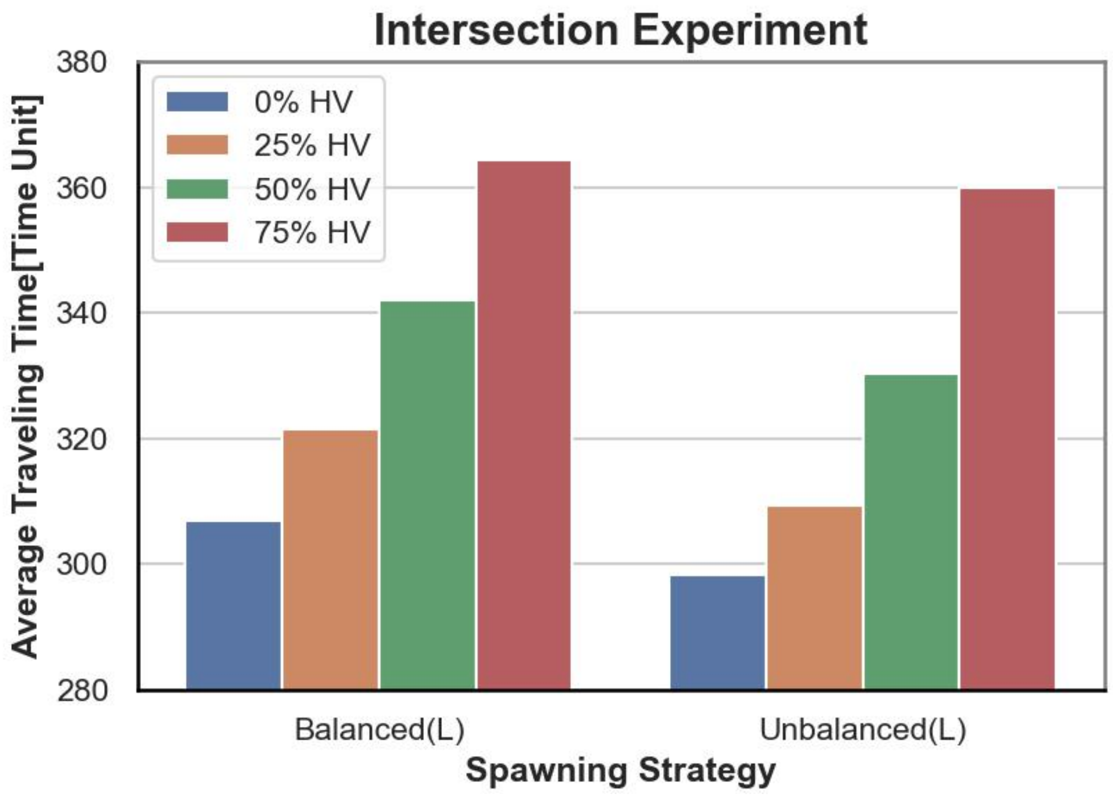

Figure 7.10: Average travelling time for varying proportion of human-driven vehicles at non-signalized intersection

Chapter 8

Related works

The versatility of edge computing makes it a viable solution for supporting autonomous driving [19]. In this realm, Tang et al. [20] proposed $\pi-$Edge for various autonomous driving services in low-speed and limited traffic scenarios, such as university campuses. This approach demonstrates the potential of energy-efficient and affordable edge solutions in autonomous vehicular applications. Meanwhile, guaranteeing ultra-reliable low latency remains a challenge for autonomous vehicles relying solely on limited onboard intelligence [21]. To address this, Lu et al. proposed a cognitive Internet-of-Vehicles system, leveraging technologies like edge computing, IoT, and AI [21]. Additionally, other studies have explored areas like edge-based smart parking [22], the efficient placement of edge computing devices [23], and utilizing vehicle on-board computers as edge devices [24].

Expanding on these findings, recent studies by Hashash et al. [25] and Wang et al.

[26] delve deeper into the application of Edge-based digital twin systems for autonomous vehicles. Hashash et al. [25] focus on a novel edge continual learning framework to maintain synchronization and accuracy in digital twins, which aligns with our research in real-time traffic simulation. Their approach explicitly minimizes desynchronization time, a critical aspect of real-time responsiveness. On the other hand, Wang et al. [26] demonstrate the practical application of digital twins in real-world autonomous driving, particularly in route planning to avoid heavy traffic for long-distance journeys. While Wang et al. focus on general route optimization, our research targets explicitly distinct traffic environments, such as highways and non-signalized intersections, extending the applicability of digital twins in more varied and complicated traffic scenarios.

A wealth of research has tried to pave the way for more efficient intersections in a future with a high penetration of CAVs. Pioneering work by Dresner et al. [9] introduced a reservation-based cooperative driving concept, deploying a First-Come-First-Serve (FCFS) strategy at non-signalized intersections. Their approach surpasses traditional intersection designs in terms of efficiency. Following this work, researchers yielded various types of scheduling schemes for adaptive traffic signals [27], centralized resource reservation [28] [10] [29] [30], distributed protocols [31] [8], optimized by platooning strategy [32].

For centralized resource reservation, the paramount challenge lies in optimizing the limited space at intersections to facilitate the maximum number of vehicles to pass through both safely and efficiently. For example, Dresner et al. [9] evaluated the effectiveness of the

First-Come-First-Serve (FCFS) method in reducing delay compared with signal-based controls. Wang et al. [8] proposed an improved FIFO slot reservation algorithm that ranks vehicles based on their projected arrival times, prioritizing those predicted to arrive earlier. Furthermore, Chen et al. [10] developed a system that prioritizes vehicles according to their potential trajectory conflicts with others, allowing those without conflicts to pass through the intersection simultaneously.

Pioneering studies in cooperative driving on highways have primarily evolved from the lane change models tailored for individual vehicles [33, 34]. Tackling the more intricate task of high-level cooperative lane change decision-making, Sun et al. [35] introduced a comprehensive model focused on lane transitions between neighboring highway lanes, emphasizing vehicular efficiency spanning both lanes. Meanwhile, Zhou et al. [36] presented varied cooperative driving tactics for mixed four-lane highway traffic, illustrating that amplifying the CAV penetration rate can effectively temper traffic congestion while amplifying traffic capacity and stability.

Our work expanded the applicability of centralized scheduling paradigm to include both non-signalized intersections and highways. Unlike the grid box method that is widely used in recent works, TG offers a more adaptable representation of diverse geographical traffic scenarios and streamlines subsequent processes. The TG-powered system can extract more sophisticated traffic patterns and, hence, accurately predict future flows in non-signalized intersection. This prediction capability allowed us to introduce an advanced priority-based scheduling policy. Our findings highlight this policy superiority over previous methods such as FCFS, arrival time-based, and potential conflict-based approaches. On highways, unlike many studies that focus solely on CAVs with fixed behaviors, our strategy is built to handle mixed traffic that include both CAVs and human-driven vehicles. We emphasized the effectiveness of our approach on a five-lane highway environment facing central obstructions, demonstrating noticeable improvements in the traffic flow efficiency.

Chapter 9

Conclusion and future work

We introduced a new traffic scheduling system that leverages concepts from digital twins, edge computing, and machine learning. One of the core elements of the new traffic scheduling system is the *topology graph* (TG) that represents vehicle and road states. This thesis shows the adaptability of TG for various traffic settings like highways and non-signalized intersections. For non-signalized intersections, LSTM-based traffic flow predictors were used to effectively handled diverse traffic flows, prioritizing vehicles by considering both current and upcoming traffic. This approach improved traffic flow efficiency in non-signalized intersections. Our experiments using the CARLA simulator demonstrated this approach surpassing previous research approaches. On highways, we demonstrated that cooperative driving between CAVs and traditional vehicles is feasible. We achieved this by predicting human driver intentions and making reservations through

TG. Our step-by-step motion predictor accurately forecasts human-driven vehicle movements, avoiding accumulated error issues commonly associated with prior sequence-to-sequence predictions. This approach also enhanced traffic efficiency surpassing traditional traffic management as shown by the experiments carried out in the CARLA simulator.

The research presented in this thesis work can be improved in many different ways. One avenue of future work is to investigate how to integrate an autopilot system based on reinforcement learning. The autopilot will work with a traffic generator to mimic real-world traffic under different scenarios. This will allow us to capture a wider range of traffic patterns, enhancing the precision of our predictions. Specifically, on highways, we aim to simulate real-world scenarios, including emergency situations, and refine vehicle localization with more realistic lane-changing trajectories.

Bibliography

[1] "How Tesla and Google autonomous car technologies differ." https://analyticsindiamag.com/tesla-google-autonomous-car-technologies-different/. Accessed: 2019-04-14.

[2] M. Maheswaran, T. Yang, and S. Memon, "A fog computing framework for autonomous driving assist:architecture, experiments, and challenges," in *CASCON '19: Proceedings of the 29th Annual International Conference on Computer Science and Software EngineeringNovember*, p. 24–33, ACM, 2019.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[5] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, pp. 117–122, 2005.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] I. Sorkhoh, C. Assi, D. Ebrahimi, and S. Sharafeddine, "Optimizing information freshness for mec-enabled cooperative autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13127–13140, 2022.

[8] Z. Wang, K. Han, and P. Tiwari, "Digital twin-assisted cooperative driving at non-signalized intersections," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 198–209, 2021.

[9] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[10] X. Chen, M. Hu, B. Xu, Y. Bian, and H. Qin, "Improved reservation-based method with controllable gap strategy for vehicle coordination at non-signalized intersections," *Physica A: Statistical Mechanics and its Applications*, vol. 604, p. 127953, 2022.

[11] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9554–9567, 2022.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6272–6281, 2019.

[14] Y. Huang, H. Dai, and V. S. Tseng, "Periodic attention-based stacked sequence to sequence framework for long-term travel time prediction," *Knowledge-Based Systems*, vol. 258, p. 109976, 2022.

[15] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1672–1678, 2018.

[16] L. You, S. Xiao, Q. Peng, C. Claramunt, X. Han, Z. Guan, and J. Zhang, "St-seq2seq: A spatio-temporal feature-optimized seq2seq model for short-term vessel trajectory prediction," *IEEE Access*, vol. 8, pp. 218565–218574, 2020.

[17] X. Feng, Z. Cen, J. Hu, and Y. Zhang, "Vehicle trajectory prediction using intention-based conditional variational autoencoder," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3514–3519, 2019.

[18] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using lstms with spatial–temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2022.

[19] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, pp. 1697–1716, Aug 2019.

[20] T. Jie, L. Shaoshan, Y. Bo, and S. Weisong, "Pi-Edge: A low-power edge computing system for real-time autonomous driving services," *arXiv preprint arXiv:1901.04978*, 2018.

[21] H. Lu, Q. Liu, D. Tian, Y. Li, H. Kim, and S. Serikawa, "The cognitive internet of vehicles for autonomous driving," *IEEE Network*, vol. 33, pp. 65–73, May 2019.

[22] H. Bura, N. Lin, N. Kumar, S. Malekar, S. Nagaraj, and K. Liu, "An edge based smart parking solution using camera networks and deep learning," in *2018 IEEE International Conference on Cognitive Computing (ICCC)*, pp. 17–24, July 2018.

[23] P. Gopika, G. Bissan, D. F. Mario, and V. Rudi, “Efficient placement of edge computing devices for vehicular applications in smart cities,” in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2018.

[24] S. S. Sarmad, A. Muhammad, M. A. Waqar, K. M. A, and R. S. Devi, “vFog: A vehicle-assisted computing framework for delay-sensitive applications in smart cities,” *IEEE Access*, vol. 7, pp. 34900–34909, 2019.

[25] O. Hashash, C. Chaccour, and W. Saad, “Edge continual learning for dynamic digital twins over wireless networks,” *arXiv preprint arXiv:2204.04795*, 2022.

[26] K. Wang, T. Yu, Z. Li, K. Sakaguchi, O. Hashash, and W. Saad, “Digital twins for autonomous driving: A comprehensive implementation and demonstration,” *arXiv preprint arXiv:2401.08653*, 2023.

[27] Y. Wang, X. Yang, H. Liang, Y. Liu, *et al.*, “A review of the self-adaptive traffic signal control system based on future traffic environment,” *Journal of Advanced Transportation*, vol. 2018, 2018.

[28] Y. Zhang, L. Liu, Z. Lu, L. Wang, and X. Wen, “Robust autonomous intersection control approach for connected autonomous vehicles,” *IEEE Access*, vol. 8, pp. 124486–124502, 2020.

[29] J. Wang, X. Zhao, and G. Yin, “Multi-objective optimal cooperative driving for connected and automated vehicles at non-signalised intersection,” *IET Intelligent Transport Systems*, vol. 13, no. 1, pp. 79–89, 2019.

[30] W. Zhao, R. Liu, and D. Ngoduy, “A bilevel programming model for autonomous intersection control and trajectory planning,” *Transportmetrica A: transport science*, vol. 17, no. 1, pp. 34–58, 2021.

[31] A. I. M. Medina, N. Van De Wouw, and H. Nijmeijer, “Cooperative intersection control based on virtual platooning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1727–1740, 2017.

[32] S. D. Kumaravel, A. A. Malikopoulos, and R. Ayyagari, “Optimal coordination of platoons of connected and automated vehicles at signal-free intersections,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 186–197, 2022.

[33] L. Yang, R. Tang, and K. Chen, “Call, put and bidirectional option contracts in agricultural supply chains with sales effort,” *Applied Mathematical Modelling*, vol. 47, pp. 1–16, 2017.

[34] H. Bai, J. Shen, L. Wei, Z. Feng, *et al.*, “Accelerated lane-changing trajectory planning of automated vehicles with vehicle-to-vehicle collaboration,” *Journal of Advanced Transportation*, vol. 2017, 2017.

[35] K. Sun, X. Zhao, and X. Wu, “A cooperative lane change model for connected and autonomous vehicles on two lanes highway by considering the traffic efficiency on both lanes,” *Transportation Research Interdisciplinary Perspectives*, vol. 9, p. 100310, 2021.

[36] Y. Zhou, H. Zhu, M. Guo, and J. Zhou, “Impact of cacc vehicles’ cooperative driving strategy on mixed four-lane highway traffic flow,” *Physica A: Statistical Mechanics and its Applications*, vol. 540, p. 122721, 2020.