UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Optimization Methods Leveraging V2X Communication and Traffic Management Apps for Transportation Control

Permalink
https://escholarship.org/uc/item/6pp9h0vm

Author
Burov, Mikhail Igorevich

Publication Date
2022

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Optimization Methods Leveraging V2X Communication and Traffic Management Apps for Transportation Control

by

Mikhail Igorevich Burov

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Murat Arcak, Chair
Professor Joan Walker
Professor Somayeh Sojoudi

Spring 2022

Optimization Methods Leveraging V2X Communication and Traffic Management Apps for Transportation Control

Copyright 2022
by
Mikhail Igorevich Burov

Abstract

Optimization Methods Leveraging V2X Communication and Traffic Management Apps for Transportation Control

by

Mikhail Igorevich Burov

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Murat Arcak, Chair

With a growing number of vehicles and an increasing complexity of transportation systems, traffic management and traffic optimization become more and more crucial in mitigating congestion, reducing travel delay and improving traffic state. The concept of a "Smart" city that incorporates various intelligent systems related to infrastructure modification, wireless communication, networking and centralized/decentralized controllers is considered to be the next evolutionary stage of the modern urban world.

This dissertation focuses on optimization models and algorithms leveraging traffic management apps (navigation and reservation) and vehicle-to-everything (V2X) communication for mitigating congestion level, reducing fuel consumption and minimizing travel delay for vehicles in urban areas. An ability to share data, receive and send requests and directions allows traffic agents, both on-road (moving) and off-road (parked), to significantly improve utilization of transportation resources. To demonstrate the impact network-level control policies have on a system's social delay, in our work [1], presented in Chapter 2, we propose a greedy optimization algorithm that eliminates "Braess" routes and derives a paradox-free subnetwork to be implemented in navigation apps. Prior literature that studied the Braess paradox was not able to provide efficient tools for improving the equilibrium state. Topology analysis methods could only predict the occurrence of the paradox but could not deal with it. Results that focused on a single link or route removal were ineffective for large networks. Other methods that discussed tolling or closing roads were too restrictive and required significant infrastructure modifications. Our approach, on the other hand, is more flexible and can be effectively applied to real-world systems to completely eliminate the inefficiency and momentarily reduce total travel time. In addition, we address the challenging task of incorporating queue delay into the network representation by introducing "phantom links". The following chapters focus on link-level models dealing with local traffic inefficiencies. In Chapters 3 (corresponding work [2]) and 4 (extension to the work [3]) we explore moving

traffic management methods and benefits their implementation has with respect to traffic throughput, travel time and fuel consumption. We demonstrate how the optimal platoon formation algorithm can improve traffic progression on urban streets and freeways. Earlier methods either focused on the Ad-hoc protocols, which have limited application in mixed traffic due to its short range, or tried to reduce the travel delay at the cost of the increased traffic disturbance, which was both ineffective and potentially harmful. Our approach, aimed at travel time minimization, takes on the local clustering method and proposes an intelligent platoon merging system that addresses a major part of its common complications and difficulties. In particular, minor infrastructure modifications accompanied by V2I communication protocols enable efficient localization and coordination with minimal traffic disruption and make it possible to artificially increase road capacity and improve congested regions. For fuel consumption, we discuss a novel queue estimation procedure, built upon the vehicle labeling system presented in [3], to be used in a prediction-based model that ties together Speed Advisory System and actuated traffic lights to reduce idling at intersections and smooth driving patterns. Moving on from thru-traffic optimization, in Chapter 5 we focus on curb management issues and propose a delivery vehicles' operation hours partitioning model to be implemented in a reservation app. Previous works that discussed parking reservations did not consider delivery vehicles and their specific arrival patterns and focused mostly on general cars. Another commonly used parking policy, dynamic parking pricing, extensively studied in the past, is not effective when dealing with delivery vehicles, since they do not usually pay for parking. Our system, on the other hand, focuses on implementing parking equilibrium by physically constraining delivery trucks' parking choices. In particular, this model is aimed at eliminating double-parking. In Chapter 6 we start looking at practical implementation of curb management techniques. More specifically, we investigate a new delivery vehicle activity monitoring method via dashcam video footage and data analysis algorithms. Switching from the non-scalable and costly data collection techniques, such as static surveillance, we propose a more flexible, low-cost and effective model for parking patterns recognition and curb management implementation.

Contents

List of Figures

List of Tables

Acknowledgments

At the top of my "deep gratitude list" I thank my advisor, Professor Murat Arcak, who has been a great mentor for all my five years in graduate school. I am grateful to him for his support and understanding, his wisdom and guidance that helped me grow as a researcher and a person. I also thank Murat for his encouragement that I find science projects that most interest me.

My graduate school experience would be much more thorny without the great support from Alexander Kurzhanskiy, who I consider to be my unofficial mentor during my time at UC Berkeley. I value many pieces of advice he gave me and the spot-on observations he shared with me regarding research, studies, and life in general. Being an international student, I appreciate the helping hand from a comrade. Thanks also for being on my qualifying exam committee.

I applaud Stan Smith for being a perfect Ph.D. student example who I could look up to during my journey. I appreciate Stan's patience when answering thousands of questions I had about preliminary and qualifying exams, dissertation writing, internships, and many other topics. I am also grateful for Stan's insights on platooning which were extremely valuable for my research. Thanks as well to Can Kizilkale for his collaboration on the last two projects concerning Braess routes and parking reservations. Can's knowledge and experience helped a lot in developing models for traffic management.

Thanks to Professor Joan Walker and Professor Somayeh Sojoudi for being on my qualifying exam and dissertation committees. I would also like to thank Somayeh for having me as a GSI in EE 128/ME 132 class in Spring 2022. Thanks to Professor Sanjit Seshia for making my first GSI experience (EECS 149/249A) smooth and exciting. My gratitude to four fellow GSIs across both classes: James Smith, Shaokai Lin, Wayne Lin, and Paul Botros. They were awesome and fun teammates.

Thanks to the people of the transportation research group: Professor Roberto Horowitz, Professor Pravin Varaiya, Ruolin Li, and Negar Mehr, for passionate discussions, brilliant ideas, and critical evaluation of my research progress that helped me stay on the right track, grow as a researcher and hone my presentation and discussion skills. Thanks as well to the MEng students who I worked with on the curb monitoring project and who made a major contribution to the project development: Aditya Kumar, Priya Jindal, Aravinth Paranan, and Sida Li.

I would also like to mention all the great people in the Arcak lab and thank them for a warm welcome, fun conversations, and support throughout my Ph.D. journey: Stan Smith, Eric Kim, Mindy Perkins, Pierre-Jean Meyer, Galaxy Yin, Kate Schweidel, Emmanuel Sin, Alex Devonport, and Neelay Junnarkar.

Graduate school is not all about research, coursework, and exams. It is also about new people, new experiences, and a new lifestyle. I am very grateful to everybody who helped me preserve the life-work balance and enjoy my time at Berkeley. Thanks to my first roommates - Thomas Brown, Jacub Kmec, Jose Cruz, Borna Maghoul, and Shahar Syed - for entertaining evenings and weird but fun conversations. Special thanks to Alex Moreno who has been my

roommate for the last 3 years, who shared many memorable moments with me, and who has been a true friend for all these years. Thanks as well to Tyler Reichanadter for bringing soccer back into my life and helping me out when I most needed it. I would also like to thank all my friends who I knew way before grad school and who I kept in touch with since, especially Savanah Wiles, Nikita Komarov, and Andreii Ryazanov. You reminded me that real friendship can not be severed by distance or time.

Special thanks to Philip Clapick, my former host father from the time I was a teen exchange student in the US, who opened a world of opportunities for me and sparked my interest in coming back for education. Of course, thanks to my brother Oleg and the entire family in Russia for their support and encouragement. Thanks to my mother Tatyana and grandmother Anna for their love and everlasting trust in me. Without their efforts and help, I would never achieve what I have achieved and would never be the person I am today. Finally, thanks to my partner Anastasia for filling my life with joy and love and making this journey easier.

Chapter 1

Introduction

The current state of transportation system management is far from being optimal. With a growing number of vehicles ([4]), urban streets and freeways are failing to efficiently accommodate heavy traffic loads, resulting in high congestion levels ([5]) in the US. Straightforward solutions, such as building additional roads, although intuitively should solve the existing issues, are not only extremely expensive considering limited spatial resources of cities, but might also have a completely unexpected effect: increased congestion and travel time (this phenomenon is discussed in Chapter 2). Therefore, more elaborate control and optimization methods are required to analyze and manage transportation systems.

The concept of a "Smart city" (studied in [6], [7], etc.), can be a potential solution to the existing transportation crisis due to its versatility, flexibility and efficient resource utilization. It is a collection of various intelligent systems serving a single purpose: creating a complete monitoring and control infrastructure for effective traffic management. Intelligent safety measures, such as "safety islands", protected turns and pedestrian protection can minimize hazardous situations, prevent accidents and accident-related traffic issues. Different types of wireless communication, such as vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-everything (V2X) or Internet-based (navigation apps), allow traffic agent to share data with the system. Providing GPS location, speed profile, ride information, surrounding traffic conditions, path choice or driver's preferences contributes to building a complete and accurate online system representation and enables fast and optimal response from the infrastructure. Advanced sensors, detectors and traffic cameras constituting a complex monitoring network are responsible for building detailed data-sets for behavioral pattern recognition, policy enforcement and analysis. The "brain" of a Smart city responsible for making decisions and passing down commands and directions can be represented by a collection of centralized and/or decentralized controllers relying on many different optimization models. These models programmed to prioritize one or several traffic parameters (travel time, parking occupancy, fuel consumption, air or noise pollution, etc.) are a powerful tool in eliminating inefficiencies. The final structure of the "brain" may be different for every region: a single ultimate controller responsible for the entire transportation system in one city or many decentralized controllers limited to a single task, such as managing a traffic

light, in another one. Building and implementing such optimization models and controllers is the main objective of traffic management.

1.1 Optimization Models in Transportation Management

This dissertation focuses on several optimization models and algorithms for transportation systems' management leveraging V2X communication, navigation and parking reservation apps. The overall structure of this work is shown in Fig. 1.1. In the first part of the dissertation, consisting of Chapters 2, 3 and 4, we will demonstrate optimization algorithms and models related to moving traffic (i.e. thru-traffic) management. First, focusing on network-level control policies in Chapter 2 (the work was presented at ITSC 2021 - [1]), we will continue with the link-level models in Chapters 3 (the work was presented at ITSC 2020 - [2]) and 4. The second part of the dissertation, Chapters 5 and 6, explores optimization methods in application to static agents (parking policies) within the link-level models. A parking reservation model, presented in Chapter 5 will be accompanied by a practical monitoring system discussed in Chapter 6. The rest of the introduction is dedicated to demonstrating the interconnection and interdependency of all part of the dissertation, as well as providing further background and motivation for the research.

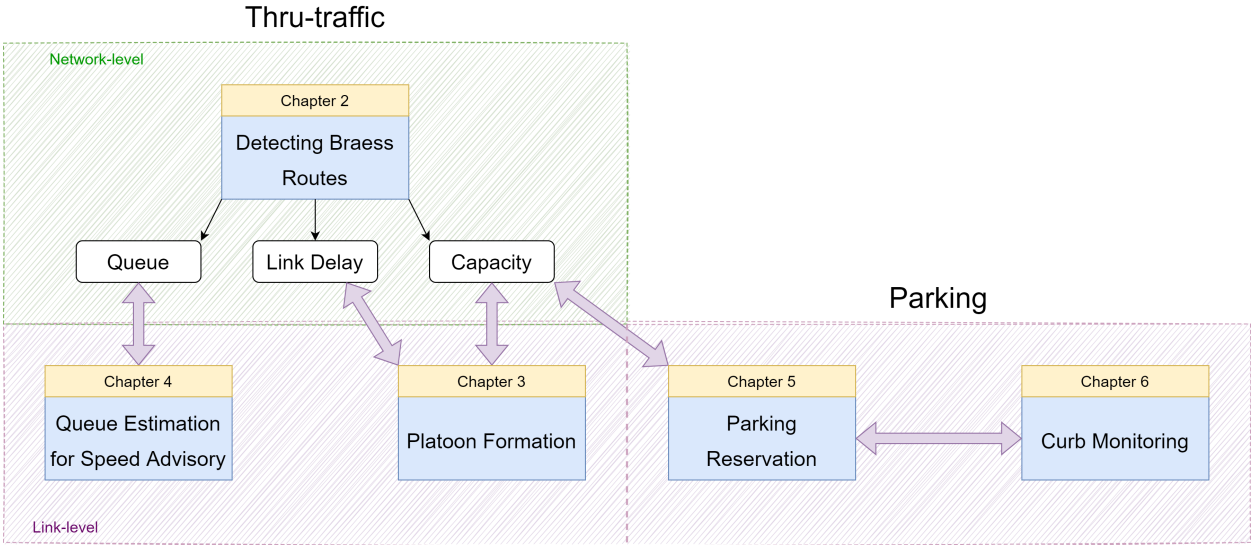


Figure 1.1: Dissertation structure. The work is clustered according to the common themes: moving traffic analysis and parking optimization; Network-level and link-level controls.

1.2 Detecting Braess Routes

The single greatest source of congestion and road utilization inefficiency is the behavioral nature of moving traffic (thru-traffic) itself. Drivers, having limited information about the local (surrounding traffic) and global (the entire network) states of the system, tend to make selfish and non-optimal decisions resulting in significant excessive travel delay. Ordinary traffic agents are not coordinated and unaware of the cumulative effects their behavior has on occurring congestion, therefore, developing optimization and management models for thru-traffic is essential. With the purpose of having a broad understanding of drivers' decisions and their collective impact on traffic state, we will first focus on the highest layer of transportation systems: *Network level*. At this layer traffic management is closely related to adding or removing available resources (roads or routes) and directing traffic flows within the system.

Unfortunately, due to the selfish nature of drivers' intentions when choosing their strategies, trivial solutions might be ineffective or even harmful. When planning their trips, non-altruistic drivers tend to follow routes with smallest immediate delay (best-response strategy) without caring about the well-being of the transportation system. Such behavior is well characterized by Nash (Wardrop) equilibrium state, the system state at which no agent (driver) benefits from individually switching his strategy (route). Being at equilibrium state, however, is not equivalent to exhibiting minimum travel delay, which means road networks often suffer from various types of equilibrium inefficiencies. The well-known Braess paradox, first studied in [8], is one of such inefficiencies. This counter-intuitive phenomenon describes scenarios where adding resources (building additional roads or introducing new routes) results in higher cost (travel time) for agents at equilibrium. One way to deal with the paradox and reduce the social cost of the system at equilibrium is to reverse the process by removing such Braess links or routes from the network bringing it back to the pre-Braess state. This approach does not guarantee the optimality of the resulting network ([9] proved the problem of finding the optimal subnetwork is NP-hard), but is capable of significantly improving congestion.

The first solution corresponding to adding and removing road links, when applied in real-world, requires effort-demanding physical modifications to the network (closing down roads) and might be challenging to implement. On the other hand, in the era of Google Maps, when most people rely on navigation apps, altering the pool of suggested routes is much easier and efficient. First, it only requires software-related adjustments to the online maps; second, it does not affect harmless (congestion-wise) short-range trips, since no links are closed down completely. Therefore, in Chapter 2 we present a route elimination algorithm that searches for Braess routes and removes them from the navigation system, leaving only a paradox-free subset visible to agents.

But first, to analyze a physical transportation system and apply optimization methods, we need to build a mathematical representation of it. We can transfer the problem to the graph modeling domain and portray the network as a combination of edges and nodes representing roads and intersections respectively (Fig. 1.2).

Each link has a corresponding delay function dependent on the link's geometry, capac-

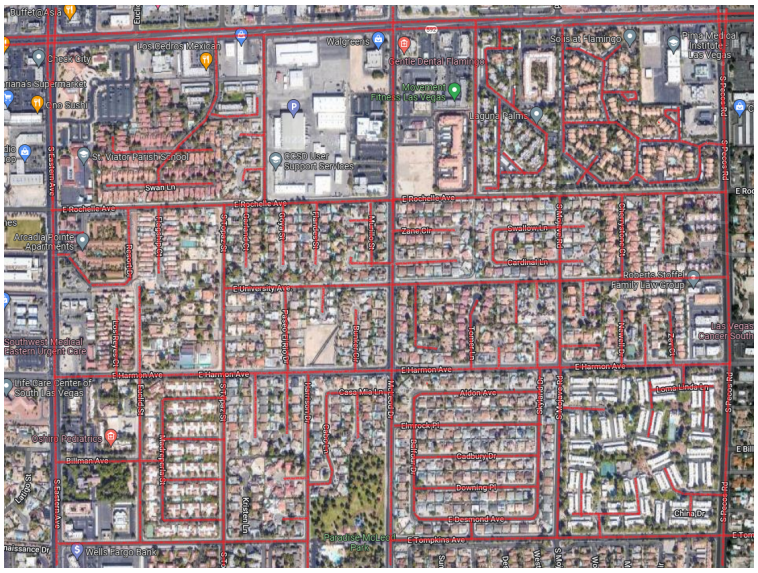


Figure 1.2: Graph representation of a real-world network in Las Vegas, NV. Graph links and nodes represent road segments and intersections respectively.

ity, number of lanes, speed-limit, etc. These delay functions must be non-negative, non-decreasing and continuous to reflect the real-world properties of travel delay and be suitable for convex optimization methods used to derive the Nash state. One of the drawbacks of a graph representation is difficulty in modeling traffic queues and the resulting delay. In Chapter 2 we address this challenge by introducing "phantom" links accounting for queuing delay. The further equilibrium derivation can be conducted using game theory, more specifically population and congestion games, and convex optimization methods utilizing Beckmann potentials.

Equilibrium analysis and optimization rely heavily on the throughput capabilities of road links and intersections. However, high-level management techniques are not able to affect these characteristics, and thus, to be most effective, must be accompanied by link-level control methods, which can help in analyzing and regulating the links' flows. In the following chapters we will explore some of these link-level models.

1.3 Platoon Formation Optimization

The capacity and speed-density characteristics of a road link are the two most important parameters defining the link's delay function and the saturation rate. Here a link's saturation rate is defined as a maximum throughput in vehicles per hour achieved when the traffic flow on the link is continuous and uninterrupted. The ability to alter these parameters can potentially improve local traffic propagation and, as a consequence, reduce travel time. One of the transportation concepts capable of such regulation is *vehicle platooning* ([10]). A

platoon is a string of connected vehicles (CV) moving in a tight formation on the road (Fig. 1.3). Due to the shorter headway maintained between the platooning vehicles, more road space is made available for other traffic agents, which contributes to the road capacity increase.

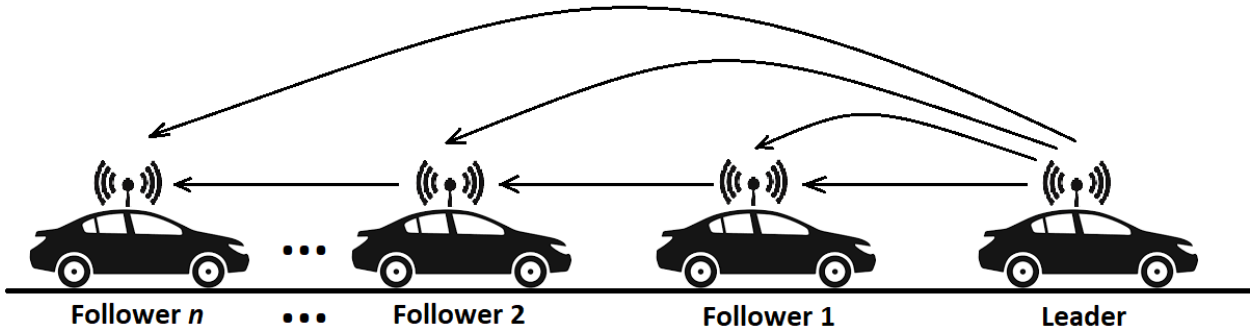


Figure 1.3: Vehicle platooning diagram. The leading vehicle shares data with all its followers, while other vehicles communicate only with their immediate followers.

Connected vehicles, as the name suggests, are capable of wireless connection or communication via V2V communication channels. In order to preserve a constant inter-vehicle distance while accelerating, cruising and decelerating, platooning vehicles must coordinate their motion. Maintaining string stability (more information on string stability can be found in [11]) requires distance tracking and estimated speed trajectory sharing ([12], [13]). Removing human reaction time from the equation allows for an immediate response to any sudden velocity changes. Such coordinated movement also helps in mitigating traffic shock-waves.

Forming platoons in heavy traffic, however, can be challenging in practice. Assuming local clustering mechanism, where CVs willing to platoon are scattered within a relatively short proximity on the road, the process of coordinating vehicles and merging them in a platoon is non-trivial. Identifying relative positions and current lanes, deciding upon the order and terminal lane without a centralized controller might be rather harmful than beneficial. Multiple vehicles simultaneously performing maneuvers, such as lane changing, slowing down or speeding up, can potentially disrupt the flow of surrounding traffic and risk the safety of other traffic agents. To address this issue, in Chapter 3 we introduce a platoon formation algorithm aimed at minimizing travel time built upon certain infrastructure modifications. These changes include but are not limited to allocating a platoon-dedicated lane (similar to a carpool lane) and installing Road-Side-Units (RSU). The latter not only enables V2I communication for movement coordination and guidance, but also allows for traffic data collection for further analysis.

1.4 Queue Estimation for Speed Advisory and Real-Time Phase Length Prediction

While in Chapter 3 we worked on improving traffic progressions on roads, in Chapter 4 we focus on traffic progression at intersections. Of a particular interest are signalized intersections where the traffic flow is regulated by a traffic light (TL). The most common, "fixed", type of a traffic light, with a constant cycle length (time interval between two consecutive green lights) and predefined phase durations (duration of green, red and yellow lights) can be considered a baseline for traffic control due to its trivial functionality. A more advanced version, *dynamic traffic light*, on the other hand, has a greater impact on traffic progression and intersection throughput. Depending on the flow volume, its phase length can be automatically extended beyond the minimum duration up to a certain limit. This extension is triggered by travelling vehicles themselves and does not require human supervision. Ability to provide longer green-light intervals to directions with higher demand can help in dissolving queues and improving traffic regulation. However, the dynamic nature of phase duration poses an implementation challenge for external intelligent systems that rely on the knowledge of TL states and the remaining time in the current phase.

One of such systems is a Speed Advisory System (SAS), introduced in [14], and designed to provide near-optimal speed trajectories reducing fuel consumption and minimizing idling at intersections. By constructing bang-singular speed trajectories, a more driver-friendly version of a bang-singular-bang pattern, SAS manages to reduce fuel-inefficient jerky movement and guide vehicles through upcoming intersections without stopping (if possible). The latter property of near-optimal speed profiles significantly improves progression quality, which is closely related to queuing delay at an intersection due to arrival-departure patterns.

To perform required computations, however, the Speed Advisory System relies on the availability of infrastructure parameters, such as distance to the intersection, speed limit on the road, TL phase and the remaining time within the current phase. When interacting with fixed traffic lights, connected vehicles can obtain this information via SPaT (signal, phase and timing) messages, but since this information is unknown in case of dynamic TL, some estimation technique is required.

In our work ([3]) we proposed a real-time phase length prediction algorithm that is capable of providing such estimations. Using traffic data from an advanced road detector, which we refer to as a "counter", it evaluates whether a certain vehicle will be able to trigger the phase prolongation at the upcoming intersection. The mentioned traffic data saved in the system for one TL cycle consists of crossing times and vehicles' speeds and is further used to compute an estimated time when each particular vehicle reaches another advanced detector, an "actuator", responsible for phase extension. In this work we assumed the time-gap actuation type, i.e. the phase is extended only if the time gap between two consecutive vehicles crossing an actuator is smaller than a certain threshold. Combining the knowledge of the actuation time gap and recorded arrival times, our algorithm labeled vehicles "PASS" or "WAIT" according to their estimated ability to pass the intersection

within the current phase. The last "'PASS"-labeled vehicle served as an indicator of a phase switch and its estimated arrival time could be used to derive the remaining time within the current phase. Using 2-way V2I communication protocols, the infrastructure shared the estimated parameters with vehicles enabling the Speed Advisory System to take actions. Since this method requires computational power and communication channels for efficient performance, the infrastructure modification must include installation of Road-Side-Units in addition to the discussed advanced detectors (Fig. 1.4).

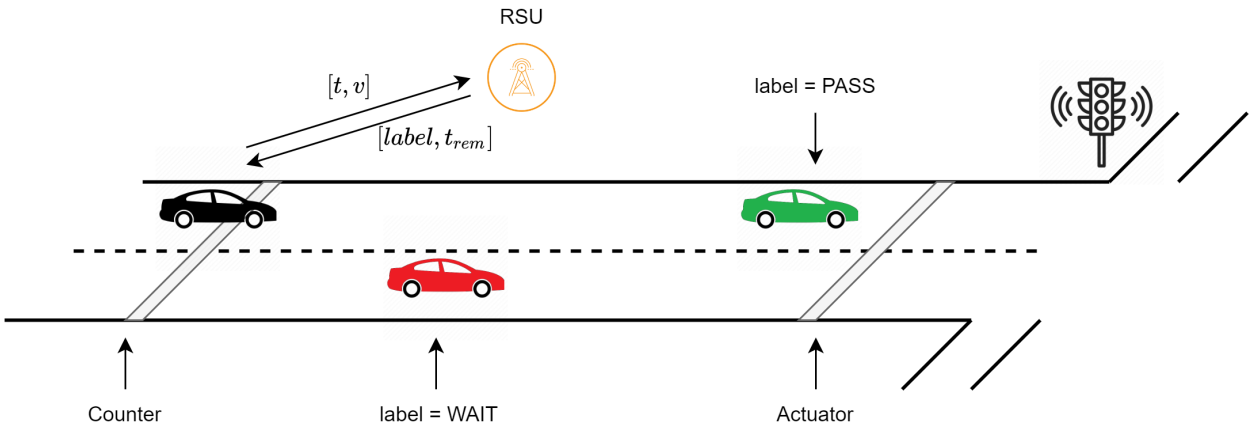


Figure 1.4: Diagram of V2I communication and labeling process. RSU receives speed and time and transmits assigned labels and estimated remaining time withing the phase.

In [3], we did not consider queues which, according to the findings in Chapter 2, make a significant contribution to travel delay and throughput reduction. Ignoring queues means overestimating road space available for completing the SAS-suggested maneuvers, which might partially negate the potential benefits. In Chapter 4, we provide our work [3] for completeness and extend it focusing on queue estimation implementation for a more accurate near-optimal speed trajectories derivation. The proposed method can further reduce fuel consumption and progression quality at intersections equipped with dynamic traffic lights.

1.5 Parking Reservation

Although thru-traffic-related inefficiencies are the major cause of congestion and excessive travel delay, the complete control over a transportation system cannot be achieved by the vehicle flow management alone. A particular group of agents, parked or seeking parking vehicles, whose actions are not affected by the control techniques presented in previous chapters, can still influence local infrastructure parameters and disrupt traffic flow. Most city streets have at least one lane allocated for on-street parking available for ordinary and commercial vehicles on a First-come-First-served basis. This principle together with the lack

of any regulation makes it impossible to ensure parking spot in advance, before arriving at the location. Due to high demand and insufficient parking space, most ordinary drivers are forced to cruise around searching for vacant parking spots slowing down the surrounding traffic ([15]).

Commercial vehicles (delivery trucks, taxis, Ubers, etc.), on the other hand, are more location- and time-sensitive when looking for a spot and usually cannot afford long searching and remote parking. Dropping off passengers or delivering goods require a close proximity to the final destination and minimal possible holdup. Their parking spot occupancy is usually brief, therefore, some drivers may find it expedient to park illegally (double park, occupy no-park zone or a bus lane) in the absence of a rapid law enforcement system (Fig. 1.5). Such behavior results in direct road capacity reduction and a bottleneck effect, which has a negative impact on traffic flow and traffic state in busy areas and during rush hours. Another parking violation concerning bus stop occupation can be considered a safety issue as well. Unable to stop at the bus stations, buses are forced to drop off passengers in less convenient and more dangerous locations exposing them to traffic and further slowing down other drivers.

Figure 1.5: Double-parked FedEx delivery vehicle blocking an entire lane during unloading. The absence of a rapid law enforcement system leaves many of such drivers undeterred.

Some curb management methods aimed at improving utilization and eliminating illegal parking, such as additional space allocation for delivery vehicles, might be effective only to a certain extent (as long as the total parking demand does not exceed the space capacity).

Dynamic parking pricing, commonly used for occupancy regulation, cannot be fully practical, since delivery vehicle drivers generally do not pay for parking due to brief parking time. One way to make sure every arriving delivery vehicle is able to legally park at a particular location is to implement an online reservation system. Drivers having access to a parking app can see available parking spots and book them beforehand, reducing unintentional parking violations.

In Chapter 5 we introduce a parking reservation model for delivery vehicles based on a partitioning of operation hours. By choosing the specific time-slot durations, we manage to physically constrain delivery trucks' parking choices and force them to follow the optimal parking strategy, which is much more effective than applying pricing management techniques. The main objective of our model is eliminating double-parking incidents caused by uncoordinated and overlapping truck arrivals. We consider delivery truck drivers' and city's preferences in the form of utility and penalty functions respectively to derive the optimal parking equilibrium. By solving series of optimization problems we split delivery vehicles' operation hours into non-overlapping intervals resulting in maximum social utility. The derived partitioning is expected to be made available for booking in an online reservation app which can also be designed to collect users' preferences and travel data for a more accurate action prediction.

1.6 Curb Monitoring

In Chapter 5 we discussed the optimization model capable of mitigating double-parking and improving curb utilization, but what is necessary for an actual real-world implementation of such systems? Which locations are in high demand among delivery vehicles and suffer the most from double-parking? What is the parking demand distribution throughout the day, week, month, year? How do delivery vehicles' behavior patterns affect the overall congestion level in the system? How effective and rapid is law enforcement in specific areas? The uncertainty in spatial and temporal parking characteristics of an area of interest makes the curb management a challenging engineering task.

The initial phase of any parking policy application is data collection and monitoring. A city planner needs to identify problematic areas, which we refer to as "hot spots", with the highest traffic density and the day-to-day delivery vehicles number. In case of well-developed social system built within a community with regular surveys and questionnaires for drivers, residents, business owners, etc., some human-related perspective can be obtained through civic agencies. Explicit monitoring can be achieved with an installation of surveillance cameras recording the traffic state 24/7. The quality and completeness of the obtained data would be exceptional, however, potential privacy issues, relatively small area of consideration and high maintenance costs may force a designer to seek other monitoring methods. Staying on the video-related side of data collection, having access to personal or public dashcam footage reflecting the vehicle-point-of-view picture can be a good alternative to fixed cameras. The obtained information is expected to be less thorough and complete, but, on the

other hand, it allows to analyze the system from various angles and evaluate the direct impact delivery trucks have on the surrounding traffic (speed variation, lane changing maneuvers, etc.). On-street advanced detectors and sensors collecting speed-density points and parking occupancy can also contribute to building an accurate representation of the transportation system. Further analysis using modern data-science and artificial intelligence methods, such as neural networks, can produce a solid ground for implementing curb management models.

Chapter 6 starts looking at the practical implementation of the monitoring phase. We focus on curb activity evaluation and delivery vehicles recognition on the South side of UC Berkeley campus, Bancroft Way, Berkeley, CA. By installing a dashboard camera on the Bear Transit bus cruising around the campus, we managed to collect hundreds of hours of video footage for extensive traffic analysis. We developed and trained YOLOv5-based neural network for delivery vehicle recognition and classification. Labeled data was used to discover busy areas, construct temporal distribution of delivery activity and analyze the delivery vehicles' parking violations.

Chapter 2

Detecting Braess Routes

2.1 Introduction

Road networks suffer from various types of equilibrium inefficiency due to selfish routing. Of particular interest is the Braess paradox, a counter-intuitive phenomenon describing scenarios in which building new road links results in higher traffic delays at equilibrium. Since its introduction in 1968 in [8], the Braess paradox has been studied extensively to find efficient ways to predict, detect and prevent its existence.

Early results, such as [16], [17], [18], [19], [20], focus on the classic diamond-shaped, four-node network with a single OD-pair. A later study [21] extends the analysis to general traffic networks to predict the occurrence of the paradox; however, the applicability of the results is limited by a restrictive assumption that all routes with non-zero flows in the original network are also utilized after the addition of a road link. Moreover, [21] as well as related theoretical papers [22] and [23] consider the special case when exactly one road link or route is built or removed from the network.

Another approach to anticipating the Braess paradox is a network topology analysis. The study [24] shows that a two-terminal network is immune to Braess paradox if and only if it is series-parallel. In addition, the paper extends the result to account for any Pareto inefficiency in a two-terminal network. References [25] and [26] extend the characterization from undirected graphs to directed graphs and allow for multiple commodities. Another theoretical study, [27], explores the concept of matroids to identify networks that are immune to the Braess paradox. Unfortunately, few transportation networks exhibit a matroid structure; therefore, the applicability is limited in practice. A further shortcoming of the theoretical studies mentioned above is that they present structures that are immune to the Braess paradox, but do not provide tools to modify existing networks to eliminate this paradox and improve the efficiency of the equilibrium.

The computational study [28] proposes a mathematical programming method to detect the Braess paradox in a given network. The problem is formulated as a bi-level structure and then transformed into a single-level mixed integer program. By setting tolls to links and

analysing the resulting network latency, the algorithm detects links that cause the Braess paradox and penalizes them to imitate road closure.

Instead of links, in our work we search for 'Braess routes' (routes that cause the Braess paradox) with a greedy optimization algorithm and remove them sequentially from the navigation system. This leaves the drivers with a subset of routes that are immune to the paradox, which they are free to choose from. We believe that removing routes is advantageous over removing links, as removing a link adversely affects all other routes going through this link. Among other scenarios, our approach enables removing through-traffic from residential roads, while allowing them to continue to serve the residents. One shortcoming of the route removal approach, however, is that customers might simply switch to another navigation system if their freedom of choice is limited. Further research is needed to address this issue, such as splitting populations into "selfish" and "altruistic" to model different levels of cooperation or creating a system of benefits to encourage drivers' participation.

Unlike other methods for detecting and eliminating the Braess paradox, our model accounts for intersection structures and queues. A more accurate graph representation achieved by fitting functions for link delays and queue delays from data allows us to make theoretical methods applicable to real-world networks. We validate our approach on an extended graph model of a road network from North Bethesda, Montgomery County, Maryland. We slightly modified the geometry of this network with additional links to be able to test a bigger number of OD-pairs and routes. We were able to demonstrate up to 12% delay reduction on this extended network. Despite this improvement, we do not claim the proposed route removal strategy is optimal. Indeed, as shown in [9], the problem of finding the optimal subnetwork is NP-hard. Instead we trade optimality with computational tractability and applicability to real-world networks.

2.2 Network Specification

We consider a network with several routes available to each OD-pair. Every route is represented by a set of consecutive links connecting the origin to the destination.

Link delay function

We estimate the time delay that a vehicle experiences on each link with a Bureau of Public Roads (BPR) function [29]:

$$\Phi(z) = t_0 \left(1 + a \left(\frac{z}{cap}\right)^b\right), \tag{2.1}$$

where

t_0 is the free-flow time ($t_0 = \frac{link-length}{free-flow-speed}$),

a and b are parameters that depend on the link's properties,

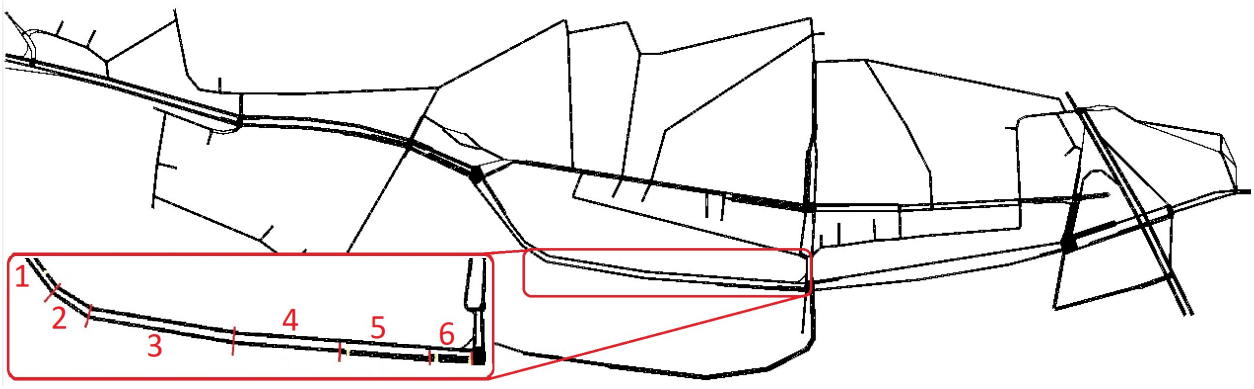


Figure 2.1: Montgomery County network, slightly modified with additional links to test a bigger number of OD-pairs and routes. The red box demonstrates how one edge is represented with a series of links, as discussed in Section 2.6.

cap is the link throughput capacity (in $\frac{veh}{hour}$).

To compute the values of parameters a and b for a particular link, we followed the method described in [30], which requires speed-density data points for a curve-fitting algorithm that outputs appropriate parameters. We used Simulation of Urban Mobility (SUMO) open-source software to build a test case based on the modified North Bethesda, Montgomery County, Maryland network around the intersections of Montrose Rd and Montrose Pkwy (Fig. 2.1). In the absence of historical data, we generated traffic data from the simulation environment and estimated the delay functions. The derivation is detailed in Section 2.6.

Queue delay estimation

Link delay functions do not account for the queues, which accumulate when the link throughput capacity is insufficient for the incoming flow. To address this issue and model queue delay we introduce "phantom" links, i.e. links that have no physical analogue in the real-world or simulation, but exist solely to account for additional delay related to queues. We insert "phantom" links into routes between consecutive edges incoming to intersections and edges leaving the same intersections (Fig. 2.2).

The analysis in Section 2.6 yields queue delay function:

$$\Phi_q(z) = \begin{cases} d_0 & \text{if } z < s \\ \alpha z + \beta & \text{otherwise,} \end{cases} \tag{2.2}$$

where

d_0 is expected constant delay due to intersection structure,

α and β are parameters that depend on the link's properties,

s is the link's saturation rate.

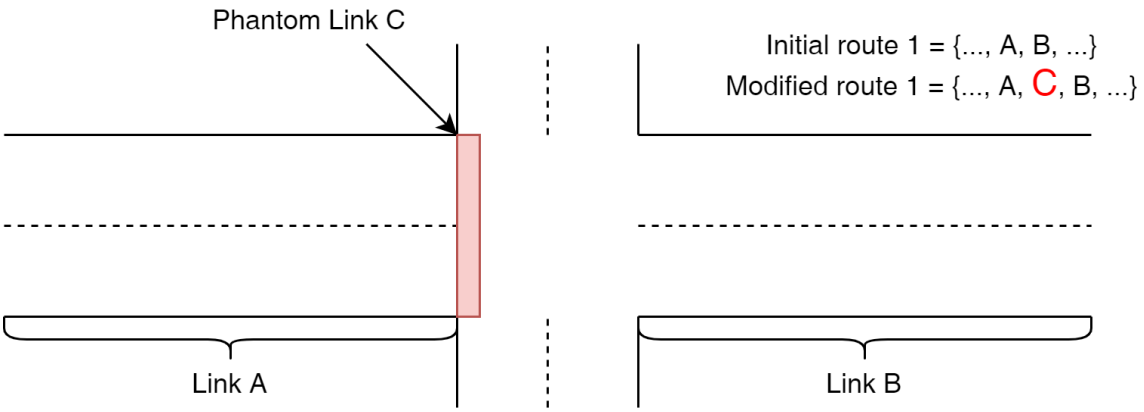


Figure 2.2: A "phantom" link is virtually inserted after the congested edge whose queue-related delay we would like to account for. "Phantom" links have no physical representation and exist only in the mathematical graph model.

The proposed function is a continuous non-decreasing non-negative piece-wise linear function. Therefore, it can be readily used in our route detection algorithm.

2.3 Braess Route Elimination Algorithm

Wardrop Equilibrium computation

To compute the Wardrop Equilibrium state we use a convex optimization problem that utilizes Beckmann potentials [31] for delay functions. The objective function of the optimization problem to minimize is the sum of Beckmann potentials, i.e. integrals of delay functions, across all links:

$$\sum_{i \in L} \phi_i(z_i) \tag{2.3}$$

where

L is the set of links in the network,
ϕ_i is the Beckmann potential of the link i ($\phi_i' = \Phi_i$),
Φ_i is the delay function of the link i,
z_i is the flow on the link i ($z_i = (R^T x)_i$),
x is the vector of route flows,
R is the routing matrix defined as

$$R_{ij} = \begin{cases} 1, \text{ if the route } i \text{ goes through the link } j \\ 0, \text{ otherwise.} \end{cases} \tag{2.4}$$

The first constraint ensures non-negative flows on routes:

$$x_j^k \geq 0, \ \forall j \in P^k; \forall k \in O, \tag{2.5}$$

where

P^k is the set of routes corresponding to the OD-pair k,

O is the set of OD-pairs.

The second constraint guarantees that flows on routes corresponding to the same OD-pair sum up to the demand for that pair:

$$\sum_{j \in P^k} x_j^k = d_k, \ \forall k \in O, \tag{2.6}$$

where d_k is the demand on the OD-pair k.

Therefore, the problem is:

$$\min_x \ (2.3) \text{ subject to } (2.5) - (2.6). \tag{2.7}$$

The solution of the problem (2.7) is the vector x^* of route flows at equilibrium, which results in the total network delay:

$$Y = x^{*T} R \begin{bmatrix} \Phi_1(z_1^*) \\ ... \\ \Phi_m(z_m^*) \end{bmatrix}_{|z_i^* = (R^T x^*)_i} . \tag{2.8}$$

Elimination procedure

The objective of our algorithm is to find subsets of links/routes such that the remaining network has reduced latency. To achieve this we remove links and routes that cause the Braess paradox, which we refer to as 'Braess' links/routes, from the original system. Removing a route implies removing it from the set of options suggested by the navigation system (e.g., Google Maps); removing a link implies either physically closing down the road segment or reducing its capacity with tolls or signaling. As explained in the Introduction, route removal is preferable in practice; however, for the completeness of the study we discuss several approaches for route and link elimination, as they are easily obtained from the equilibrium computation method of the previous section. An important constraint in these approaches is to keep the connectivity of the network unchanged, i.e. every OD-pair from the original network must retain at least one route in the reduced network.

In each approach we attribute values to the links/routes with the help of the optimization problem (2.7). First we obtain the equilibrium network delay, Y, for the original system.

Then, we tentatively remove a link/route from the network (remove corresponding rows and columns from the routing matrix (2.4)) and solve (2.7) again, deriving the new equilibrium network delay, Y_{new}. Having both the original and the new network delays, the value V of the removed link/route is:

$$V = Y_{new} - Y. \tag{2.9}$$

Links/routes with negative values are associated with the Braess paradox, because removing them reduces latency.

Greedy Single Link Removal

1. Find the link with the smallest value V_{min}.
2. If $V_{min} < 0$, remove the link and revert to step (1).
3. Otherwise, terminate.

As stated earlier, removing a link affects the entire set of routes going through this link, which compromises the effectiveness of the algorithm. Moreover, this approach is slower than some algorithms discussed further.

Link Combination Removal

1. Compute the values of all possible combinations of links and find the one with the smallest value V_{min}.
2. If $V_{min} < 0$, remove the corresponding combination.
3. Otherwise, the network is Braess paradox-free.

Since the number of link combinations grows exponentially with the number of links, this method is computationally expensive and does not scale well to large networks.

Link-Route Combination Removal This approach modifies the first method and tries to address the issue with subset of routes elimination. Instead of removing links completely,

1. For each link find the subset of routes utilizing this link that results in the minimal network delay.
2. Remove the routes that were not present in at least one optimal configuration.

Unlike previous algorithms, this method removes routes, but does it indirectly by working with link-route combinations.

Greedy Single Route Removal

1. Find the route with the smallest value V_{min}.
2. If $V_{min} < 0$, remove the route and revert to step (1).
3. Otherwise, terminate.

This approach, unlike the first method, deals with routes directly, which is faster since in practice the number of commonly used routes is smaller than the number of links. An OD-pair can potentially have a large number of possible routes; however the dominant part of the drivers uses the a very limited subset of those routes. It also allows to identify the occurrence of the Braess paradox after the first iteration: if the route with the minimal value has zero initial flow at equilibrium, then the network is Braess paradox-free.

Route Combination Removal

1. Compute the values of all possible combinations of routes and find the one with the smallest value V_{min}.
2. If $V_{min} < 0$, remove the corresponding combination.
3. Otherwise, the network is paradox-free.

This approach is similar to the second approach and has the same limitation due to computational complexity.

All five approaches were tested to identify the fastest and most accurate method for further implementation and verification. The improvements achieved on the sample network were comparable. Therefore, we report below the results for the fourth method (Greedy Single Route Removal) due to its computational speed and advantages in implementation.

2.4 Model Verification via Simulation

Series of simulations were conducted to demonstrate the improvement, evaluate the prediction accuracy and estimate the computational precision of our algorithm when applied to physical networks. The testing was designed to model the behavior of a real-world traffic system, therefore a complex structure of OD-pairs and corresponding routes was built upon (Fig. 2.1). The parameters of the simulation model are presented in Table 2.1.

Table 2.1: Simulation parameters defining network characteristics and demand volume.

Parameters	Values
Number of OD-pairs	12
Number of routes per OD-pair	1-5
Total number of routes	38
Number of simulated vehicles	500 - 3200

A side benefit of the route removal approach is that it may help small side-roads in residential areas that usually suffer from congestion due to drivers attempting to avoid busy freeways. In the era of Google maps, Waze and other navigation systems, which try to

discover short-cuts and guide vehicles through neighborhoods to free up main roads, residents of these regions face busy traffic consequences and spend significantly more time than usual on short trips [32].

The testing procedure consists of the following steps:

1. Solve the problem (2.7) for the original network to obtain the equilibrium flow distribution, x^*, and equilibrium network delay, Y.

2. Derive the route configuration resulting in minimal delay via route elimination algorithm and reduce the network.

3. Solve the problem (2.7) for the reduced network to obtain the new equilibrium flow, x^*_{new}, and network delay, Y_{new}. The theoretical delay reduction is: $I_{th} = \frac{Y - Y_{new}}{Y}$.

4. Feed x^* into the SUMO to obtain the simulation network delay, Y^{sim}, as a sum of travel times of all vehicles.

5. Feed x^*_{new} into the SUMO to obtain the new simulated network delay, Y^{sim}_{new}. The simulation delay reduction is $I_{sim} = \frac{Y^{sim} - Y^{sim}_{new}}{Y^{sim}}$.

6. Compare the theoretical and simulated delay reductions I_{th} and I_{sim}, and estimated total delays Y and Y^{sim}.

Every simulation set was run with different initial demands corresponding to OD-pairs (last row of Table 2.1). Congestion scenarios were of particular interest, because they allowed us to test the developed queue delay estimation model.

It is important to mention that our model does not account for spillbacks (full occupation of a link that causes the queue propagation to the upstream edge). Therefore, upper bounds on demands were applied to avoid spillbacks. Further research is required for an appropriate spillback representation.

2.5 Results

In this section we present the results of our algorithm for several scenarios. The first two rows in Table 2.2 show the size of the simulated traffic. We addressed conditions associated with different times of day.

We are interested in three main metrics when analyzing the efficiency of our approach, which are presented in Table 2.2. The first metric (Row 3) is the theoretical improvement in the network delay, i.e. total travel time saved by all vehicles after implementing our algorithm. In the low demand scenario, the network was Braess paradox-free, and no travel time reduction was achieved. For moderate and congested traffic the improvement ranges from 3.2% to 11.8%, which is reasonable considering the insignificant effort required to modify the network configuration.

Table 2.2: Simulation results for various traffic demands.

	Set-up 1	Set-up 2	Set-up 3	Set-up 4
Demand	Low	Medium	High	High
Number of vehicles	500	1600	2600	3300
Improvement (I_{th})	0%	3.2%	11.8%	8.1%
Improvement Diff.	0%	5.3%	15.3%	10.7%
Network Delay Diff.	1.7%	8.1%	9.5%	9.2%

The second metric (Row 4) shows the difference between the theoretically predicted delay improvement and the corresponding simulation result. For the free-flow set-up there was no Braess paradox and no comparison was needed. For medium and high demands, the difference varies between 5.3% and 15.3%. Taking into account the improvement value itself, we can conclude that the algorithm makes a relatively accurate prediction. Additionally, the theoretical solution always resulted in actual simulation improvement, i.e., $I_{th} > 0 \Rightarrow I_{sim} > 0$.

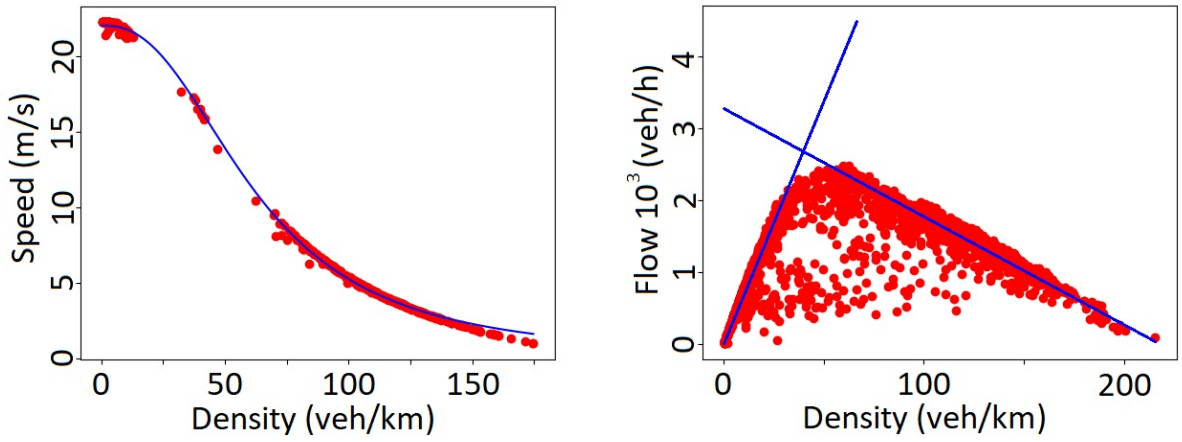
The third metric (Row 5) is the difference between the predicted network delay and the simulated one. According to the results, we managed to keep the deviation under 10% for all scenarios. Furthermore, some cases demonstrated almost identical values for the network delays. The accuracy of the prediction suggests that the proposed graph representation accounting for queue delays yields a reasonable model to reflect real traffic conditions.

2.6 Derivation of Delay Functions

Link Delay Function

In this section we present a detailed link delay function derivation from empirical data points. We treated flow data for each link individually, simulating traffic on one edge at a time. To capture a wide range of flow values, we generated a new random flow from the interval [0,3000] $\frac{veh}{h}$ every 200 seconds. The provided data were sufficient to construct BPR-functions for 90% of links (Fig. 2.3). Tuning simulation parameters covered additional 70% of the remaining links (Fig. 2.4). For the rest of the failed edges, we used parameter values corresponding to delay functions of upstream successful links with minor modifications. We assumed these parameters are likely to have close values and, therefore, can be almost interchangeable. However, if this assumption is false the discrepancy between ground truth delay and estimated delay produced by inaccurate parameter substitution of one link is insignificant in a network scale.

The negligibility of this difference follows from the fact that road links constituting a network are relatively short in general. One edge in the graph network representation is



(a) BPR-function fitting.

(b) Capacity estimation.

Figure 2.3: Examples of successful BPR function construction and road capacity estimation based on the simulated traffic data.

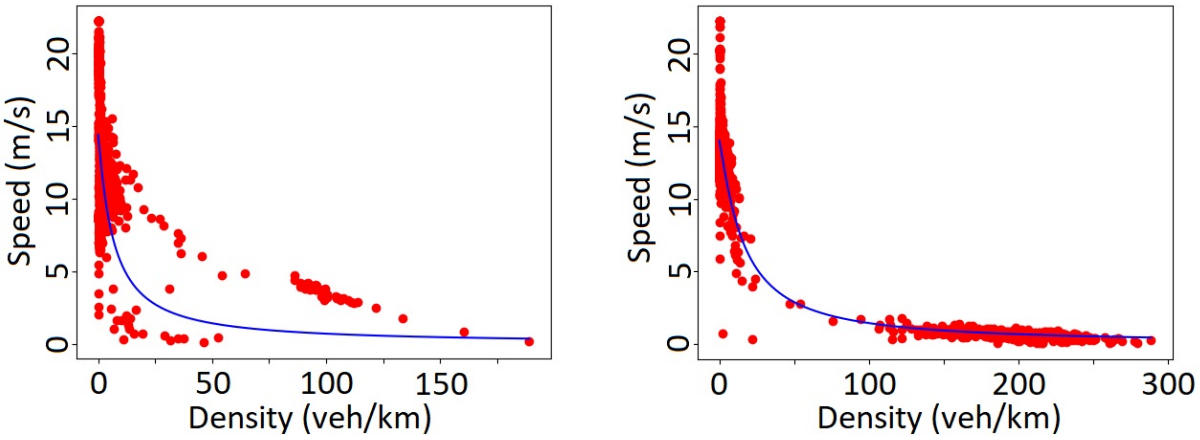


Figure 2.4: Examples of failed BPR function construction due to inability to simulate high-quality traffic data.

usually displayed by a series of short connected links in the SUMO simulation network (Fig. 2.1). The travel time contribution of one link is in the order of few seconds, which is insignificantly small to deviate from the ground truth. Moreover, since the number of poorly-fitted links makes up less than 2% of the number of all links, we can conclude that the proposed design is sufficiently accurate.

Additionally, the link capacities can be extracted from fundamental diagrams built upon

the collected data (in $\frac{veh}{h}$) (Fig. 2.3b). The first method is to set the capacity estimate to the maximal recorded flow value. Another approach is to use a piece-wise-linear approximation to derive the capacity as a y-coordinate of the intersection of two approximation lines.

Queue Delay Function

In this section we present a detailed queue delay function derivation based on the queue formation analysis. Queues occur when the flow (z) on the link exceeds the saturation rate (s) of this link. The saturation rate is the upper bound on the number of vehicles able to leave the link within a period of time. Therefore, the queue formation depends on the difference between the inflow and the outflow on a particular link. Depending on the link type and its relative position with a specific intersection, we can distinguish several possible options for saturation rate estimation:

- **If the link incomes to a signalized intersection**, its corresponding saturation rate equals to the maximum number of vehicles (n) able to pass the intersection on green light within one cycle normalized to one hour:

$$s = \frac{3600n}{D} \tag{2.10}$$

  where, D is the cycle duration (in seconds).

- **If there is a STOP sign at the end of the link**, its corresponding saturation rate is:

$$s = \frac{3600}{w} \tag{2.11}$$

  where, w is the delay (in seconds) a vehicle experiences when forced to stop at the STOP sign on an empty road. This value depends primarily on the speed limit on the link and has a small fluctuation from edge to edge.

- **If the link is free from any traffic regulation causing vehicles to stop or slow down**, the saturation rate equals to the link physical throughput capacity.

- **The link is the secondary link, having no priority on an unsignalized intersection**. Up to this point, the queue delay is given as $\Phi_q(z)$ which is a function of the flow on the corresponding link. In this case, however, the delay incurred by the intersection structure on the secondary link is also a function of the flow of the primary link. Although this violates the fundamental assumption of delay being a function of the corresponding link flow only, if the delay was symmetric for both the primary and secondary links, we could have still implemented this in our optimization problem and computed the equilibrium points. However, primary link sees no delay from the intersection hence the delay function is asymmetric which makes it hard if not impossible to compute equilibrium points the same way. Therefore, this scenario is not represented in our model and is subject to further research.

Similar to the link delay function derivation, we need to use real-world or simulated empirical data to estimate the parameters for queue delay function. We present the detailed function computation procedure for only the first scenario, which features a link incoming to a signalized intersection. Other scenarios utilize a slightly modified approach, which we will mention at the end of this section.

Knowing the traffic light cycle length (D) and keeping in mind the queue size dependency on the inflow-outflow difference, we estimate the one cycle queue growth rate as:

$$dq = \frac{(z - N)D}{3600}, \tag{2.12}$$

where N is the throughput of the link.

Based on the flow value, we can distinguish two scenarios:

- **The flow is smaller than the saturation rate ($z < s$).** In this case, $dq = 0$, all vehicles are able to pass the intersection and no queue is forming. To account for a potential red phase arrival, instead of setting the queue delay to zero, we choose a specific constant d_0, which is the expected value of a delay due to phase change:

$$d_0 = \mathbb{E}(y) = \frac{1}{D}\sum_{i=1}^{L_{red}} i = \frac{L_{red}(1 + L_{red})}{2D}, \tag{2.13}$$

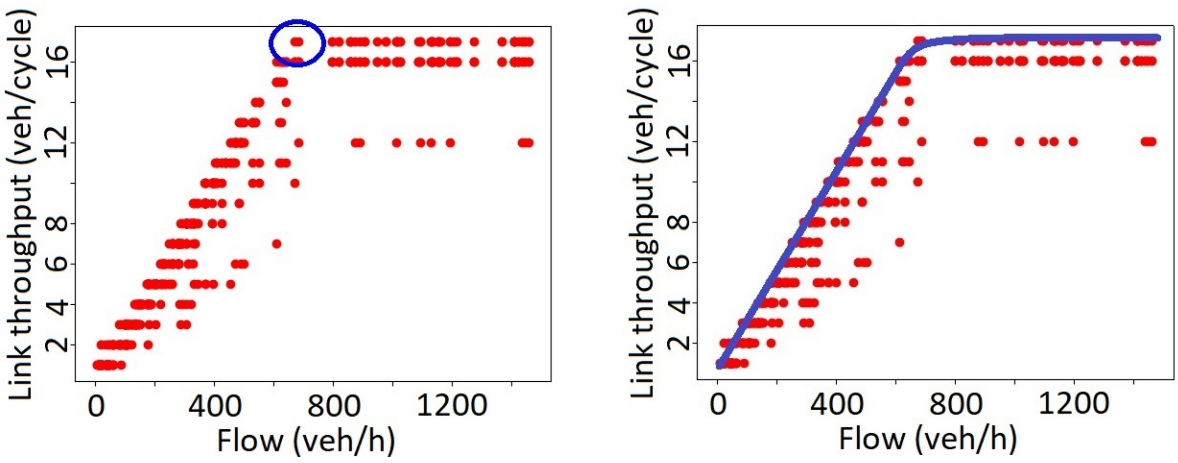
  where L_{red} is the duration of the red phase.

- **The flow is greater than the saturation rate ($z > s$).** In this case, $dq > 0$ and the queue is growing. To find the growth rate, the flow-throughput dependency and, thus, the saturation rate for a particular link are required.

Based on the queue formation model presented earlier, queue delay function depends on the saturation rate of the link. To find the one-cycle throughput capacity of the link we simulate flows of various values in ascending order for a short period of time (200 seconds) each and record the number of vehicles leaving the link (entering the intersection) within one cycle. The flow point at which the throughput linear growth stops (Fig. 2.5a) corresponds to the one-cycle saturation rate of the link. To obtain the value for one-hour period, scale the result by $\frac{3600}{D}$. The throughput of the link (in $\frac{veh}{h}$) is a piece-wise function of the flow which has the form:

$$N = \begin{cases} z & \text{if } z < s \\ \frac{2(z-s)}{z} + s & \text{otherwise.} \end{cases} \tag{2.14}$$

For the flows smaller than the saturation rate, the throughput is a simple linear function, because all vehicles are able to leave the link. Otherwise, the throughput equals to the sum of the saturation rate and an additional term, which never exceeds 2. This term accounts for



(a) Saturation rate.

(b) Throughput function.

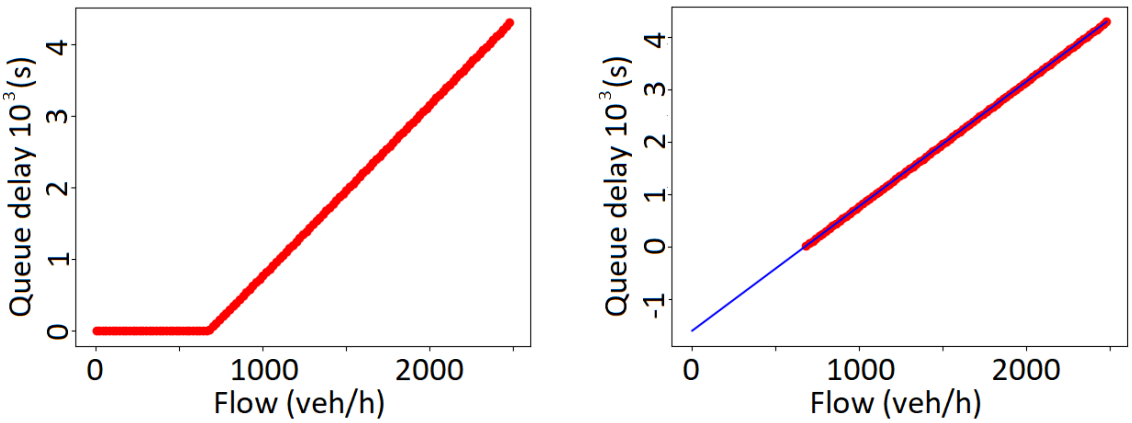
Figure 2.5: Link throughput at a signalized intersection. The link's saturation rate and the throughput function are highlighted blue.

rare scenarios with one or multiple abnormally fast-moving vehicles that manage to exceed the usual saturation rate.

Feeding the derived throughput function back into equation (2.12), we obtain a one-cycle gain in numbers of vehicles to the queue due to excessive flow. It is important to note that the imposed queue delay varies among vehicles within the same flow, and drivers in the head of a heavy traffic would spend significantly less time in the queue, than the ones in the tail. To avoid complications in the optimization problem formulation, an average delay can be assigned to every vehicle on the link. Plotting the computed delay results in a piece-wise linear function (Fig. 2.6a).

The last step is to fit a linear function $\alpha z + \beta$ to the non-constant area of the graph (Fig. 2.6b) and learn the parameters α and β resulting in equation (2.2).

The presented approach can also be used to determine queue delay functions for links ending up with the STOP sign. Both the expected delay d_0 and the cycle length D would be artificially set to w from equation (2.11), since every vehicle without exception is required to stop at the STOP sign, which takes it exactly w seconds to do.



(a) Queue delay data.

(b) Fitting delay function.

Figure 2.6: Signalised intersection flow-delay dependency. The plateau region corresponds to an expected value of a delay due to phase change. After passing the saturation rate value, the function becomes linear.

Chapter 3

Platoon Formation Optimization

3.1 Introduction

Vehicle-formation methods (column, line, echelon, wedge) [33] and vehicle connectivity can drastically improve traffic state. Of particular interest is column platooning, first introduced in [10] by PATH (Partners for Advanced Transit and Highways) for *Intelligent Vehicle Highway System (IVHS)*, which aims to significantly increase road capacity and alleviate congestion by clustering vehicles into *road trains*. It was shown that platooning can increase throughput, improve safety and smooth traffic flow. In [34], the impact of Cooperative Adaptive Cruise Control (CACC) on freeway merge capacity was estimated. The results indicate that the road capacity increases more rapidly as the penetration rate of CACC-equipped vehicles increases. Moreover, it has been shown that platooning can improve fuel economy and decrease emissions due to reduced air drag [35] among other benefits [36].

Platooning consists of three fundamental maneuvers: platoon formation, steady-state cruising and platoon splitting. Various works focus on merging and splitting protocols to establish a more detailed guidance. The study [37] gives an overview of the organization of truck platoons. Vehicle dynamics, longitudinal and lateral control for an *ad-hoc* organization scenario is studied in [38]. Challenges such as nonlinear vehicle dynamics, string-stable operation and merge/split maneuvers in the presence of communication constraints were handled in the design of the longitudinal control policy. The study in [39] presents different control protocols based on Vehicular Ad-hoc Network (VANET). Platoons traveling on a special reserved lane execute merging, splitting and lane-changing maneuvers. Introducing a platoon-dedicated lane improves feasibility and safety, ensures minimal interaction between platoons and the rest of the traffic, allowing them to perform basic maneuvers without disturbing traffic flow or creating hazardous situations.

Another merging algorithm was discussed in [40], which proposes a Hybrid approach for truck platooning alternative to the existing catch-up and slow-down methods. The hybrid platooning strategy combines those two approaches in an optimal manner, forcing downstream and upstream trucks to change their speed trajectories to achieve the fastest merging

time using the maximal platooning speeds of trucks.

A more detailed study of merging protocols is presented in [41], which promotes a distributed decision making algorithm. Another interaction protocol for platoon merging discussed in [42] was used to develop a communication framework for the i-Game project. The work studies two platoons travelling in separate lanes and merging into one convoy due to roadwork ahead. The study [43] focuses on traffic interaction scenarios, such as cut-ins from ordinary cars or slower vehicles blocking the road.

Going beyond low-level control within merging protocols, the local clustering method [44] attempts to match vehicles desiring to merge (VDMs) with existing platoons. Connected vehicles and platoons within a certain range of each other are directed to slow down, speed up or change lanes depending on their current positions. This approach is employed in the SARTRE project [45], which studied platooning in a mixed setting of both heavy and light vehicles. Local coordination was used to match potential participants through a third-party provider. Drivers had an ability to reserve spots in platoons and once in close proximity, the dashboard HMI (Human-Machine Interface) would guide a vehicle to complete the maneuver.

More challenges when implementing the local clustering mechanism were addressed in [44]. According to the study, the biggest difficulty is accurately determining the relative positions of all traffic participants and transferring that information to drivers.

Local clustering method serves as a foundation for many platoon formation procedures. Platoon organization algorithms discussed in [37] and [45] cluster vehicles into groups and order them within each group based on vehicle characteristics (length, weight, size, braking capability, etc.) and drivers' qualifications and preferences. This platoon formation approach works with a mixed set of vehicles and optimizes the drivers' comfort and local fuel consumption. However, the algorithm does not consider the travel times of vehicles and is not aimed to minimize delays. Moreover, placing vehicles in a particular order on highways can be challenging due to surrounding traffic and potential disturbance to the traffic flow.

Some of the attempts in optimizing travel time include works [46] [47], which consider origin-destination (OD) pairs for optimal assignment of vehicles to platoons. Vehicles are clustered to maximize the distance over which the platoon stays intact. Results show an improvement in road utilization at the cost of increased lane changes, which can disturb other traffic participants. Although the travel time in medium and high demand was reduced, the improvement is not significant.

The assignment of vehicles to platoons is still a relatively unexplored concept while it has a high potential to mitigate congestion and increase road utilization. Unlike the algorithms mentioned earlier, our approach aims to minimize travel time. To do so we introduce a double-layered algorithm, where the first layer optimally assigns VDMs to platoons and the second layer chooses one of the vehicles to perform the merging. The first layer is represented by a mixed-integer optimization problem to be solved by individual VDMs. It assigns vehicles to platoons travelling in the dedicated high-occupancy platoon (HOP) lane to minimize local travel time and derives *score* functions (estimated merging time) for each vehicle. Since guiding all vehicles to the HOP lane results in significant flow disturbance and in inability to predict system dynamics, we allow only one vehicle to merge at a time. The

second layer is designed to select the vehicle with the minimal *score* among those competing for the slot to minimize the total travel time of the system. We validated our algorithm via extensive simulation studies and observed that up to 20% travel time reduction can be achieved compared to random assignment procedures.

3.2 Platoon Formation Procedure

The implementation of our algorithm requires a specific highway structure where the left-most lane is dedicated to platoons (similar to carpool lanes). We denote this lane a high-occupancy platoon (HOP) lane and assume that non-platooning vehicles are prevented from travelling in the HOP lane. Restricting platoons to a special lane can potentially allow them to travel faster and consume less fuel (since their motion is not disturbed by the surrounding traffic). This, in turn, would encourage drivers to form platoons and, thus, improve the flow of traffic.

For simplicity, we focus on freeway links that are composed of only two lanes: one regular and one HOP. Scenarios with more than two lanes differ only in lane-switching complexity and traffic interaction density. We further assume that vehicles in the platoon dedicated lane move faster than those in the regular one, providing an incentive for forming platoons of vehicles. Moreover, we ignore the physical length of platoons to simplify calculations. Thus, a single vehicle is treated as a unit mass point and a m-vehicle platoon is treated as a point of mass m. Travel time of such platoon is considered to be mt_{pv}, where t_{pv} is the travel time of a single vehicle in that platoon.

We split the road into segments with finite length (300 meters each) equipped with local centralized controllers, i.e. road-side units, that are responsible for collecting and transmitting data between platoons and VDMs via V2I (vehicle-to-infrastructure) communication, making assignment decisions based on the optimization problem solutions and granting permissions to perform maneuvers. Every platoon in the range of communication (within the zone) reports its speed, position and length (number of vehicles) to the infrastructure. This information is further sent to VDMs. After necessary computations the data is transferred back to the controller and transmitted to platoons. Since the length of every zone is small, communication delays are assumed to be insignificant.

First, we consider one VDM and multiple platoons on a short road segment and then generalize this later to multiple vehicles and multiple platoons.

One VDM, multiple platoons

First, consider a scenario with one VDM and k available platoons (Fig. 3.1). We assume that a vehicle can join a platoon only if the platoon is located behind the vehicle at the moment of decision making. We refer to such platoons as "active" platoons. This is a reasonable assumption since the speed difference between the two lanes makes it impossible for the VDM to catch up with the platoons that have already passed the VDM.

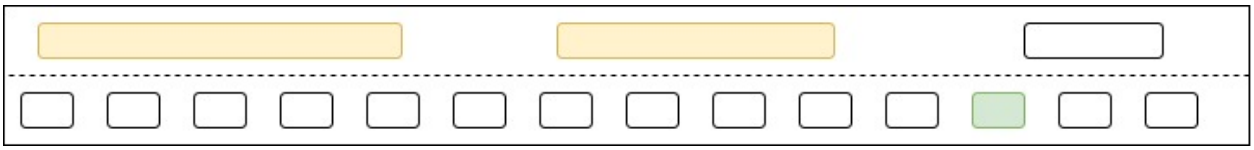


Figure 3.1: The scenario with one VDM and k platoons on the road. A vehicle willing to merge is highlighted green, "active" platoons are highlighted yellow.

After receiving the necessary data (HOP lane speed, positions of "active" platoons and numbers of vehicles in each platoon) the algorithm determines the optimal platoon choice. To do so, we choose a sample distance d long enough to complete all possible lane-switching and merging maneuvers (300m) and precompute the times required for all participants to reach the position $p_v + d$ (where p_v is the current location of the vehicle): t_x^i, t_y^i, t_{z0}^i and t_v for platoon i and the vehicle respectively. The travel time of platoon i when joined from the front is:

$$t_x^i = \begin{cases} T_{FSD}^x, & \text{if } i = 1 \text{ and the platoon slows down} \\ T_{NSD}^x, & \text{if the platoon does not slow down} \\ T_{NFSD}^x, & \text{if } i \neq 1 \text{ and the platoon slows down} \end{cases} \tag{3.1}$$

where

$$T_{FSD}^x = t_{FSD}^m + t_{FSD}^t = \frac{v_p - v_v}{a_v} + \frac{d - \frac{v_p^2 - v_v^2}{2a_v}}{v_p}, \tag{3.2}$$

$$T_{NSD}^x = \frac{d + p_v - p_p^i}{v_p} = t_{NSD}^m + t_{NSD}^t = \tag{3.3}$$

$$t_{NSD}^t + \begin{cases} \frac{2(p_v - p_p^i)}{v_p - v_v}, & \text{if } i = 1 \\ \frac{p_v + p_p^{i-1} - 2p_p^i - d_{st}}{v_p - v_v}, & \text{otherwise.} \end{cases},$$

$$T_{NFSD}^x = t_{NFSD}^m + t_{NFSD}^t = \tag{3.4}$$

$$\frac{p_v - p_p^{i-1} + d_{st}}{v_p - v_v} + \frac{v_p - v_v}{a_v} +$$

$$\frac{d + p_v - p_p^{i-1} - v_p \frac{p_v - p_p^{i-1} + d_{st}}{v_p - v_v} - \frac{v_p^2 - v_v^2}{2a_v} + d_{st}}{v_p}.$$

Each of the colored terms represents merging time. t_{FSD}^m is the time required for the VDM to accelerate from v_v to v_p with acceleration a_v and join the platoon. t_{NSD}^m is the time required for the VDM to let non-optimal platoons pass (in case $i \neq 1$) and accelerate from v_v to v_p to synchronize the velocities with the platoon. t_{NFSD}^m is the time required for the VDM to let the non-optimal platoons pass and accelerate from v_v to v_p with the acceleration a_v. To perform merging maneuvers traffic participants need physical space which is further substituted from the distance d to compute the travel times t_{FSD}^t, t_{NSD}^t and t_{NFSD}^t required to cover the remaining distance with the speed v_p.

The platoon i travel time if joined at the back is ($v_p = v_{sl}$):

$$t_y^i = \frac{p_v - p_p^i + d_{st}}{v_p - v_v} + \frac{v_p - v_v}{a_v} + \frac{d - v_v \frac{p_v - p_p^i + d_{st}}{v_p - v_v} - \frac{v_p^2 - v_v^2}{2a_v}}{v_p} =, \tag{3.5}$$

$$t_y^m + t_y^t = t_{wait}^m + t_{acc}^m + t_{sl}^m + t_y^t.$$

$$t_y^m = \frac{p_v - p_p^i + d_{st}}{v_p - v_v} + \frac{v_p - v_v}{a_v} + \frac{v_p - v_{min}}{2a_p} + \frac{(v_{min} - v_v)^2}{2a_v(v_p - v_{min})} + \frac{d_{st}}{v_p - v_{min}}. \tag{3.6}$$

The merging here consists of three maneuvers: waiting for platoons up to and including the optimal one to pass (t_{wait}^m time required), accelerating from v_v to v_p with acceleration a_v (t_{acc}^m time required) and catching up with the optimal platoon to join it (t_{sl}^m time required). The last term represents platoon dynamics during the catch-up phase. The platoon slows down to the speed v_{min}, which is slightly smaller than v_p, to avoid serious traffic delays, then it cruises at this speed and finally accelerates back to the speed of v_p to synchronize the velocities with the vehicle and complete the merging.

The travel time of platoon i when not joined at all and not affected by platoons downstream is simply a time required to travel the distance $d + p_v - p_p^i$ at the speed v_p:

$$t_{z0}^i = \frac{d + p_v - p_p^i}{v_p}, \tag{3.7}$$

and the travel time of the not-merging vehicle is:

$$t_v = \frac{d}{v_v}, \tag{3.8}$$

where

$\boldsymbol{v_p}$ is the speed of the HOP lane (desired platoons' speed);

$\boldsymbol{v_v}$ is the speed of the regular lane;

$\boldsymbol{v_{SL}}$ is the speed limit on the road;

$\boldsymbol{v_{min}}$ is the minimal comfortable speed that the platoon will slow down to so that another vehicle can catch up;

$\boldsymbol{a_v}$ is the maximum comfortable acceleration of vehicles;

$\boldsymbol{a_p}$ is the maximum deceleration rate of a platoon;

$\boldsymbol{p_v}$ is the current position of the vehicle;

$\boldsymbol{p_p^i}$ is the current position of platoon i.

Our objective is to minimize the total travel time composed of travel times of the VDM and all active platoons:

$$\sum_{i=1}^{k} t_i - (\sum_{i=1}^{k}(x_i + y_i) - 1)t_v \tag{3.9}$$

where

$\boldsymbol{t_i}$ is the decision variable related to the total travel time of platoon i required to reach the position $p_v + d$, considering all possible interactions with other platoons,

$\boldsymbol{x_i, y_i \in \{0, 1\}}$ are binary decision variables, with $x_i = 1$ and $y_i = 1$ representing merging to platoon i from the front or back respectively,

$\boldsymbol{t_v}$ is the non-merging vehicle's travel time given by (3.8).

If one of the x_i's or y_i's (among all i) equals to one, the second term in (3.9) becomes zero and the VDM is accounted for in t_i; otherwise, t_v is added separately.

Since the vehicle can physically join only one platoon and only from one side, the following constraint ensures that at most one of the variables from the set $\{x_i, y_i\}_{i=1}^{u}$ equals to 1 (where u is the number of active platoons):

$$\sum_{i=1}^{k}(x_i + y_i) \leq 1. \tag{3.10}$$

Platoon i is allowed to accommodate at most L_{max}^i vehicles to eliminate the possibility of infinite length platoons. Thus, we impose the constraint:

$$\alpha_i + x_i + y_i \leq L_{max}^i;\ i = 1, \ldots, k, \tag{3.11}$$

where

$\boldsymbol{\alpha_i}$ is the current number of vehicles in platoon i.

The set of safety constraints ensures that the vehicle does not switch lanes if the platoon is too close behind (crash avoidance) by checking if the distance between the platoon and the vehicle in the beginning of the lane switching maneuver is greater than the safe distance:

$$(p_p^{i-1} - p_p^i - d_{st})x_i \geq d_{safe}x_i;\ i = 2, \ldots, k, \tag{3.12}$$

$$(p_p^i - p_p^{i+1} - d_{st})y_i \geq d_{safe}y_i;\ i = 1, \ldots, k-1, \tag{3.13}$$

$$(p_v - p_p^1)x_1 \geq d_{safe}x_1, \tag{3.14}$$

where

$\boldsymbol{p_p^i, p_v}$ are the current positions of platoon i and the VDM respectively,

$\boldsymbol{d_{safe}}$ is the minimal ("safe") distance between the merging vehicle and the closest platoon behind it that is sufficient to perform emergency braking and avoid collision,

$\boldsymbol{d_{st}}$ is the estimated distance between the VDM and the platoon that just passed the VDM, which exists due to the short delay between passing and beginning of lane-switching.

Here we introduce another decision variable t_z, which is a vector of size k. Each entry t_z^i represents the travel time of platoon i, when it is not joined by the VDM, which is different from t_{z0}^i presented earlier. Unlike t_{z0}^i, t_z^i takes into account the impact from downstream platoons and can be increased if the platoon $i-1$ in front slows platoon i down:

$$t_z^1 = t_{z0}^1, \tag{3.15}$$

$$t_z^i \geq t_{z0}^i,\ i = 2, \ldots, k. \tag{3.16}$$

Equation (3.15) has the form of equality because the first active platoon cannot be influenced by platoons in front of it, so t_z^1 equals to t_{z0}^1. Equation (3.16) means that platoon i cannot move faster than when nobody slowed it down.

Moreover, we want to set a lower bound on every t_i according to the pre-computed values of t_x^i, t_y^i and decision variables t_z^i, so that the vehicle and platoon dynamics are taken into

account. The idea is that according to the merging option front, back or not merging (in case of insufficient amount of space, significant delays or exceeding the maximal platoon length), the bound on t_i equals to t_x^i, t_y^i or t_z^i respectively, scaled by number of vehicles in platoon i:

$$(\alpha_i + 1)(x_i t_x^i + y_i t_y^i) - \alpha_i (x_i + y_i - 1) t_z^i \leq t_i, \quad i = 1, \ldots, k. \tag{3.17}$$

Finally, since we do not explicitly compute platoon dynamics, we want to make provisions for collision avoidance when slowing down and accelerating. The following set of constraints ensures that the distance between two tailgating platoons does not drop below d_1 (minimal safe distance). This is enforced by making sure that travel time of platoon i is greater than the travel time of platoon $i-1$ by at least the time required to travel the distance d_1 at speed v_p:

$$-x_i(-\frac{d_1}{v_p} + t_z^{i+1} - t_x^i) \leq 0, \ i = 1, \ldots, k-1 \tag{3.18}$$

$$-y_i(-\frac{d_1}{v_p} + t_z^{i+1} - t_y^i) \leq 0, \ i = 1, \ldots, k-1 \tag{3.19}$$

$$-x_i(-\frac{d_1}{v_p} + t_z^{i+s} - t_z^{i+s-1}) \leq 0, \quad i = 1, \ldots, k-2; \ s = 2, \ldots, k-i \tag{3.20}$$

$$-y_i(-\frac{d_1}{v_p} + t_z^{i+s} - t_z^{i+s-1}) \leq 0, \quad i = 1, \ldots, k-2; \ s = 2, \ldots, k-i \tag{3.21}$$

From before, our decision variables are vectors x, y, t, t_z, but, only the values of x and y matter, since they represent the optimal platoon choice. Moreover, x_i's and y_i's are binary, and thus, the problem can be cast as a mixed-integer program:

$$\min_{x,y,t,t_z} \quad (3.9) \qquad \text{subject to} \quad (3.10) - (3.21) \tag{3.22}$$

In addition to optimal platoon choice, the estimated merging time (t_m) is extracted from (3.2)-(3.6) depending on the problem solution. After receiving the suggested optimal trajectory, the vehicle activates a joining protocol.

Furthermore, we introduce the *score* (S) of the VDM given as a function of several variables: merging time t_m, current waiting time (t_w - the time passed after sending a request to join the HOP lane), etc. For now we consider the *score* to be merging time ($S = t_m$). This value is necessary for complex scenarios with multiple vehicles desiring to platoon.

Multiple VDMs, multiple platoons

So far we considered a single VDM scenario. Now assume there are $k \geq 0$ platoons in the HOP lane and $m \geq 0$ VDMs in the regular lane (Fig. 3.2). Allowing more than one vehicle to merge at a time requires additional constraints and computations to take into account

potential maneuver interference. Every VDM would have to estimate system dynamics and adjust safety constraints depending on different combinations of merging vehicles. Moreover, lane-switching maneuver might be initiated at different times for different vehicles influencing system dynamics and forcing traffic participants to accommodate for the changing system states in real time. Since the number of pre-computed parameters and combined vehicle dynamics grows exponentially, it significantly increases computational time and problem complexity. Thus, we allow only one VDM to merge at a time.

The zone has two flag configurations: "green" and "red". Green flag indicates that all merging and lane-switching maneuvers have been completed and the speed of each platoon has stabilized to a constant velocity. Red flag indicates that some maneuver has begun within the zone and not been completed yet. Our optimization procedure starts when the flag turns green. This is required to determine initial positions of vehicle and platoons and set-up the optimization problem.

Step 1. Each VDM receives traffic data from a local centralized controller (road-side unit) and solves the optimization problem (3.22) independently of the rest of the vehicles. As a result, every VDM obtains an optimal platoon assignment and a score. The score, estimated merging time (if $S \neq t_m$) and optimal platoon choice are reported back to the controller.

Step 2. The algorithm picks the VDM with the minimal score and grants it permission to start the joining maneuver according to the solution from step 1. The optimal platoon to merge into receives the vehicle's ID and estimated merging time from the controller. This information is used to compute the speed profile required for merging. The zone receives the red flag indicating that a merging maneuver is taking place and the rest of the VDMs are forbidden from changing lanes.

Step 3. Once the merging procedure is completed, the infrastructure is informed and the zone switches back to the green flag, meaning that the remaining VDMs have permission to solve their corresponding optimization problems.

Step 4. The procedure terminates if no VDM is remaining and reverts to step 1 otherwise.

We note that our algorithm requires a certain local centralized infrastructure to control the mechanism (gather traffic data, transfer it to VDMs and platoons and make merging decisions). However, the optimization problem is solved in a decentralized manner by the individual vehicles, reducing the overall computational time and taking a significant part of computational effort from the infrastructure.

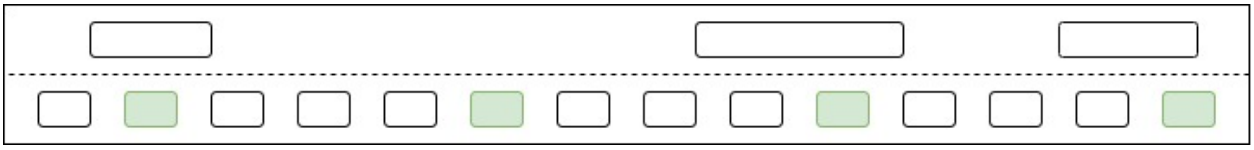


Figure 3.2: The scenario with m VDMs and k platoons on the road. Vehicles willing to merge are highlighted green.

Infrastructure

In this section we generalize the algorithm on a long freeway link. Consideration of all vehicles at once is no longer possible due to large distances between them. Thus, we propose splitting the road into alternating zones: "buffer" and "enabled", 300 meters each (Fig. 3.3). Enabled zones represent road segments discussed in Sections 3.2 and 3.2, i.e. finite-length segments suitable for optimal trajectory computations and platoon assignment. Each enabled zone is equipped with a road side unit.

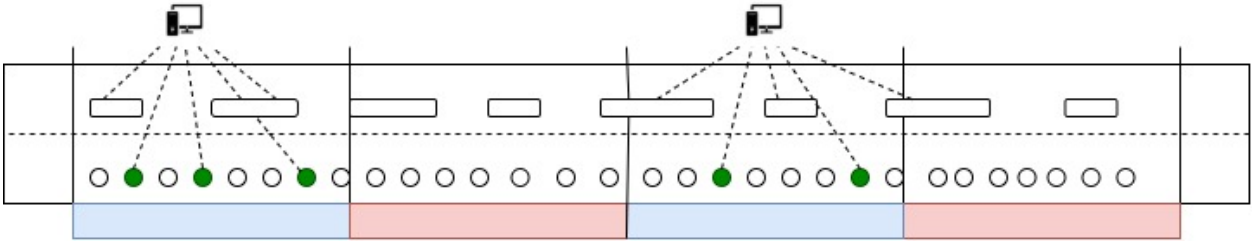


Figure 3.3: Infrastructure diagram. "Buffer" zones, "enabled" zones and VDMs are highlighted red, blue and green respectively.

Since the zones are assumed to be isolated and incapable of communication among themselves, scenarios with two neighboring zones selecting VDMs close to the common edge (Fig. 3.4) are possible. To avoid such scenarios, we introduce buffer zones that disallow merging or lane-switching maneuvers, unless it started in the proceeding enabled zone. This ensures that vehicles simultaneously merging into HOP lane are at least 300 meters apart to prevent any potential interference. This implies that only VDMs in the enabled zones will solve optimization problem (3.22).

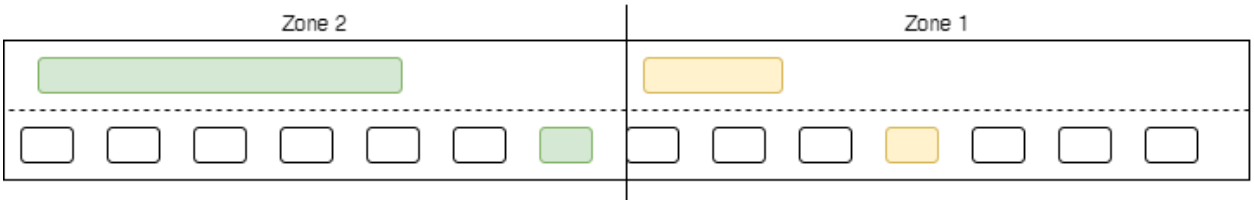


Figure 3.4: The scenario with conflicting merging vehicles. Green merging VDM is too close to the yellow merging VDM, which can cause unpredictable traffic dynamics.

3.3 Simulations and Results

We study the performance of our algorithm in a simulated environment in MATLAB. We use the following setup: the road segment of length 7200m ([-4800; 2400]) is inhabited

with platoons and VDMs (Fig. 3.5). Platoons are distributed along the entire road segment, whereas VDMs are only present along the last 2400 meters of that interval. Since platoons are designed to move faster, this initial configuration ensures that all VDMs have an available platoon to merge into within the road segment.

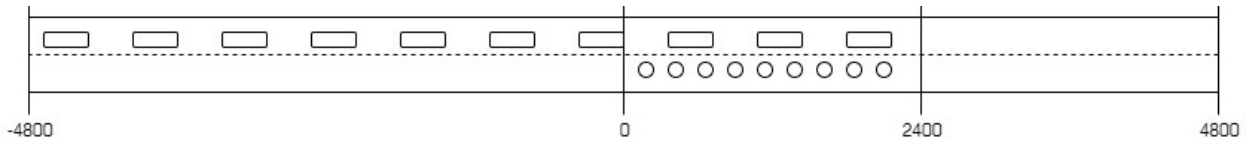


Figure 3.5: Schematic initial configuration of the network. Platoons are scattered within the [-4800, 2400] interval. Ordinary cars are present only in the $[0, 2400]$ region.

The number of vehicles in each platoon is set randomly to a value between 2 and L_{max}. The numbers of simulated VDMs n_{veh} and platoons n_{plat}, maximum acceleration a_v and deceleration a_p, minimal comfortable platoon speed to allow a VDM to catch up v_{min} are chosen to reflect a realistic traffic scenario. The values of the parameters used in simulations are compiled in Table 3.1.

Table 3.1: Simulation parameters

Param.	Value	Param.	Value
d_r	2400 m	n_{veh}	100
L_{max}	5	n_{plat}	120
v_p	20 $\frac{m}{s}$	v_{min}	15 $\frac{m}{s}$
v_v	5 $\frac{m}{s}$	a_v	2.5 $\frac{m}{s^2}$
d_{st}	5 m	a_p	4 $\frac{m}{s^2}$
d_1	5 m	d	300 m

The main objective of the algorithm (not to be confused with the optimization problem objective) is to minimize the total travel time (TTT), which can be estimated from:

$$TTT = \sum_{i=1}^{n_{plat}} t_p^i + \sum_{j=1}^{n_{veh}} t_v^j, \tag{3.23}$$

where

$\boldsymbol{t_p^i}$ is the time that takes platoon i to reach a point of 4800m from $max(p_{init}^i, 0)$ (recall the simulation grid from Fig. 3.5), where p_{init}^i is the initial position of platoon i,

$\boldsymbol{t_v^j}$ is the time that takes VDM j to travel a distance of 2400m.

This calculation method ensures that all vehicles have the same weight in the combined outcome regardless of initial positions. Moreover, the location of the endpoint guarantees enough space to complete merging maneuvers and to stabilize platoons' dynamics.

To evaluate the performance of our algorithm, we consider two baselines. Both of them allow for only one VDM to join at every time instance. We consider the first baseline to be the one that decides on both the VDM and platoons randomly. We call it RTR (Random-To-Random). In the second baseline which we call FTF (First-To-First), we let the first VDM in a zone to join the closest "active" platoon.

The total travel time savings are presented in Table 3.2. First column represents the savings for VDMs only. The second column contains the TTT improvement for only affected simulation participants (platoons that were forced to change their speed and all VDMs). The third column presents TTT reduction for all VDMs and all platoons.

Table 3.2: Travel time reduction comparison for three baseline procedures

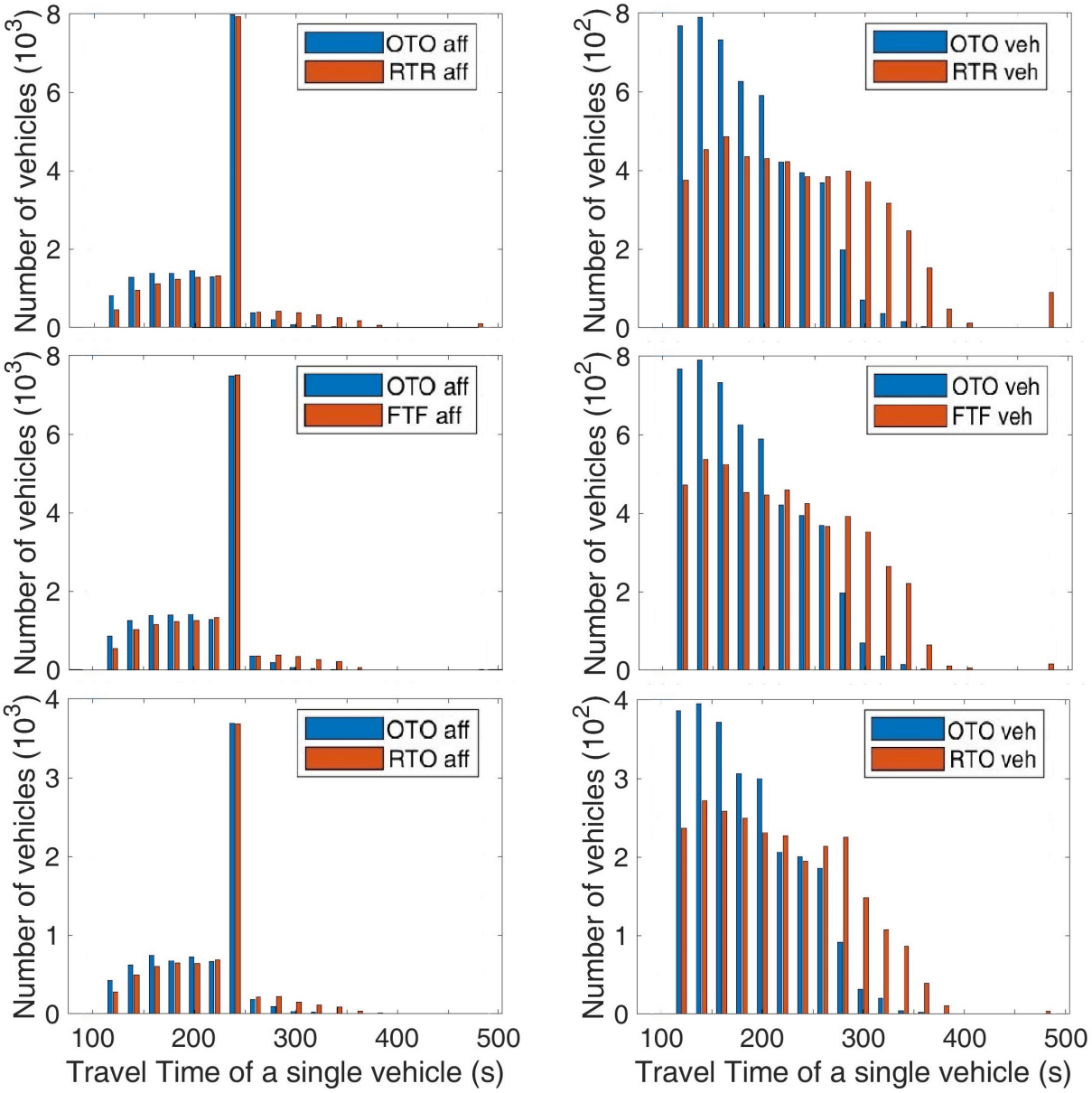
Procedure	VDMs	Affected participants	All participants
RTR	19.6%	6.3%	4.6%
FTF	14.8%	4.8%	3.3%
RTO	14%	4.5%	3.1%

Note that the results in the last two categories may not be representative, because the majority of traffic participants are platooning vehicles, which do not experience significant delays even when slowed down. Thus, the total travel time savings are relatively small compared to the total time spent on the road. However, the trend is obvious: the vehicle-platoon system benefits from the introduction of our optimization algorithm: 4.6% and 6.3% time savings for all and affected participants respectively. On the other hand, the savings for VDMs are much more representative. The double-layered optimization algorithm achieves 19.6% delay reduction compared to RTR. Moreover, our algorithm outperforms the FTF procedure, which will probably be the most common behavior on the road due to its practical heuristic, by 14.8%.

In addition, to understand the significance of each layer, we further compare our algorithm to another baseline, which we call RTO (Random-To-Optimal). This procedure has one layer of optimization, i.e. a vehicle to merge is chosen randomly, however the platoon it should join is assigned based on the optimization problem (3.22) solution. The introduction of optimal platoon assignment leads to 5.6% savings of travel time. Optimal choice of a vehicle to merge decreases the travel time by additional 14%.

To further depict delay reduction we used histograms representing the travel time distribution among traffic participants (Fig. 3.6). Greater bins on the left-hand side represent more fast-travelling vehicles with fewer delays. Fig. 3.6a is related to affected vehicles and platoons and show a slight improvement in travel time. Fig. 3.6b, on the other hand, shows that our algorithm was able to significantly reduce the delay of VDMs, as evidenced by the leftward shift of the distribution of the travel times in the histogram. High peaks around 100-200 indicate that more VDMs joined the platoons within the first few minutes of the simulation period. Moreover, our algorithm was able to accommodate all vehicles, i.e. guide

them to the HOP lane, which is indicated by an absence of blue bins at the right side of the figure: the area to vehicles that had no opportunity to merge.



(a) (b)

Figure 3.6: Travel time distribution for our algorithm (OTO - Optimal-to-Optimal) and three baselines: Random-to-Random (RTR), Firts-to-First (FTF) and Random-to-Optimal (RTO). (a) Affected participants. (b) Vehicles desired to merge.

Chapter 4

Queue Estimation for Speed Advisory and Real-Time Phase Length Prediction

4.1 Introduction

Vehicles equipped with the Speed Advisory System (SAS) [14] use traffic light information and traffic data to obtain the near-optimal speed trajectories to reduce fuel consumption. These trajectories also benefit progression quality by minimizing idling at intersections. One of the key parameters required by the SAS is the estimated remaining time until the end of the current traffic light (TL) phase. In the case of static traffic light (constant phase length) this information can be easily obtained directly from the traffic light via Signal Phase and Timing (SPaT) messages or any other signaling system. However, if an intersection is equipped with an actuated TL, the phase length depends on the traffic demand (vehicle flow) and, therefore, may vary from "minimum duration" to "maximum duration" - specific variables characterizing a particular traffic light. Since the exact phase length value is unavailable, a prediction algorithm is required to provide an estimation to be used in near-optimal speed trajectory derivation.

The software presented in [48] encourages drivers to use smartphone cameras to identify the traffic light color at the upcoming intersection and estimate the remaining time within the current phase. The study reports error rates from 7.8% to 12.4%. The phase length estimation, based on the five previous green-red/red-green transitions is also inefficient: according to the algorithm, the best prediction is just slightly better than estimating the current phase length to be the same as the previous phase length.

A follow-on study [49] addresses the problem of finding optimal speed trajectories to minimize fuel consumption. However, the work considers only pre-timed signals, leaving behind issues with adaptive phase duration. The studies [50] and [51] also focus on pre-timed traffic lights.

References [48] and [52] use noisy measurements of a signal phase to process SPaT estimation. The study analyzes a large number of GPS position and speed samples from 4300 buses within a period of one month to estimate phase duration, cycle length and cycle start time. However, a large percentage of the recorded data was not suitable for the analysis, which limited the accuracy of the algorithm (6s error for 36s phase duration). The later study [53] estimates the waiting time spent by buses in queues and presents significantly better results for the SPaT estimate.

All of the noisy measurement-based algorithms are implemented only for pre-timed traffic lights. In addition, collecting and processing noisy measurements may be computationally inefficient, since most of the signal data are available from transportation authorities.

The study [54] makes probabilistic SPaT predictions based on the intersection traffic data. At the beginning of each cycle, the empirical frequency distribution is computed. Furthermore, for every second within the cycle, the algorithm tries to predict whether a certain phase is **G**(green), **R**(red) or **M**(uncertain) with some level of confidence. Further estimation of the phase residual time requires the knowledge of within-cycle time, which even if available does not guarantee the accuracy of the prediction. In addition to the fact that "80% confidence" predictions may be incorrect, the algorithm does not provide firm guarantees and uncertainty may grow with the increase of confidence level.

An algorithm presented in [55] relies on the historical data from several intersections in Munich and uses a Kalman Filter to predict future probability distributions of phase durations. Although a high level of accuracy was achieved (95%), the practical applicability is limited since the availability level is only 71% on average.

Another study [56] suggests using both the historical phase measurements and the real-time information that locates the current time within the current phase to predict all future phase transitions. Two approaches are considered: "conditional expectation based prediction" and "confidence based prediction". These methods greatly improve the prediction of the residual time for the current phase as well as for the subsequent phase; however, as stated in the paper, the proposed algorithms "pose a challenge to the design of speed profiles that reduce fuel consumption". Since both algorithms update their predictions every second, SAS-equipped vehicles are forced to reevaluate the speed trajectories every second as well, causing jerky motion and fuel consumption increase. Several other papers study vehicle flow estimation at arterial roads using adaptive signal control predictions [57], [58], [59].

Most algorithms discussed above try to estimate or predict the phase length/residual time of the phase using historical data and statistical methods. In contrast, our primary objective is to determine whether or not a vehicle can pass the upcoming intersection during the current green phase. Using real-time traffic data from advanced detectors and the upcoming actuated TL, the algorithm analyzes downstream traffic, estimates the time vehicles can reach the TL and assigns labels ("PASS"/ "WAIT") depending on the vehicle's passing capability. According to the obtained labels, the algorithm derives an estimated phase residual time for near-optimal speed trajectory computation for every participating vehicle.

4.2 Vehicle Labeling Algorithm

We consider traffic lights with fixed cycle length and one actuated axis (later in the work "green / red / yellow phase" refers to the phase of the actuated axis). Therefore, knowing the time within the cycle allows us to compute the precise remaining time until the next cycle (i.e. next green phase, assuming that every cycle starts with the green phase). In other words, for a non-green phase the remaining time until the next green is known and no predictions are required. However, if the current phase is green, the algorithm is used to estimate the phase length.

All the computations and simulations were conducted in an open source simulator SUMO (Simulation of Urban Mobility).

Simple and Complex Network Architecture

The project consists of two parts. First, we analyze a simple symmetric signalized intersection (Fig. 4.1) with East ↔ West actuated axis to test an idealistic set-up and obtain a benchmark for further evaluation. The second part is a simulation of North Bethesda, Montgomery County, Maryland network (around the intersections of Montrose Rd and Montrose Pkwy), a complex system of 9 actuated traffic lights with different geometries and signal schedules (Fig. 4.2). This set-up allows us to test the algorithm in realistic conditions in the presence of various uncertainties.

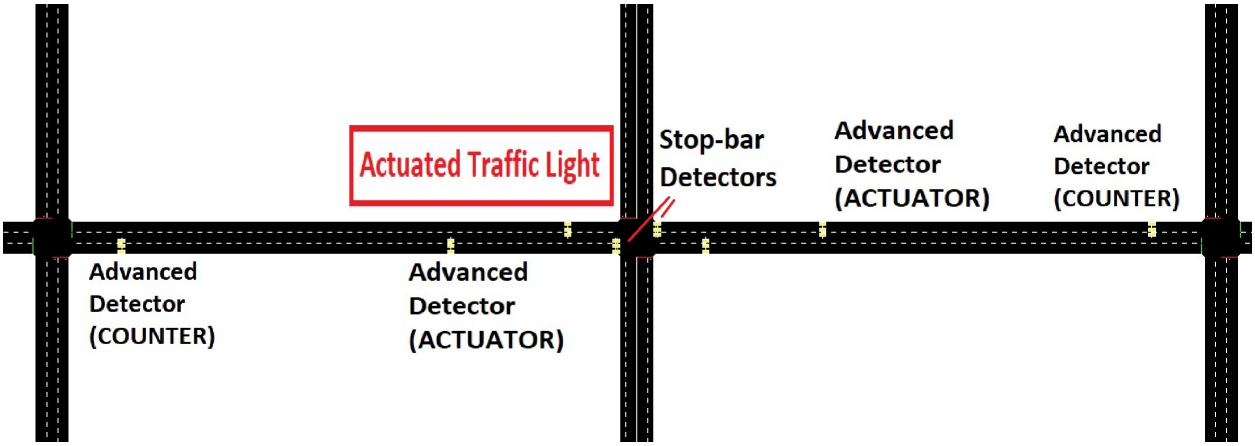


Figure 4.1: Diagram of a simple network used in the first part of the project. The intersection in the middle is equipped with an actuated traffic light.

Every incoming link on the actuated axis is equipped with a stop-bar detector and two advanced detectors: an "actuator" and a "counter". Actuators are responsible for prolonging the green phase when a vehicle is detected. If all necessary conditions are satisfied, a vehicle crossing an actuator triggers the green light extension. Actuators are placed 40-100 meters before the intersection.

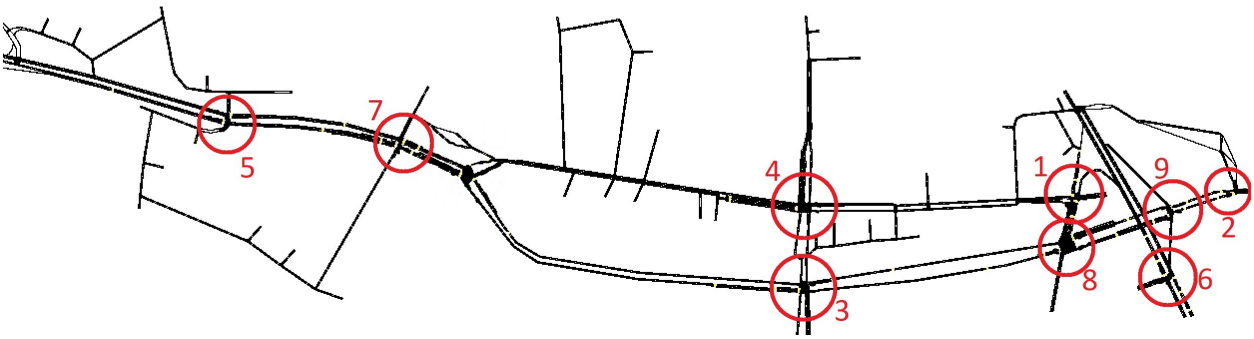


Figure 4.2: Diagram of a complex network based on North Bethesda, Montgomery County, MD. All nine intersections considered in the work are highlighted red.

Counters are advanced detectors that collect necessary information about the traffic state and are located at least 50 meters upstream from actuators. When a vehicle crosses a counter, the infrastructure records and stores the vehicle's speed and time of passing for one cycle. At the end of every cycle, the data are erased, since it is no longer relevant to the current phase actuation. We assume T_{th} seconds is enough to travel from the counter to the intersection in moderate traffic, therefore, counter data corresponding to the previous cycle has no impact on the current one. The case when the vehicle fails to reach the intersection within T_{th} seconds implies congestion that forces the vehicle to switch to the car-following model.

Traffic Light Properties

As stated earlier, the cycle length of the TL j is fixed and equals $cycLen_j \in [90, 120]$ seconds. The cycle consists of 4 phases in the simple case (Tab. 4.1) and up to 8 phases in the complex set-up.

Table 4.1: Traffic light states: groups of three from left to right: North → South, West → East, South → North, East → West; r - red, G - green, y - yellow

Phase	TL States	Min Duration	Max Duration
0	rrrGGGrrrGGG	39 s	48 s
1	rrryyyrrryyy	6 s	6 s
2	GGGrrrGGGrrr	30 s	39 s
3	yyyrrryyyrrr	6 s	6 s

Moreover, we assume that traffic lights are time-gap actuated, i.e. a vehicle can activate it only if the previous actuation was at most *minGap* seconds ago (3 seconds for our simulations) and the maximum phase duration is not exceeded.

Furthermore, the algorithm requires the knowledge of the travel time from the actuator to the corresponding intersection j at speed limit: $T^j_{a-i} = \lceil \frac{D^j_k}{SL_k} \rceil$, where D^j_k is the distance from the actuator on the incoming link k to the intersection j and SL_k is the link's speed limit. Since the duration of the phase must be at least $minDuration^j$, the actuation must be enabled only after the time passes a specific threshold $T^j_{th} = minDuration^j - T^j_{a-i}$. The first vehicle must arrive within $minGap$ after the threshold to prolong the phase. If such a vehicle exists, the next car has $minGap$ seconds to trigger the TL again. The process terminates when either no such vehicle is found or the $maxDuration^j$ is reached.

Speed Advisory System Modification

Since we augment our prediction algorithm with a simplified version of SAS proposed in [14], we briefly summarize the main points of this system. The optimal (in terms of fuel consumption) speed trajectory consists of bang-singular-bang segments: (1) accelerate with maximal acceleration / decelerate with engine off - (2) keep the constant speed - (3) decelerate with engine off / accelerate with maximal acceleration. The singular segment is present only at very low speeds, so most of the time the optimal trajectory is bang-bang shaped. This is both hard to implement in real life and uncomfortable for drivers, so [14] suggests a near-optimal speed trajectory: bang-singular ((1) accelerate with maximal acceleration / decelerate with engine off or with minimal deceleration - (2) keep the constant speed).

We further assume the acceleration and deceleration to be constant. According to the original dynamics, for speeds under $30^m/_s$ the engine-off deceleration ranges from $0.1460^m/_{s^2}$ to $0.1480^m/_{s^2}$. The difference is insignificantly small and, therefore, rounding the deceleration up to $0.15^m/_{s^2}$ would not make any noticeable impact on the system compared to other uncertainties and assumptions. The acceleration also follows an almost linear pattern, so it was decided to set it to a constant value a_{max} ($2.5^m/_{s^2}$ in our model). In addition, it simplified the simulation implementation, since the model presented in SUMO uses constant acceleration.

The resulting near-optimal speed trajectory used in our simulation is one of the following (Fig. 4.3):

1. **accelerate** with a constant maximal acceptable acceleration to a certain desired speed (not exceeding the speed limit) and **cruise**,
2. **decelerate** with an engine off ($\approx 0.15^m/_{s^2}$) to a certain desired speed and **cruise**,
3. **apply** a necessary constant **braking** to meet boundary conditions.

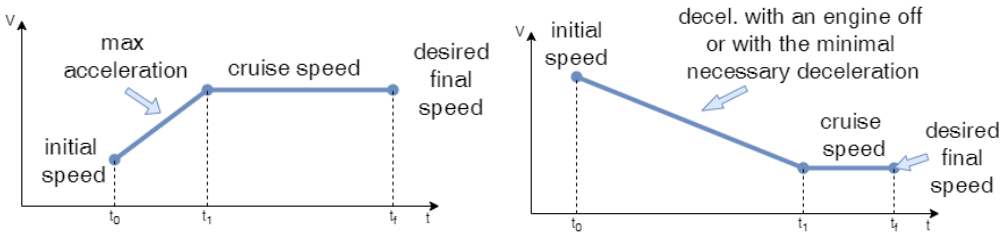


Figure 4.3: Simplified near-optimal speed trajectories implemented in the project. Nonlinear areas corresponding to acceleration and deceleration have been approximated with linear regions. (a) Accelerate-then-cruise case. (b) Glide-then-cruise case.

Algorithm

Estimation of actuation time

the moment a vehicle crosses the counter the algorithm estimates the time when this vehicle is going to reach the downstream actuator and maps it to the time within the current cycle.

Step 1. Compute the time required for the vehicle i to travel from the counter to the actuator on the link k (accelerate with a_{max} to reach the speed limit and cruise):

$$T^i_{travel} = \frac{SL_k - v_i}{a_{max}} + \frac{d_k - \frac{SL_k^2 - v_i^2}{2a_{max}}}{SL_k} \tag{4.1}$$

where v_i is the current speed of the vehicle i and d_k is the distance between the counter and the actuator on the link k.

Step 2. Compute the "counter" crossing time in the traffic light's frame of reference for the vehicle i: $T^i_{count} = T_{current} - T_{start}$, where T_{start} is the time when the current phase started and $T_{current}$ is the current time of the day.

Step 3. Compute the estimated time within the phase when the vehicle i is expected to arrive at the actuator: $T^j_{est} = T^i_{count} + T^i_{travel}$ and store it for one cycle for further computations.

Remark 1: This part of the algorithm is expected to be performed by the *infrastructure*, more specifically, by the computer installed at the intersection.

"PASS" or "WAIT" procedure

After obtaining the estimated arrival time T^k_{est} for the vehicle k, the algorithm proceeds to determine whether or not this vehicle will make it through the intersection within the current phase (Fig. 4.4). To receive the "PASS" label, the vehicle k must either arrive to the actuator before the initial actuation threshold expires ($T^k_{est} < T_{th} + minGap$) or have a "PASS"-labeled vehicle $k-1$ right in front of it and follow it with neither breaking the

minimum gap nor exceeding maximum phase duration. In any other case, the vehicle k receives the "WAIT" label. The "PASS" label implies that the vehicle is expected to be able to pass the intersection within the current green phase. The "WAIT" label, in turn, suggests that the remaining time is insufficient for the vehicle to cross the intersection and advises it to wait for the next green phase.

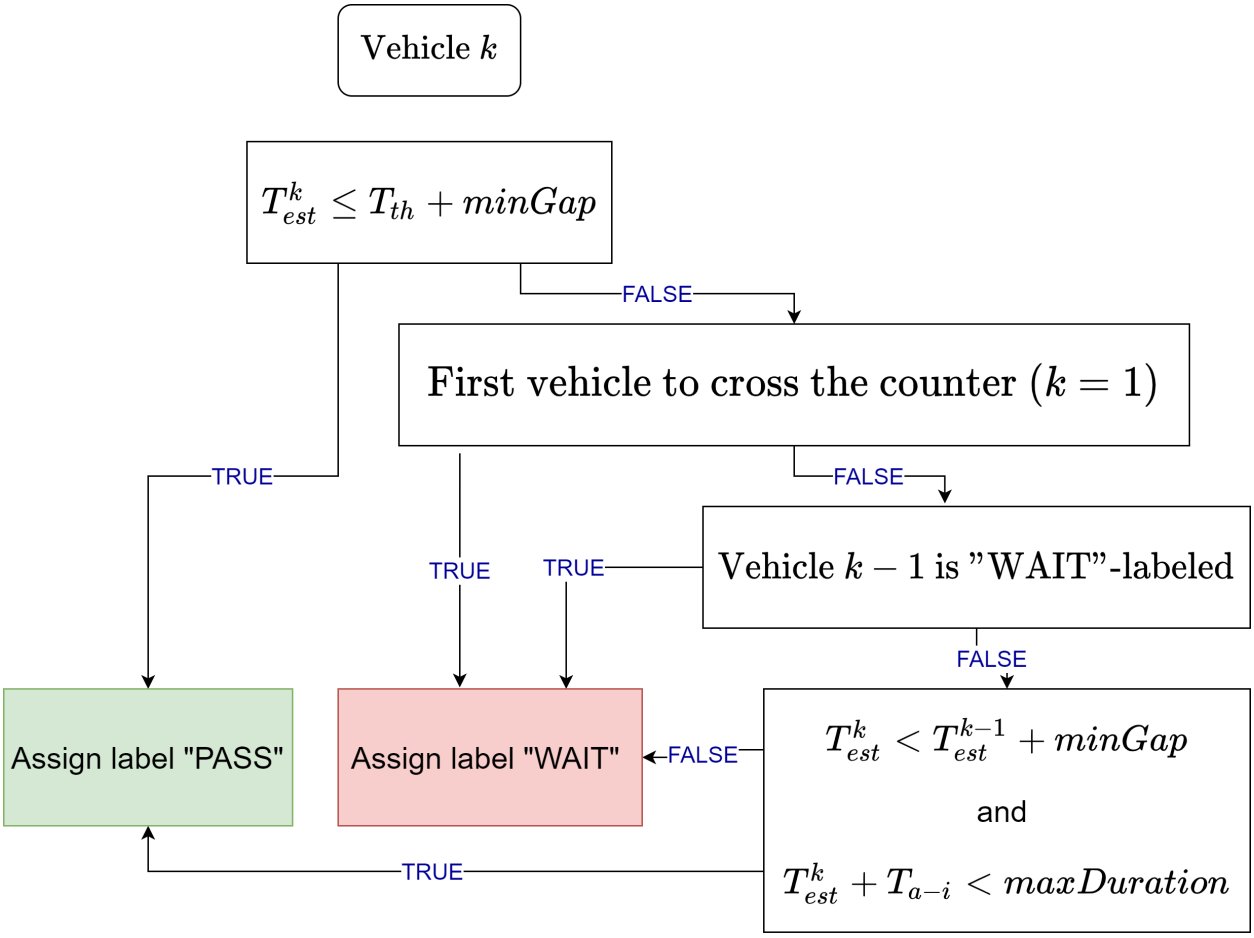


Figure 4.4: Labeling algorithm diagram demonstrating the decision-making procedure based on the counters data and estimated arrival times. "PASS"-labeled vehicles are advised to proceed; "WAIT"-labeled vehicles are advised to slow down.

Complete labeling of all vehicles allows the algorithm to derive an estimation for the current phase length:

$$T^j_{green} = max(T^*_{est} + T^j_{a-i}, minDuration^j) \tag{4.2}$$

where T^*_{est} is the estimated arrival time for the last vehicle with "PASS" label if any.

Remark 2: T^j_{green} is not necessary for the near-optimal speed trajectory computation, but might be important for further development of the algorithm, giving vehicles on the "secondary road" (non-actuated axis) an opportunity to construct their desired trajectories.

Remark 3: The proposed computations can be executed by either the *infrastructure* or *vehicles*. A detailed examination of these options can be found in Section 4.4.

Combining predictions and SAS

At this stage, the near-optimal speed trajectory can be computed using the procedure illustrated in Fig. 4.5. Vehicles labeled "PASS" are advised to proceed as fast as possible to minimize their travel time. On the other hand, being labeled "WAIT" is fundamentally equivalent to not being labeled at all (crossing the counter during a non-green phase). In both cases the residual time until the beginning of the next green phase is $T_{res} = cycLen_j - T^i_{count}$ and can be obtained directly from the infrastructure.

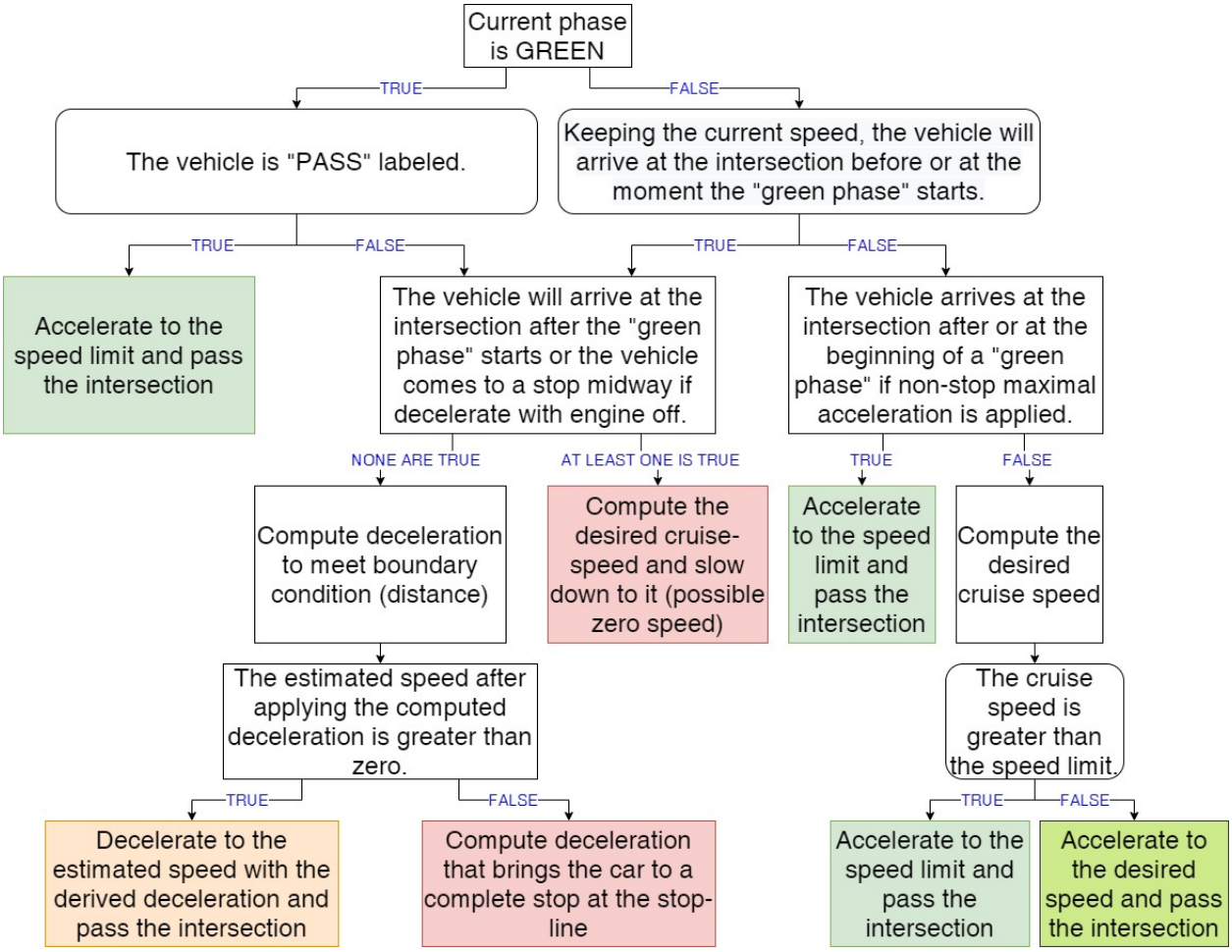


Figure 4.5: Speed Advisory System diagram demonstrating the near-optimal speed trajectory derivation based on the vehicle labeling.

Remark 4: The near-optimal speed trajectory is derived independently by the Speed

Advisory System installed at each participating *vehicle*.

4.3 Algorithm Testing and Verification

The two main objectives of our algorithm are the correct prediction of the vehicles' passing capability and accurate phase residual time estimation for the Speed Advisory System. To test how well our approach meets these objectives, we simulate every vehicle with and without active SAS, and for each intersection on their path we compare the cycle numbers during which both versions of the vehicle crossed that intersection. Moreover, we evaluate the effectiveness of our algorithm by estimating fuel consumption in both cases and deriving the resulting gas savings. Vehicles without any driver-assistance system, which we refer to as "ordinary" vehicles, follow the Krauss car-following model.

Simple case

Simulations

To study various possible scenarios we conducted series of simulations featuring three different traffic demands: low demand ($\frac{1}{40}\frac{veh}{sec}$); medium demand ($\frac{1}{10}\frac{veh}{sec}$) and high demand ($\frac{1}{3}\frac{veh}{sec}$). Moreover, for every demand, different penetration rates of SAS-equipped vehicles were tested: 0%, 20%, 60%, and 100%. Examination of different combinations of penetration rates and demands not only allows us to compare the changes in fuel consumption, but also to study the impact of SAS-equipped vehicles on other traffic participants.

Accuracy of "PASS"-"WAIT" algorithm

Given the required traffic data, Speed Advisory System proposes the near-optimal speed trajectory (Fig. 4.3) that not only reduces fuel consumption but also results in minimal travel time. Therefore, if a vehicle has a chance to cross the intersection within the current phase, the algorithm must not advise it to stop and wait for the next green light. Failure to guide the vehicle through the intersection even though the vehicle is capable of crossing it is referred to as a "mismatch". The simulation results demonstrating the number of mismatches are compiled in Tab. 4.2.

The algorithm demonstrates 100% prediction accuracy in free traffic, allowing all vehicles to pass the intersection within the earliest possible cycle.

For medium demand, we observe rare mismatches, which, however, are not caused by the miscalculation but rather by a model specification. Speed Advisory System treats yellow and red signals identically, disallowing intersection crossing during non-green phases. Ordinary vehicles, on the other hand, do not hesitate passing the intersection on yellow, creating a mismatch in our model.

Table 4.2: Cycle mismatches for various traffic demands

SAS % / Demand ($\frac{veh}{sec}$)	20%	60%	100%	
Low ($\frac{1}{40}$)	58	176	285	# Simulated Cars
	0	0	0	# Mismatch
Medium ($\frac{1}{10}$)	176	565	960	# Simulated Cars
	1	1	3	# Mismatch
High ($\frac{1}{3}$)	400	1251	2054	# Simulated Cars
	3	20	27	# Mismatch

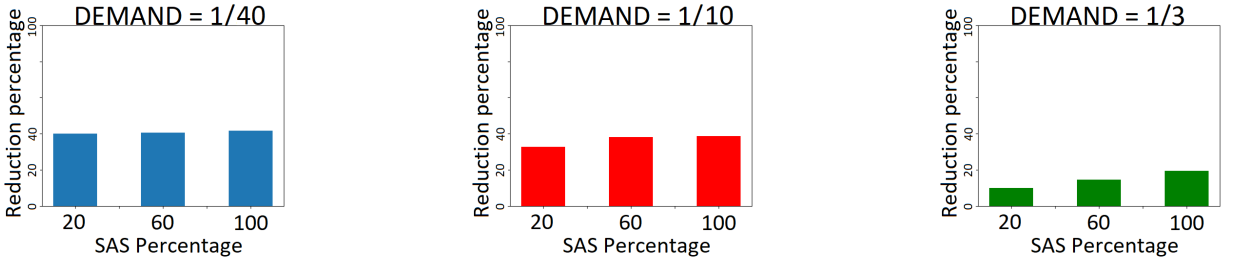
The congested traffic brings more uncertainty to the prediction calculation making it less accurate. As a result, the number of errors increases; however, the accuracy remains significantly high: more than 98%.

Fuel Consumption

The key benefit of the Speed Advisory System is fuel consumption reduction. By following near-optimal speed trajectories, vehicles manage to reduce idling at intersections and increase energy efficiency. Introduction of prediction-based SAS demonstrates a significant improvement in fuel consumption for low and medium traffic demands: 35% - 40% (Fig. 4.6a - 4.6b). According to Fig. 4.6a, the improvement is equivalent for all penetration rates of SAS-equipped vehicles in free traffic. The number of ordinary cars on the road is insufficient to affect speed profiles of controlled vehicles as long as the SAS penetration rate is above 20% for the $\frac{1}{10}\frac{veh}{sec}$ demand scenario. In this case, however, the interference with the SAS-equipped vehicles becomes significant enough to slightly decrease fuel savings, but not enough to drastically deviate controlled vehicles from their desired speed patterns.

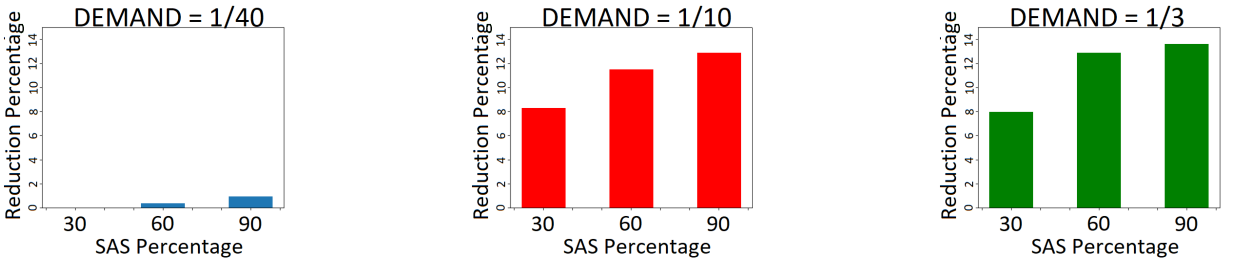
Furthermore, according to Fig. 4.6c, congested traffic neutralizes most of the benefits of the Speed Advisory System. Queues forming due to excessive demand force vehicles to switch from the driver-assistance system to the car-following model, which negatively affects fuel consumption.

In addition, we studied the impact the presence of SAS-equipped vehicles had on ordinary cars. We discovered that traffic participants with no driver-assistance system also manage to reduce fuel consumption, since they are forced to adjust their speed profiles to match the patterns proposed by surrounding controlled vehicles. In free traffic, the change is negligibly small: less than 1% (Fig. 4.7a). However, in mildly and highly congested scenarios, the reduction is quite significant, ranging from 8.3% to 12.5% and from 8% to 13.2% respectively (Fig. 4.7b - 4.7c).



(a) (b) (c)

Figure 4.6: Fuel consumption reduction for SAS-equipped vehicles in mixed traffic. (a) Low demand. (b) Medium demand. (c) High demand



(a) (b) (c)

Figure 4.7: Fuel consumption reduction for ordinary vehicles in mixed traffic. (a) Low demand. (b) Medium demand. (c) High demand.

Phase Utilization

In addition to the primary objectives, several other performance measures were considered. First, we analyzed phase utilization, which can be effectively characterized by phase termination metric [60]. There are four possible reasons for phase termination. An actuated phase can be *omitted* when there is no actuation during the cycle; it can *gap-out* when the TL was actuated at least once and then the actuation gap was broken; and it can *max-out* when the phase duration reaches its maximum allowed length. Max-outs indicate that the phase is exceeding capacity, while gap-outs and omits indicate that there is capacity to spare. Our goal is to study the impact of the Speed Advisory System on the capacity utilization.

According to the simulation results (Fig. 4.8), driver-assistance system leaves the phase utilization unchanged for all levels of traffic demands.

Progression Quality

Another important performance measure is progression quality (PQ), which is strongly related to the queuing delay at an intersection due to arrival-departure patterns: high values of progression quality correspond to low delay. We analyzed Percent-on-green (POG) values:

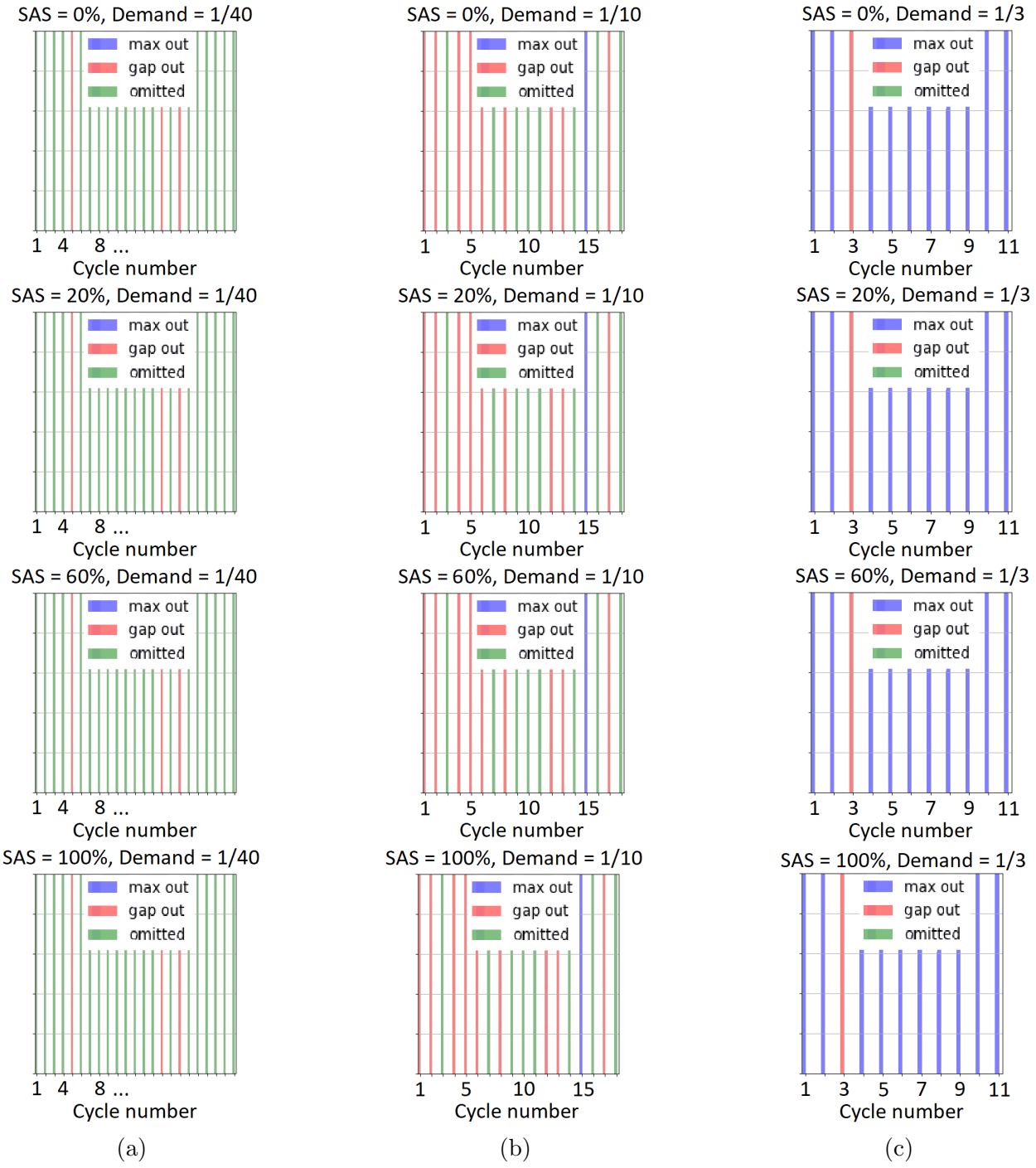


Figure 4.8: Phase termination reasons for various demands and SAS percentages. (a) Low demand. (b) Medium demand. (c) High demand.

$\frac{N_g}{N}$, where N_g is the number of vehicles arriving during red and N is the total number of vehicles arriving withing a cycle, to build an accurate representation of the PQ.

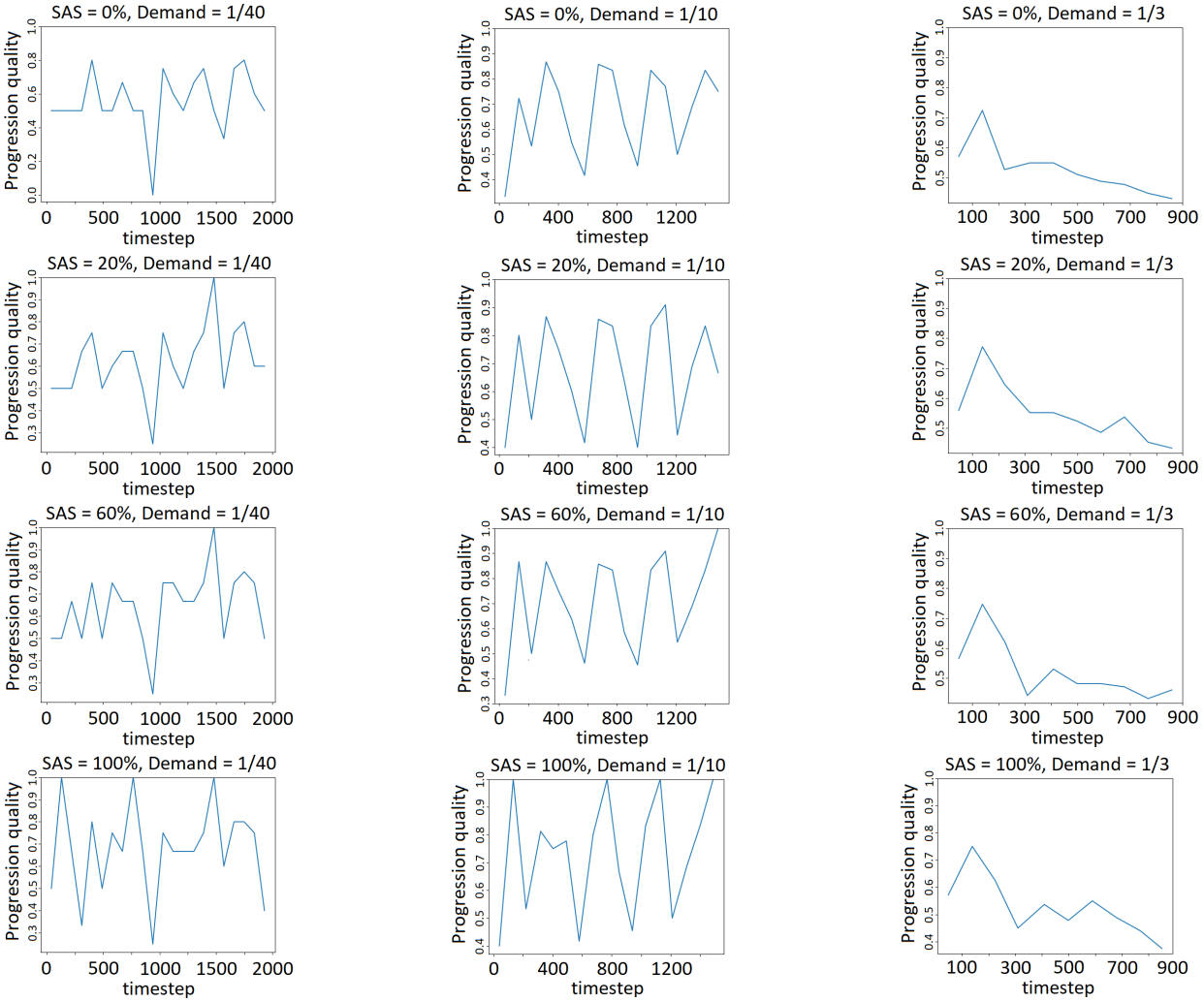


Figure 4.9: Percent on green (POG) for progression quality measure. (a) Low demand. (b) Medium demand. (c) High demand.

According to the low-demand simulation (Fig. 4.9a), we were able to increase the POG to 100%, which corresponds to zero delay, for several cycles with 20% and 100% SAS penetration levels. Average and minimal POG values also benefited from the presence of connected vehicles.

In case of medium traffic demand (Fig. 4.9b), the algorithm demonstrated an improvement only for the 100% SAS penetration rate scenario: three peaks of 100% POG compared

to zero peaks with no SAS introduced. For any other SAS penetration level, the progression quality is at least as good as the case with no SAS-equipped cars.

Finally, congested traffic (Fig. 4.9c), as expected, results in a relatively low progression quality, which can hardly be improved due to constant indissoluble or slowly dissoluble queues. In these conditions, Speed Advisory System is active for a short period of time before the vehicle switches to a car-following model and, therefore, has very limited opportunity to influence the vehicle's behavior.

Montgomery County Network

Simulations

Due to the complex structure of the network it was possible to observe various traffic loads (free traffic, medium demand and congestion) in one setting. We tested three different SAS penetration levels: 0%, 50% and 100%. In addition, three possible options for vehicles' accelerations were implemented to analyze robustness of the algorithm: predetermined and fixed acceleration ($a = 2.5\frac{m}{s^2}$), random but known acceleration and random unknown acceleration (for the last two scenarios a is drawn from a uniform distribution in the interval $[2, 3.5]$). In the latter case the algorithm assumed that all vehicles had an average value of acceleration ($a = 2.75\frac{m}{s^2}$).

Accuracy

The algorithm's prediction accuracy for 9 intersections is demonstrated in Table 4.3. Green rows indicate free traffic, yellow rows correspond to moderate demand, and red rows are related to congestion. According to the data, the algorithm performs with an accuracy of at least 99% in low demand for all possible acceleration scenarios. Having fewer vehicles on the road implies minimal interference, which, in turn, ensures precise calculations.

Medium and high demands demonstrate at least 89% accuracy in the worst case and $\approx 95\%$ on average. Vehicles interacting with each other are forced to slow down, accelerate or stop causing rare miscalculations. Moreover, intersection's geometry can have an impact on accuracy. Intersection 6 has relatively short incoming links, which makes the estimation of travel time difficult due to the discrete nature of the algorithm.

Furthermore, we compare the prediction errors of our algorithm (we will refer to it as "Algorithm A") with those resulting from the statistics-based approach [61], which we refer to as "Algorithm B". This approach is represented by an optimization problem, where phase duration estimation is addressed as a constraint of the form: $c_p^i \geq c_r^i + F^{-1}(\eta)$, where c_p^i is the vehicle passing time in TL i cycle clock, c_r^i is minimal red-phase duration of the TL i, F^{-1} is the inverse of the CDF function F of random variable α representing stochastic time of delay, and η is the required reliability level. The algorithm relies on the assumption that CDF is continuous and bijective. Moreover, the distribution function is non-parametric in general and may vary depending on arbitrary conditions. Although the paper presents impressive

Table 4.3: "PASS" algorithm prediction accuracy

SAS % / Intersection	50%	100%	
Intersection 1	100%	99.4%	Fixed known acc
	100%	99.4%	Random known acc
	100%	99.4%	Random unknown acc
Intersection 2	99.6%	99.8%	Fixed known acc
	99.6%	99.5%	Random known acc
	99.3%	99.6%	Random unknown acc
Intersection 3	98.7%	99.1%	Fixed known acc
	99%	99.2%	Random known acc
	99%	99.2%	Random unknown acc
Intersection 4	96.4%	98.7%	Fixed known acc
	98.3%	97.5%	Random known acc
	98.1%	97.8%	Random unknown acc
Intersection 5	98.5%	98.6%	Fixed known acc
	97.5%	97.5%	Random known acc
	97.7%	97.8%	Random unknown acc
Intersection 6	89.3%	91.3%	Fixed known acc
	91.1%	91.3%	Random known acc
	91.2%	91.3%	Random unknown acc
Intersection 7	93.4%	92.5%	Fixed known acc
	93.2%	94.9%	Random known acc
	94.6%	94.2%	Random unknown acc
Intersection 8	97.9%	99.7%	Fixed known acc
	99.4%	99.7%	Random known acc
	99.5%	99.7%	Random unknown acc
Intersection 9	98%	98.1%	Fixed known acc
	99.3%	99.2%	Random known acc
	98.5%	99.3%	Random unknown acc

results in terms of fuel consumption (50% - 57%), the algorithm can be used only on the secondary road with no actuation capability (Effective Red implies that the perpendicular direction is the actuated one).

Although our algorithm does not provide the actual value of the green phase duration to the Speed Advisory System, we are able to estimate it based on the predicted arrival times, as discussed in Section 4.2. These estimated phase durations were used in comparison with the values provided by the Algorithm B for two confidence levels: $\eta = 0.8$ and $\eta = 0.1$.

Fig. 4.10 - 4.12 demonstrate the difference (in seconds) between the predictions and

the actual registered data for both algorithms. Positive and negative values of these errors correspond to overestimation and underestimation of the phase duration respectively. Each row of every figure corresponds to one of the three tested acceleration scenarios (top to bottom): fixed known accelerations, random known accelerations and random unknown accelerations. We also assume that errors of less than 3 seconds are insignificant due to time discretization and vehicle dynamics simplification.

Fig. 4.10 contains data for the intersection 1 with low traffic demand. All three methods demonstrate high accuracy levels for every simulated scenario. Having fewer vehicles on the road implies low probability of triggering the traffic light actuation, which means that historical phase duration is almost always at minimal duration. Therefore, the CDF used in Algorithm B is almost constant and the prediction is relatively accurate. Although most of the errors can be viewed as insignificant, our method managed to precisely predict the phase duration more often than Algorithm B.

Medium traffic demand is represented by the intersection 5 (Fig. 4.11). According to the histograms, the performance of our algorithm is much better than the performance of Algorithm B. Most of the errors produced by our method do not exceed 2 seconds compared to up to 10-second deviation for the statistics-based approach and occur less frequently. In the cases of random acceleration, greater errors appear, since Algorithm A has to rely on approximations of unavailable parameter values. However, these errors are relatively rare and have an insignificant impact on traffic flow. Regarding Algorithm B, setting the reliability level η to 0.8 results in heavy overestimation for many cycles (up to 8-second errors). On the other hand, with $\eta = 0.1$ we obtain a serious underestimation of the phase length. In moderate traffic phase duration may vary from cycle to cycle with potentially high deviation. One additional data point does not change the CDF function significantly, and therefore, the phase length prediction for cycle k will most likely be very similar to the prediction for cycle $k - 1$ even if the actual phase durations differ dramatically.

Congested traffic data recorded at intersection 7 are compiled into the Fig. 4.12. High demand resulted in max-out termination (reaching the maximum allowed phase length) for most of the green phases during the simulation. The initial CDF for the Algorithm B corresponded to medium demand and required some time to receive enough data points to adjust and provide accurate predictions. Within the first several cycles, we observe poor performance for $\eta = 0.1$ with drastic underestimations up to 10 seconds. On the other hand, Algorithm A and Algorithm B with reliability level at 0.8 demonstrate impressive results with 100% accuracy for most cycles. The terminal stage of the simulation corresponds to congestion dissolution and, as a result, to phase duration reduction. Algorithm B for both reliability levels cannot adapt to the sudden change in demand and continues to output maximum duration as the phase length prediction. Algorithm A, in comparison, manages to reflect the change in the traffic state and correctly estimate the new phase length. Statistics-based approach struggles to demonstrate sufficiently accurate results in the case of changing traffic demands, while the real-time algorithm that relies on the current traffic data is able to quickly adapt and accurately predict the phase length.

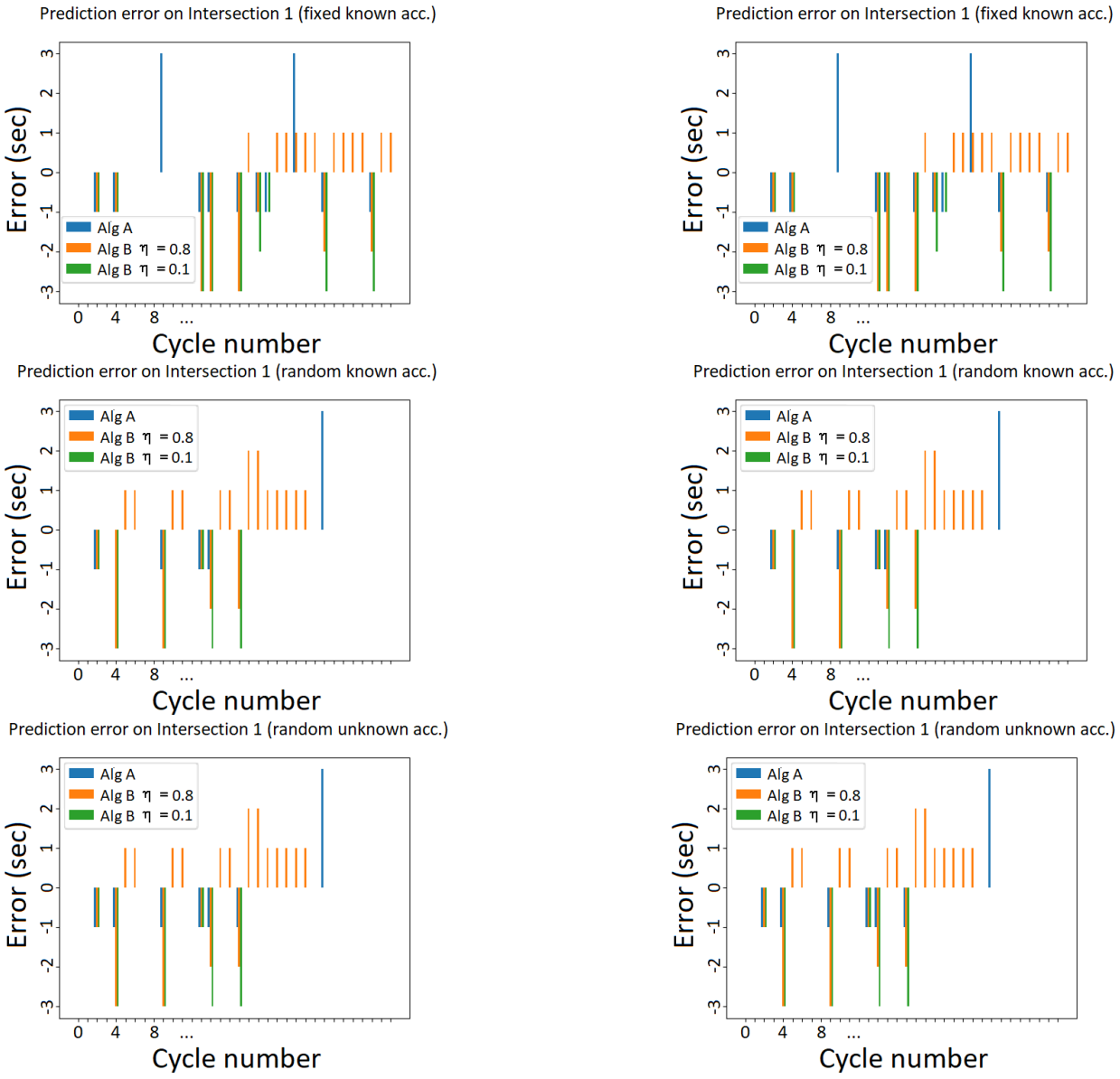


Figure 4.10: Prediction errors in free traffic for Algorithm A and Algorithm B with $\eta = 0.8$ and $\eta = 0.1$. (a) 100% SAS penetration. (b) 50% SAS penetration.

Fuel Consumption

We also compared the impact on fuel consumption for both algorithms (Fig. 4.13). According to the histograms, Algorithm A performs better on average than Algorithm B for both tested reliability levels. Setting η to 0.1 results in much smaller fuel consumption reduction for all intersections and traffic demands. Moreover, choosing the reliability level

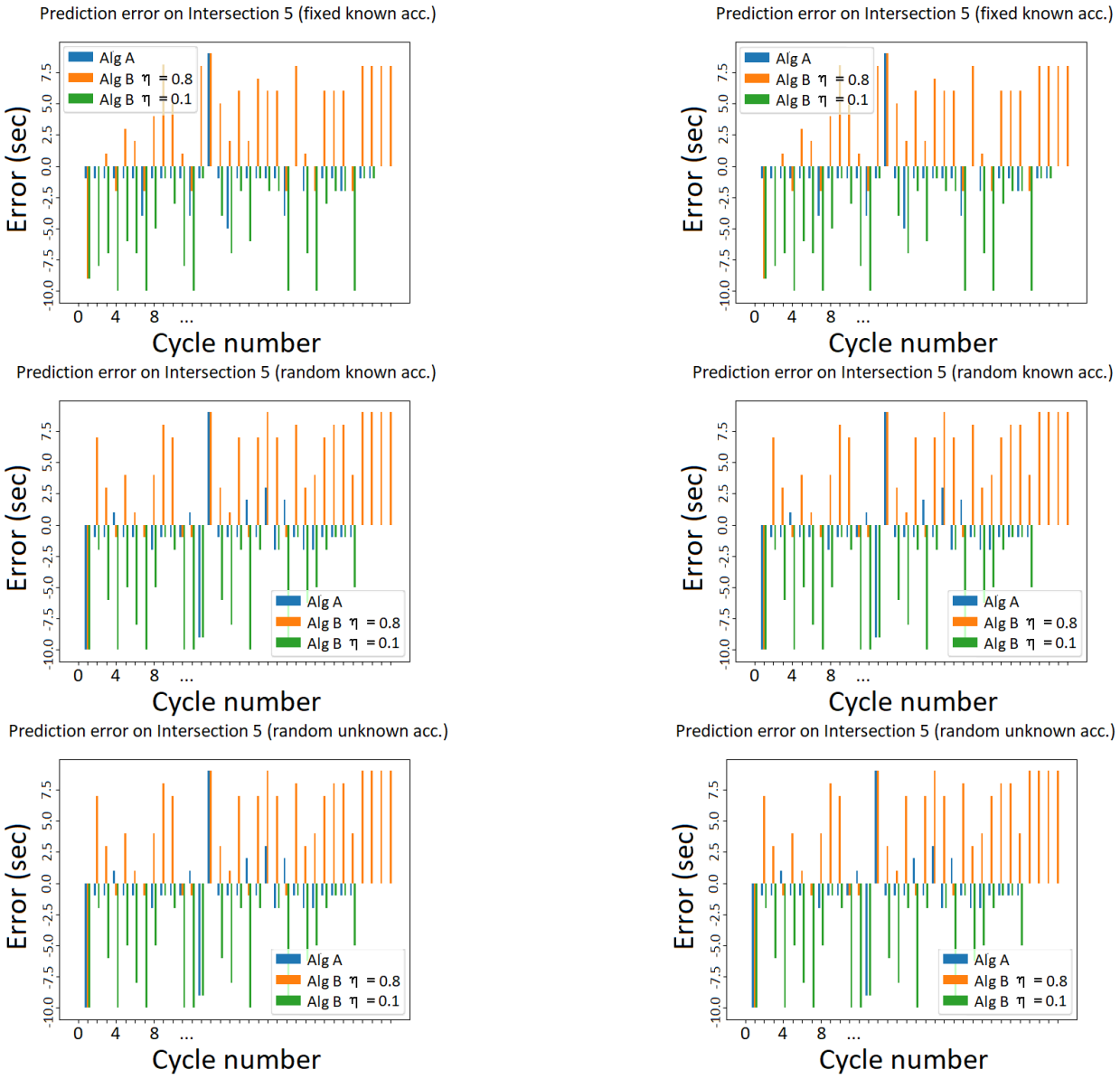


Figure 4.11: Prediction errors in medium demand for Algorithm A and Algorithm B with $\eta = 0.8$ and $\eta = 0.1$. (a) 100% SAS penetration. (b) 50% SAS penetration.

to be 0.8 for Algorithm B results in a similar pattern as applying our algorithm. Switching between scenarios with different acceleration settings and SAS-vehicle penetration rates does not provide a significantly distinct outcome. In rare cases some intersections benefit more from Algorithm B; however the difference in savings is small.

Next step is to analyze our algorithm's performance independently of any other procedure.

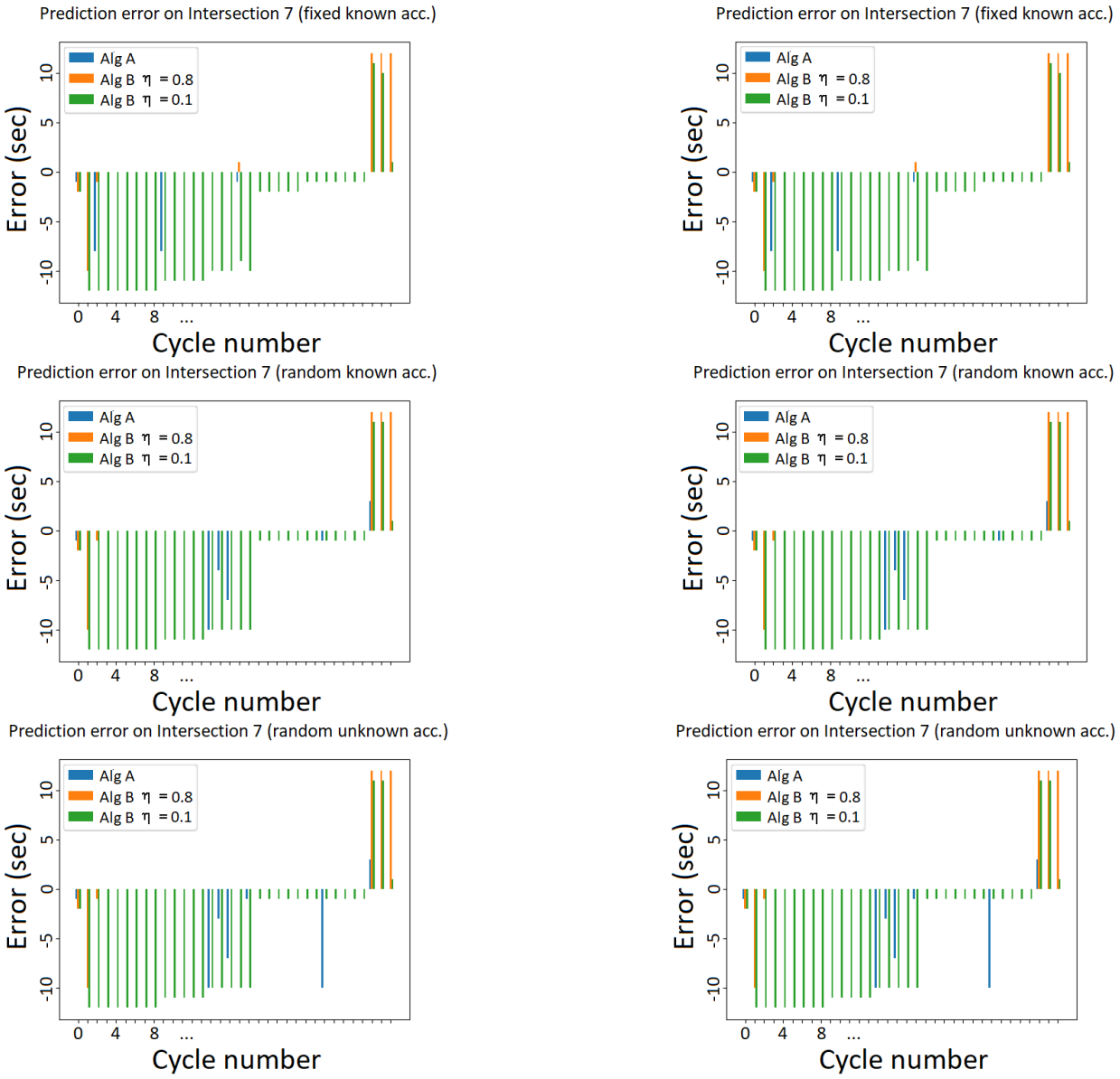


Figure 4.12: Prediction errors in high demand for Algorithm A and Algorithm B with $\eta = 0.8$ and $\eta = 0.1$. (a) 100% SAS penetration. (b) 50% SAS penetration.

We managed to achieve up to 29% fuel consumption reduction for free traffic set-ups. Medium and high demand scenarios also benefit from the presence of real-time prediction algorithm resulting in up to 18% and 7% fuel savings respectively. These results correlate with the simple case simulations. The Speed Advisory System is most effective in terms of fuel consumption in free and moderate traffic, because vehicles are less likely to be distracted from

following the proposed near-optimal trajectories due to low interaction rates. Congestion, on the other hand, forces traffic participants to switch from SAS to car-following model and lose most of the impact driver-assistance could have on energy savings.

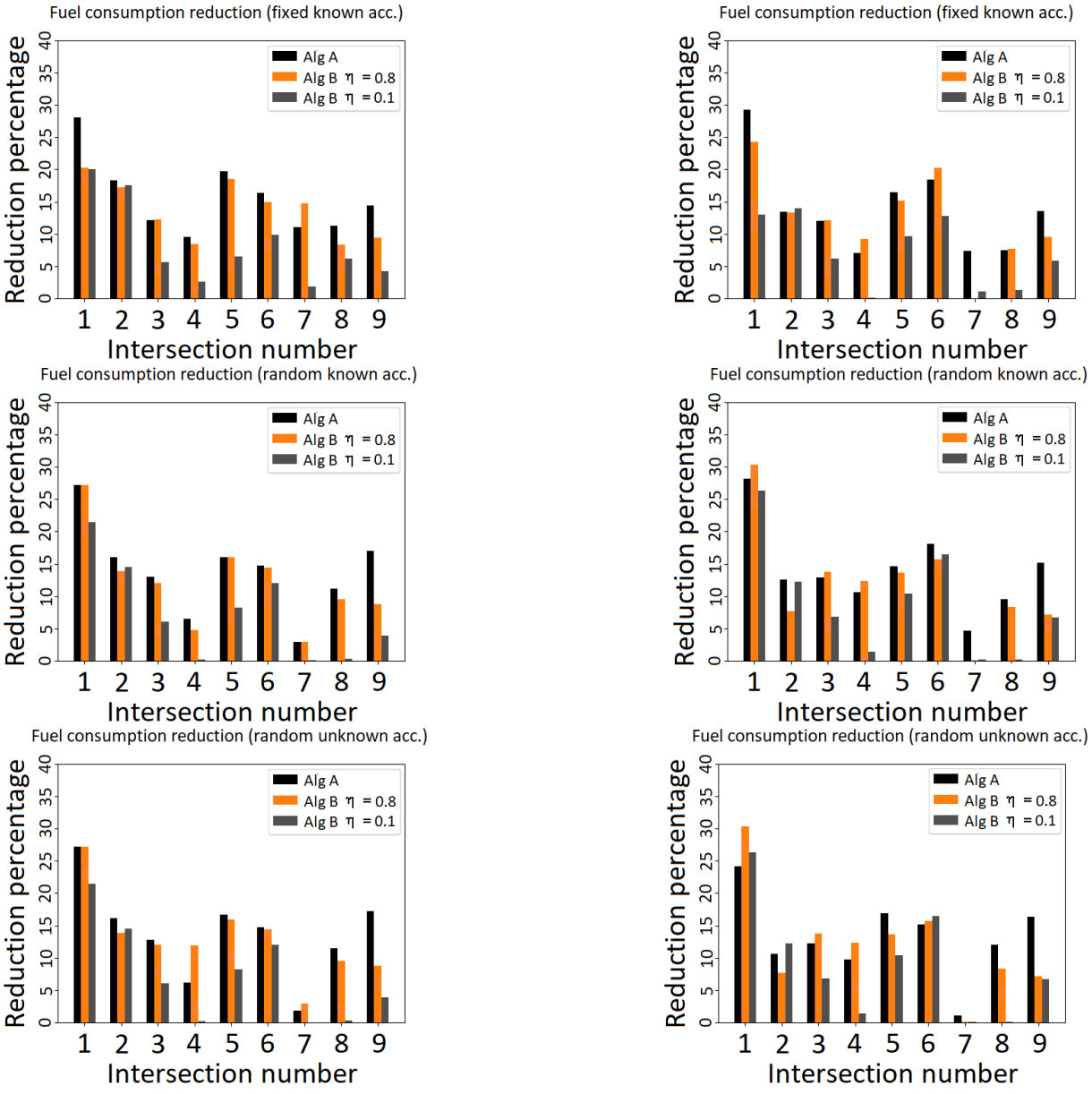


Figure 4.13: Fuel consumption reduction for Algorithm A and Algorithm B with $\eta = 0.8$ and $\eta = 0.1$ in mixed traffic. (a) 100% SAS penetration. (b) 50% SAS penetration.

Phase utilization

We present the results for only known fixed acceleration scenario, since the other setups (known random and unknown random) produce similar outcomes. The data from intersections 1, 5 and 7 are compiled into Fig. 4.14. The first, second and third rows corresponds to 0%, 50% and 100% SAS-equipped vehicle penetration rates respectively.

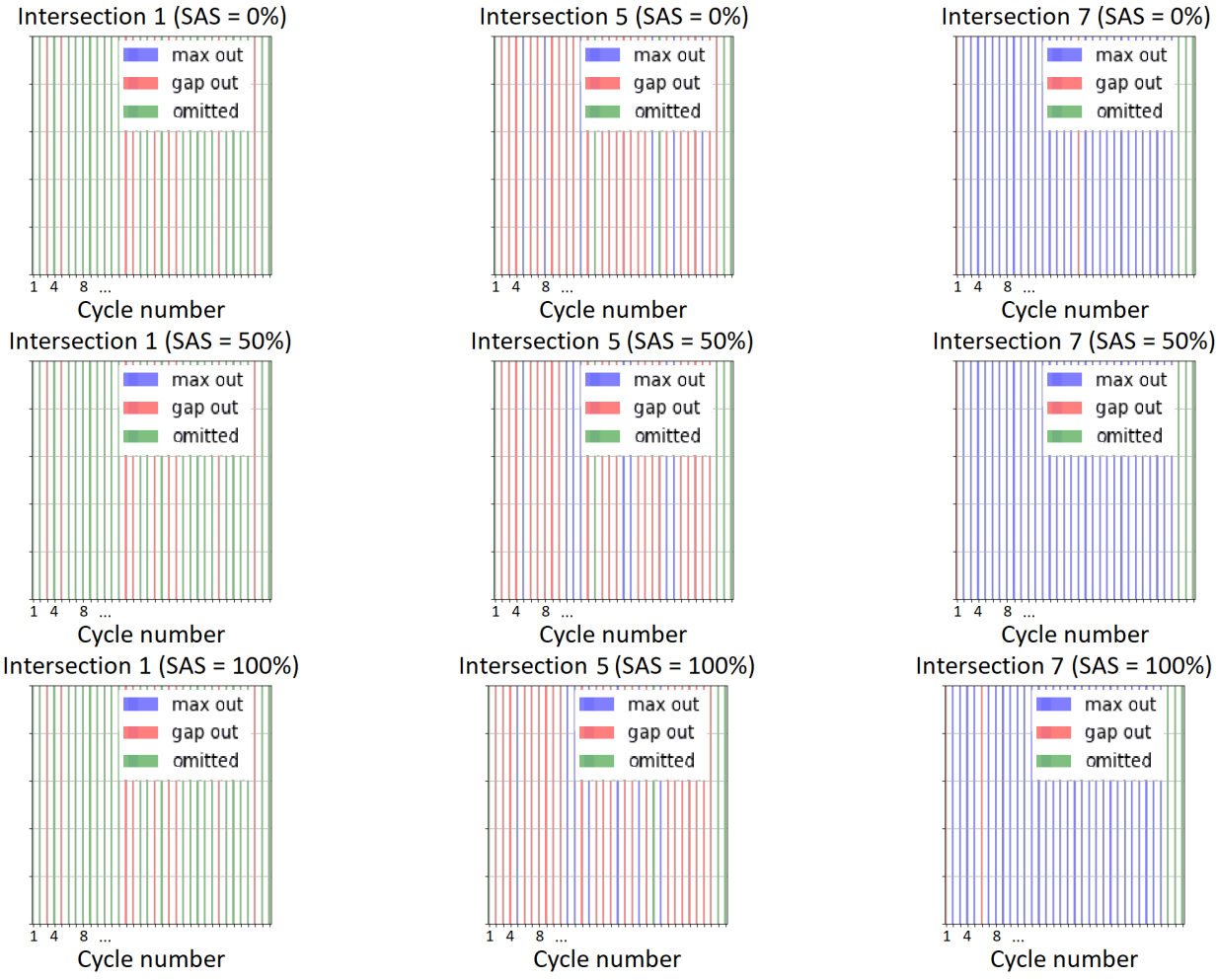


Figure 4.14: Phase utilization for different SAS-equipped vehicle penetration levels. (a) Low demand. (b) Medium demand. (c) High demand.

Similarly to the simple case, phase utilization is barely affected by the introduction of the Speed Advisory System. Therefore, we can conclude that even if the difference is insignificant, the Speed Advisory System does not harm the traffic state of the network.

Progression Quality

Progression quality comparison is presented in Fig. 4.15. Each graph contains outcomes for 0%, 50% and 100% SAS-vehicle penetration. All three tested acceleration options are compiled by rows in the following order (starting from the top): known fixed, known random and unknown random accelerations.

According to the results (Fig. 4.15a), progression quality for low traffic demand slightly benefits from the introduction of SAS-equipped vehicles. Being able to achieve 100% POG more frequently implies that during some cycles we managed to eliminated delays completely. An important detail worth mentioning is that equipping 50% of traffic participants with the Speed Advisory System results in almost the same improvement as making all of the vehicles controlled.

In the case of moderate traffic (Fig. 4.15b), the algorithm achieves more impressive results. Implementation of both 50% and 100% SAS penetration rates resulted in noticeable improvements in progression quality by at least 10% and 8% on average respectively. Moreover, in the case of known fixed accelerations and 100% penetration rate the maximal PQ increase within one cycle was 20%; and in the case of unknown random acceleration and 50% penetration rate the growth reached 33%. These results indicate that real-time prediction was able to provide accurate enough information to significantly improve traffic conditions at intersections in medium demand scenario.

Similarly to low demand, congested traffic (Fig. 4.15c) is not significantly affected by the Speed Advisory System in terms of progression quality. On average, the impact on the metric does not exceed 2%, which correlates with the simple case results.

4.4 Task distribution

It is important to highlight the responsibility distribution for each stage of the algorithm execution.

First of all, as mentioned in the previous section, for each vehicle we need to estimate the time within a cycle when this vehicle arrives at the actuator T^i_{est}. This is expected to be done by the **infrastructure**, since the traffic data get stored at the road-side unit, i.e. computer, and no delay for data transmission is necessary.

The next step is labeling vehicles with "PASS" or "WAIT" labels, which can be done by either the **infrastructure** or the **vehicle** itself. Choosing the first option, we face a challenge of distinguishing vehicles, finding the correct recipient for a particular message and transmitting the information to the vehicle in heavy traffic. On the other hand, by forcing *vehicles* to conduct these calculations, we risk making a mistake in matching received data with the current vehicle's state due to delays and differences in timing. Further research is needed to address these issues.

The final stage of the procedure consists of computing the near-optimal speed trajectory, which is expected to be done by the Speed Advisory System installed at the **vehicle**.

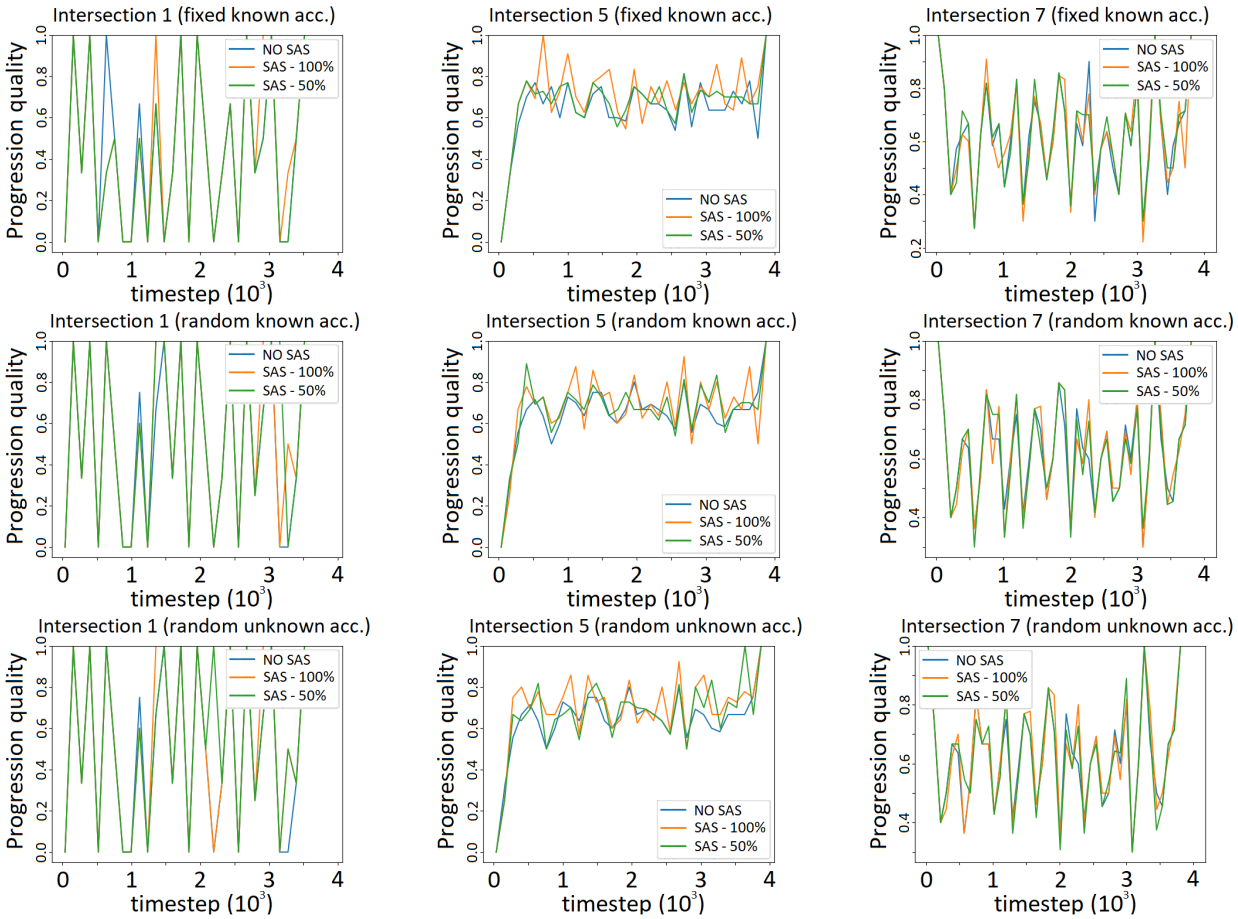


Figure 4.15: Progression quality for different SAS-equipped vehicle penetration levels. (a) Low demand. (b) Medium demand. (c) High demand.

4.5 Extension: Queue Tail Location Estimation

In the previous sections we introduced the vehicle labeling algorithm for real-time phase length prediction and remaining time within the current phase estimation. The derived approximations together with the infrastructure parameters (cycle length, counter-to-actuator and actuator-to-intersection distances, etc.) were shared with connected vehicles enabling further near-optimal speed trajectory computations. The proposed algorithm, however, did not consider queues and their impact on road occupancy and travel delay for the incoming vehicles, resulting in the Speed Advisory System under-performance. The suggested speed profiles, ignoring the presence of any congestion, attempted to utilize the entire length of the road for the required maneuvers, often making the vehicle hit the queue end at a large velocity. Consider the scenario in Fig. 4.16, where a SAS-equipped connected vehicle (red)

approaches a congested intersection with a dynamic traffic light. The Speed Advisory System at its current state proposes the same near-optimal trajectory (green) as if there was a free road up until the intersection. According to the computed speed profile, the red car is supposed to gradually slow down over the entire distance x_{TL} before coming to a stop at the intersection stop line. However, the road segment $x_q - x_{TL}$, in contrast with the assumptions made by SAS, is occupied and cannot be traversed, forcing the vehicle to brake sharply when approaching the end of the queue and switching to a car-following model. Such speed trajectory (red line in Fig. 4.16) is not only uncomfortable for a driver and potentially dangerous in mixed traffic, but can also result in higher fuel consumption.

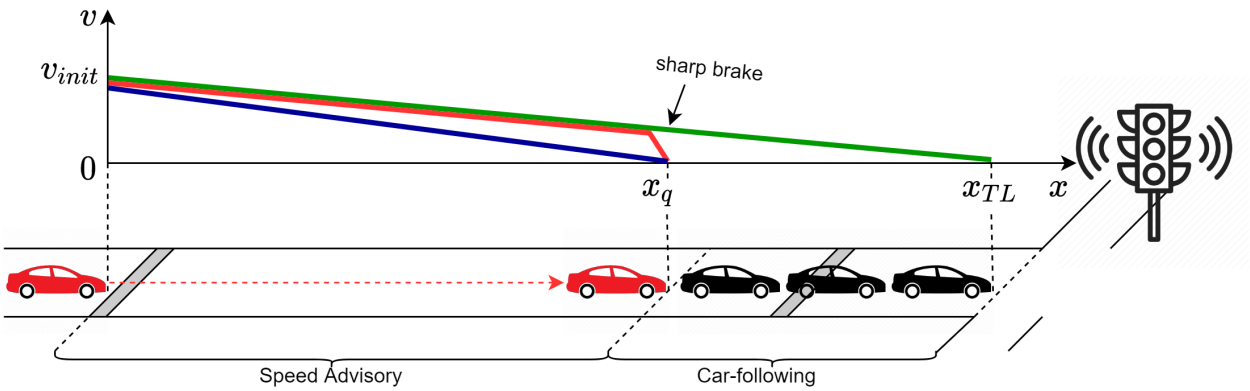


Figure 4.16: Near-optimal speed trajectory inaccuracy due to the presence of queuing at intersection. The predicted speed profile (green) is not feasible. The actual trajectory (red) currently characterized by a sharp breaking must be converged to a more optimal (blue) profile.

To address this issue, in this section we propose a queue estimation model based on the labeling algorithm from Section 4.2, which improves the accuracy of near-optimal speed profile computations in an under-saturated intersection scenario. Adding queue length to the model inputs allows the system to assess the available road space and correctly predict the stopping position of a connected vehicle. Our goal is to make the transition from the speed advisory mode to the car-following model smooth and uninterrupted (blue trajectory in Fig. 4.16), further reducing fuel consumption. One of the advantages of our model is its ability to estimate the future length of the upcoming queue when the vehicle of interest reaches its end. The detailed information about downstream traffic collected via counters and the complete labeling of passing vehicles within the current cycle makes it possible to propagate the state of the queue and predict its development.

The rest of the section is structured as follows. First, we will review the existing research related to queue estimation. Then we will present our algorithm and discuss the statistics-based methods in deriving an expected queue size for a complex intersection structure case.

At the end we will demonstrate the simulation results and compare the proposed model to the original algorithm from [3] with respect to multiple traffic metrics.

Literature Review

A road queue, as demonstrated in Chapter 2, has a significant impact on travel delay and intersection utilization; its length is an important measure of intersection performance. In order to develop effective intersection monitoring and management models, the position of a queue end, among other traffic parameters, must be accurately approximated. Queue size estimation methods can be divided into 2 groups depending on the source of traffic data: vehicle-based and infrastructure-based approaches.

The first group relies on advanced positioning and communication technologies, mobile sensors and their implementation within smartphones and vehicles. Probe vehicles (PV) equipped with advanced sensors are able to build estimated trajectories of other traffic agents for further analysis and state prediction. The estimation models using vehicle-produced data can be further split into two major sets: probabilistic and trajectory-based methods.

The statistics-based work [62] derived analytical models for conditional expectations of a queue length based on the last probe vehicle's position. The later study, [63], in addition to the PVs' locations, also considered their joining times to estimate the total queue length. The work [64] proposed closed-form real-time queue length prediction models assuming Poisson arrivals with known arrival rates. Discrete wavelet transform (DWT), used in [65], is another method in estimating a queue size, which does not rely on any specific arrival patterns and is robust to the probe vehicles' penetration rate variation.

Among the trajectory-based studies is the work [66] that used sample travel times from PVs and intersection delay pattern changes to reconstruct the queue length in real time. The study [67] estimated queue shockwave profiles based on the spatiotemporal probe data and the LWR traffic theory. Another work based on the shockwave analysis, [68], predicted cycle-based end of queue using low-penetration-rate probe trajectories. The method consisted of two steps: estimating an one-cycle arrival rate (traffic state) via solving the maximum likelihood problem with expectation maximum procedure and applying the shockwave theory to derive initial and final queue lengths in cycle.

The second group of estimation models exploits fixed-location sensors and advanced road detectors for traffic data collection. Cumulative input-output models, [69], [70], [71], combined arrival and departure data to analyze the flow state of a single incoming link and estimate the queue length. Another approach, presented in work [72], measured the intersection queue length via queue dissipation modeling in the immediate past cycle using the shockwave theory.

Some works ([73] and [74]), however, did not resort to the vehicle trajectories or the infrastructure-based data alone and successfully incorporated both approaches (fusion algorithm) for an accurate queue estimation. The former study presented an event-based approach utilizing signal timing and probe trajectories for various PV penetration rates. A weighted combination of loop detector data and probe trajectories was introduced as a data

fusion method. The second study proposed a second-by-second traffic state (queue tail location, vehicle accumulation and outflow) estimation model using the input flow from fixed detectors together with the location and speed from connected vehicles. A probability-based approach is used to compensate for low-penetration-rate-related errors.

Both the pure infrastructure-based and the fusion models rely heavily on static advanced detectors, which has a significant downside: potentially high installation and maintenance costs. Data collection by a single detector is limited to a single road link, therefore, to cover a large network, a substantial number of detectors must be installed, which might be costly as a strictly monitoring technology. To lighten the financial burden associated with road sensors, in our work we exploit the already existing advanced detectors (counters, actuators and stop-line detectors) previously used for the phase length prediction and the labeling algorithms. Furthermore, our approach, although adopts the input-output concept of queue modeling, does not simply count the excessive flow, but rather utilizes the labeling system and statistics-based analysis for a queue tail location prediction.

Number of Queuing Vehicles Estimation

The first step in predicting the queue tail location for an individual incoming vehicle is estimating the total number of queuing cars downstream of that particular vehicle. In this section we demonstrate how this can be achieved using traffic data collected from the counters and the complete information about the traffic agents' labeling. As stated in the previous section, our approach is designed for an under-saturated intersection scenario, which is characterized by a full dissipation of the queue accumulated due to the traffic red light in the previous immediate cycle. In this case, two consecutive cycles have non-overlapping queue sets and can be treated as independent and isolated time intervals with their own queue dynamics and traffic agent arrays. The over-saturated case is not considered, since it exhibits the cycle-to-cycle queue propagation, which results in the road blockage preventing new incoming vehicles from crossing the intersection. All vehicles, regardless of the assigned labels, will most certainly hit the end of the queue and switch from the Speed Advisory System to a car-following model. In this case, our current labeling algorithm is unable to provide any substantial benefits for a queue length estimation. Further research is required to develop a more elaborate models for over-saturated intersection scenarios.

Both connected and ordinary vehicles passing over advanced detectors (counters) receive labels "PASS" or "WAIT" according to their ability to go through the intersection within the nearest green phase, as discussed in Section 4.2. A schematic system diagram is presented in Fig. 4.17. Vehicles in the green dynamic region ("PASS"-labeled) comply with the traffic light extension requirements and will almost certainly make it through the intersection without a significant delay in ideal conditions, i.e. conditions defined by the absence of a propagating queue. These vehicles do not contribute to queue formation in the under-saturated scenario and can be ignored in queue length estimation. Therefore, we turn our attention to the vehicles in the red dynamic region ("WAIT"-labeled) and analyze their impact on the queue formation.

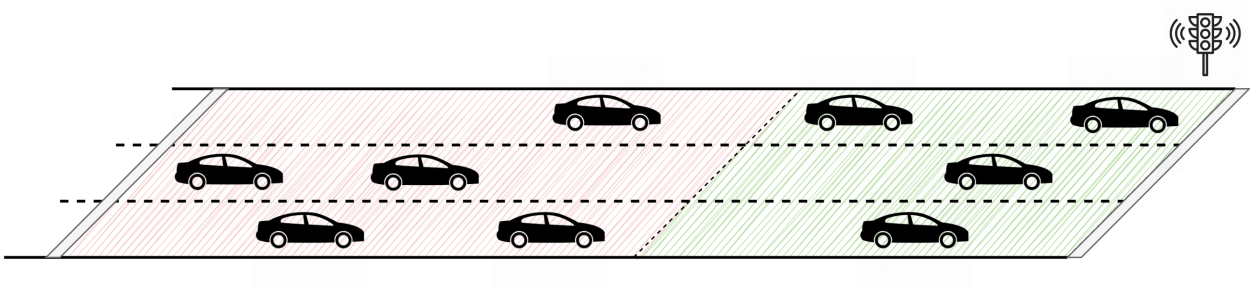


Figure 4.17: Labeling algorithm application to passing traffic in under-saturated intersection scenario. Vehicles in the green zone are labeled "PASS"; vehicles in the red zone are labeled "WAIT".

Let us define W and Q to be the sets of all "WAIT"-labeled and queuing vehicles within the current cycle respectively. Then, *in an under-saturated intersection scenario, an incoming vehicle will inevitably become a queuing vehicle if and only if it is "WAIT"-labeled.* In other words, $W = Q$.

According to the Section 4.3, "WAIT"-labeled vehicles with near 100% certainty will not leave the link within the current green phase due to an actuation time-gap violation or exceeding maximum green light duration. Therefore, while waiting for the next green light to cross the intersection, these vehicles will be forces to join the queue at the end of the road link: $W \subseteq Q$.

On the other hand, "WAIT"-labeled vehicles are essentially the only source of the queue formation for the remainder of the cycle, since the previous cycle queue has successfully dissipated and "PASS"-labeled vehicles from the current cycle, as we demonstrated earlier, have already crossed the intersection. Therefore, the queue consists of only "WAIT"-labeled vehicles from the current cycle: $Q \subseteq W$. Combining these two statements, we observe that $W = Q$.

As a corollary, to estimate the queue end location for each incoming vehicle it is sufficient to analyze the downstream "WAIT"-labeled cars. It is important to note that our queue approximation model is vehicle-dependent, meaning that the queue length estimate provided for a "WAIT"-vehicle k might be different from the estimate shared with a "WAIT"-vehicle $k+1$ behind it. This is due to the fact that the downstream-queuing-car sets Q_k and Q_{k+1} for vehicles k and $k+1$ respectively are not equivalent. In fact, $Q_{k+1} = Q_k \cup \{k\}$, which can cause estimation discrepancies for these two vehicles. Moreover, the queue tail location prediction for each vehicle is computed assuming all its downstream "WAIT'-labeled cars have come to a complete stop. To visualize the statement, consider an example in Fig. 4.18. Assume vehicles $k-2$ and $k-1$ are stopped, while vehicles k and $k+1$ are still moving. Since the first two cars on the link are stationary, the queue end position estimate for the vehicle k is simply the location x_k. The vehicle $k+1$, on the other hand, has one moving car (k) downstream, so the algorithm predicts where the vehicle k will stop and derives the new queue tail location x_{k+1} to be shared with the vehicle $k+1$.

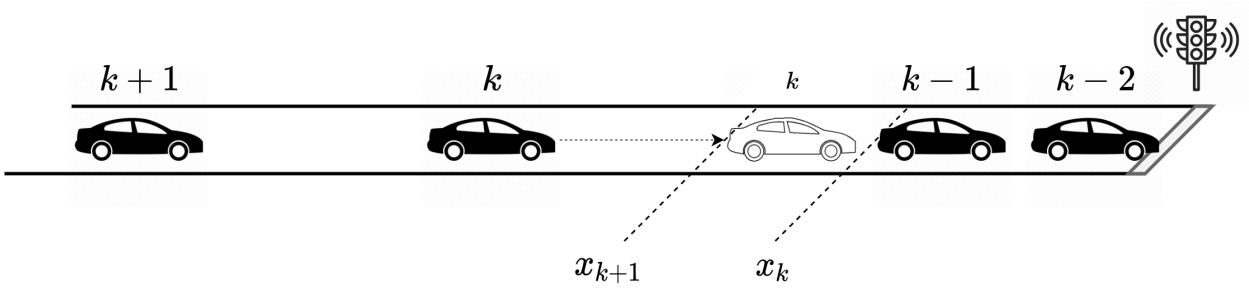


Figure 4.18: Queue tail location estimation for "WAIT"-labeled vehicles. The location is estimated assuming all downstream traffic has come to a stop.

Queue Tail Location Estimation

Our main objective is to derive the queue tail location prediction for each "WAIT"-labeled vehicle approaching the intersection. The number of queuing downstream cars, obtained using the labeling system, cannot be trivially converted into a queue length, considering an m-lane road link. One exception is an one-lane road ($m = 1$), where the queue size simply equals to the number of downstream "WAIT"-labeled cars. For $m > 1$, due to a potentially non-uniform distribution of queuing vehicles among lanes, a simple mean value computation is insufficient for an accurate queue estimation. In this section we propose two statistics-based approaches for deriving a queue tail location using the historical traffic data and vehicle routing systems.

Each road link incoming to an intersection has a finite number $n \geq 1$ of possible exiting direction (Fig. 4.19) depending on the intersection geometry, traffic laws and number of lanes. Some directions are uniquely mapped to single lanes (e.g. left-turn lanes); the others can be allowed across multiple lanes. Similarly, a lane can either be reserved for a particular direction or allow for multiple direction at the same time. We define D_i as a set of exiting direction allowed from the lane i. Since the general vehicles' moving patterns utilize some exiting directions more often than the others, deriving turning ratio (T_R) distributions for road links is essential for approximating lane occupancy and estimating the lane queue size. A turning ratio of an exiting direction is defined as the probability of a vehicle leaving the link to take this particular direction; the TR can be derived using historical data: road sensor measurements, navigation system data, surveillance camera footage, etc.

From the number of downstream queuing vehicles and the turning ratio distribution of a link it is possible to estimate lane-related queue sizes at the intersection. Consider an m-lane road link allowing for n possible exiting direction with corresponding turning ratios $\{T_R^1, \dots T_R^n\}$. We assume that the lanes assigned with the equal sets of exiting directions (e.g. 2 lanes with only the "straight" direction) have an equal probability of being utilized, therefore, resulting in the equivalent expected queue size. Moreover, we assume that a vehicle's lane choice is independent of the current queue size on the lane, resulting in cumulative

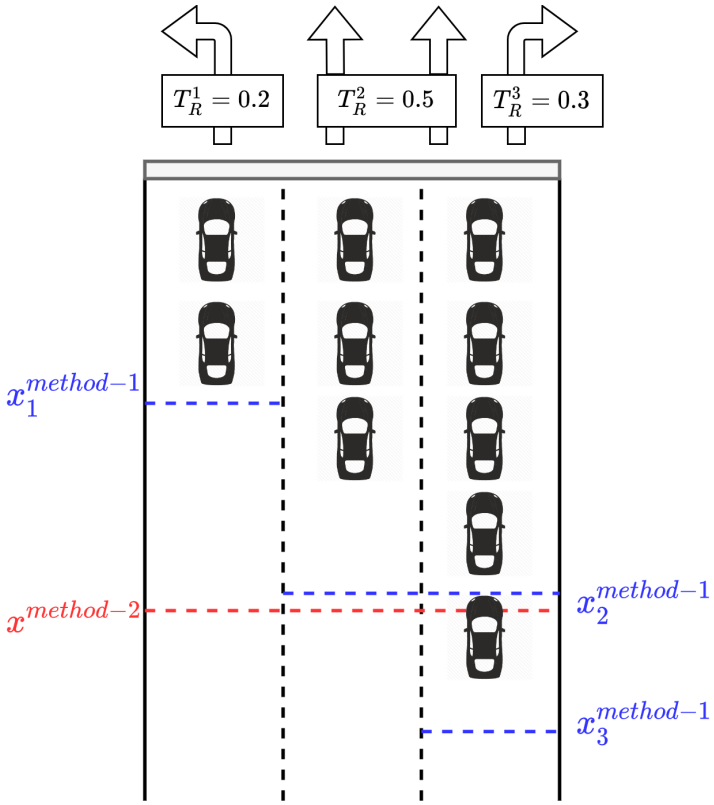


Figure 4.19: Turning ratio distribution for a 3-lane road link. Queuing vehicles' positions are derived based on exiting direction utilization probability.

utilization probabilities for lanes with multiple exiting directions. In other words, if a lane allows for $k > 1$ exiting direction, its utilization probability will sum up from the weighted turning ratios of each of these directions. Let p be the number of queuing vehicles at the intersection, then, considering both assumptions, we derive the expected queue size for a lane i:

$$N_q^i = p \sum_{j \in D_i} \frac{T_R^j}{n_j}, \tag{4.3}$$

where n_j is the number of lanes allowing for the exiting direction j.

Example 1. *Consider a 3-lane road link from Fig. 4.19 with 3 possible exiting directions: left, straight and right. The turning ratios for these directions are* $T_R^1 = 0.2$, $T_R^2 = 0.5$, $T_R^3 = 0.3$ *respectively. Note that the leftmost lane is a left-turn lane; from the middle lane vehicles can go only straight; and the rightmost lane can be used for both straight and right movements. This corresponds to* $n_1 = 1$, $n_2 = 2$ *and* $n_3 = 1$. *Assuming the number of queuing vehicles is 10 (*$p = 10$*), we can derive queue sizes for each of the three lanes, using Eq. 4.3:*

$$N_q^1 = p \times \frac{T_R^1}{n_1} = 10 \times \left(\frac{0.2}{1}\right) = 2$$
$$N_q^2 = p \times \frac{T_R^2}{n_2} = 10 \times \left(\frac{0.5}{2}\right) = 2.5$$
$$N_q^3 = p \times \left(\frac{T_R^2}{n_2} + \frac{T_R^3}{n_3}\right) = 10 \times \left(\frac{0.5}{2} + \frac{0.3}{1}\right) = 5.5$$

One of the possible vehicle distributions among lanes is reflected in the figure.

The further queue tail location estimation can be performed using one of the two methods depending on the vehicle's navigation data accessibility and the existence of proper communication channels. The first method relies on a vehicle's ability to share its desired exiting direction j with the infrastructure via V2I communication. In this case, the system needs to analyze only a subset of available lanes allowing the specified direction j, which negates the impact of irrelevant lanes on the queue length estimation. The expected queue tail position conditioned on the knowledge of the vehicle's direction preference is give by:

$$x^{method-1} = \mathbb{E}[\hat{x}|j] = (l+h) \times \frac{1}{n_j} \sum_i N_q^i \mathbb{1}_{D_i}(j), \tag{4.4}$$

where l is the average vehicle length, h is the average headway distance between two queuing vehicles, $\hat{x}$ is the random variable that takes values $N_q^1, \ldots, N_q^m$ with probabilities corresponding to the likelihoods of the vehicle taking each of the m lanes, and $\mathbb{1}_A : X \mapsto \{0, 1\}$ is the indicator function, defined as

$$\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Going back to the Ex. 1, the expected queue end locations of each potential exiting direction for the incoming vehicle:

$$\textit{Left: } x_1^{method-1} = (l+h) \times N_q^1 = \mathbf{2(l+h)}$$
$$\textit{Straight: } x_2^{method-1} = (l+h) \times \frac{1}{n_2}\left(N_q^2 + N_q^3\right) = \mathbf{4(l+h)}$$
$$\textit{Right: } x_3^{method-1} = (l+h) \times N_q^3 = \mathbf{5.5(l+h)}$$

Corresponding predicted stopping positions are represented by the blue lines in the Fig. 4.19. Note that the left and right directions are less than a vehicle-length away from the ground truth. Straight direction has a slightly higher error, however, this discrepancy is not significant, considering the manual speed adjustment required from the driver when switching from the Speed Advisory System to a car-following model.

The second method assumes no access to the vehicles' preferences and no means of predicting the desired exiting directions. Therefore, all exiting directions must be accounted for when computing the expected queue tail position:

$$
\begin{aligned}
x^{method-2} =& \mathbb{E}[\hat{x}] = (l+h) \times \sum_i N_q^i \mathbb{P}(\text{the vehicle needs lane } i) = \\
=& (l+h) \times \sum_i \sum_{j \in D_i} N_q^i \frac{T_R^j}{n_j} = \\
=& (l+h)p \times \sum_i \left(\sum_{j \in D_i} \frac{T_R^j}{n_j} \right)^2
\end{aligned} \tag{4.6}
$$

Applying the second method to the system setup from Ex. 1, we calculate the unconditional expected queue tail location for the incoming vehicle:

$$
\begin{aligned}
x^{method-2} =& 10(l+h) \times \left(\left(\frac{T_R^1}{n_1} \right)^2 + \left(\frac{T_R^2}{n_2} \right)^2 + \left(\frac{T_R^2}{n_2} + \frac{T_R^3}{n_3} \right)^2 \right) \\
=& 10(l+h) \times \left(\left(\frac{0.2}{1} \right)^2 + \left(\frac{0.5}{2} \right)^2 + \left(\frac{0.5}{2} + \frac{0.3}{1} \right)^2 \right) \\
=& \mathbf{4.05(l+h)}
\end{aligned}
$$

The estimated stopping position is represented by the red line in the Fig. 4.19. Note a significantly larger error for the left direction prediction compared to a similar value obtained by the first method. This is due to a major contribution of the other two directions (80%) in the queue end position computation while having no impact on the actual left-lane queue formation.

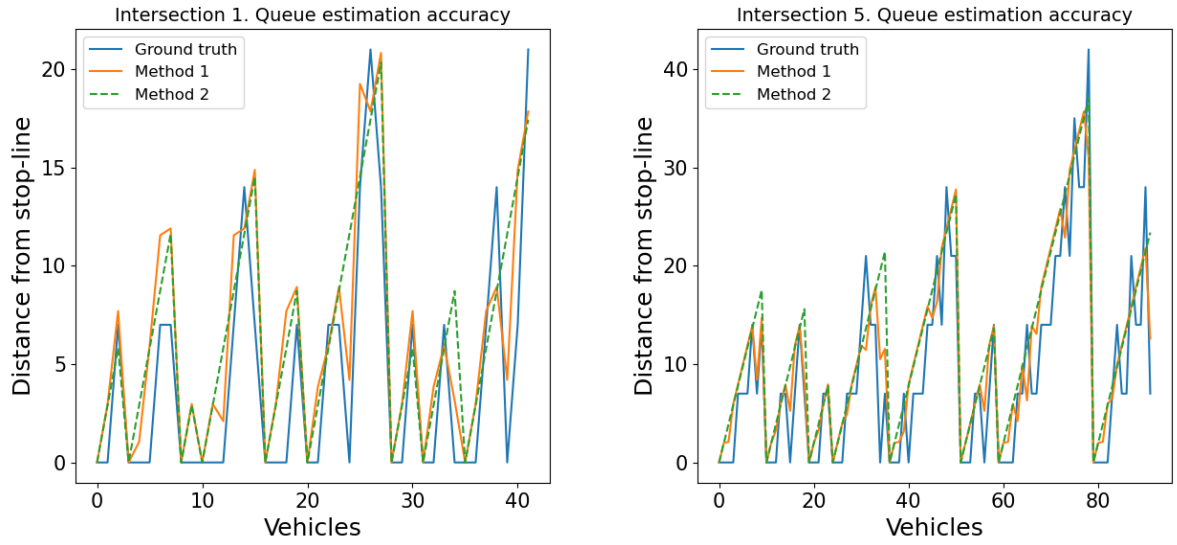
Simulations and Results

To evaluate the accuracy of the proposed methods and their impact on fuel consumption, intersection progression quality and phase utilization, we focus on the Montgomery County network SUMO simulation (Fig. 2.1), which is similar to the one discussed in Section 4.3. More specifically, we present the comparison between different vehicle configurations: with or without the active SAS; using the original near-optimal speed trajectory computation or the queue-related one.

Queue Tail Location Estimation Accuracy

The queue tail estimation methods play an important role in the near-optimal speed trajectory computation. The main purpose of the queue prediction implementation, fuel consumption reduction and intersection progression improvement, can be corrupted by incorrect estimations causing traffic flow disturbance. To avoid the undesirable system behavior we must first evaluate the accuracy of the proposed methods. Since the lane-to-lane count of queuing vehicles depends only on the total number of these vehicles and the intersection's turning ratio distribution, different SAS-quipped vehicle penetration rates and acceleration

options (discussed in Section 4.3), having no impact on the prediction accuracy, will not be considered. To simplify the simulation process, we assume a fixed-known acceleration set-up and a 50% SAS-penetration rate. The prediction accuracy is evaluated based on the difference between the predicted queue tail location for a particular vehicle and its actual stopping position (both in meters from the intersection stop-line). The simulation results for the two under-saturated intersections are presented in Fig. 4.20.



(a) (b)

Figure 4.20: Queue tail location comparison between two prediction methods and the ground truth. (a) Low demand - Intersection 1. (b) Moderate demand - Intersection 5.

Due to the complete queue dissipation between cycles, each new queue prediction starts from the point zero growing as vehicles arrive at the intersection. General graph shapes for low and medium demands are similar despite the significant difference in the numbers of queuing cars. According to the results, both methods managed to estimate the queue tail positions relatively accurately considering the total queue sizes. The predictions provided by the first method, as expected, are more precise (within 2 vehicle-lengths from the ground truth and with the mean error of 3 meters) and are able to track the queue formation for directions with low turning ratios. The second method, on the other hand, results in a slightly higher average error of 5 meters and up to 3 vehicle-lengths deviations from the actual values at maximum. These large over-estimations are mostly related to the less probable exiting directions as discussed in the Ex. 1.

Fuel Consumption

The first metric we discuss is fuel consumption. In order to obtain a comprehensive comparison we ran 24 simulations with various combinations of the following settings: (1) 50% and 100% SAS-equipped vehicle penetration rates; (2) three acceleration options (fixed known, random known and random unknown); (3) four algorithm set-ups (no SAS, SAS without the queue estimation and SAS with the both queue estimation methods).

According to Fig. 4.21, the first queue tail location estimation method utilizing the shared vehicles' desired exiting directions demonstrates the best performance among the considered algorithms. It provides up to 4 additional percent of fuel savings, compared to the no-queue estimation scenario. The second method, although is less effective than the first one, still manages to reduce fuel consumption by up to 3 additional percent. The highest energy-wise improvement is observed at intersections with moderate demands, since the queues on the corresponding incoming links are large enough to cause significant speed trajectory modifications when the queue estimation methods are implemented. Moreover, varying SAS-penetration rates and acceleration types seems to have no influence on the queue estimation algorithm performance. The only intersection that does not benefit from the queue estimation is the intersection 7, which experiences high demand and is over-saturated most of the time. However, since we mentioned that the proposed labeling algorithm is ineffective for over-saturated intersections, this result is expected and can be discarded from the further analysis.

Phase utilization

In this section we analyze the impact of the queue estimation methods on phase utilization. Fig. 4.22 demonstrates the differences in phase utilization for Intersections 1 (low demand) and 5 (moderate demand). According to the graphs, neither the free traffic scenario nor the moderate demand case are affected by the introduction of the queue prediction models. These findings follow directly from the fact that the implementation of our queue tail position estimation algorithms does not change vehicles' labels, i.e. ability to cross the intersection, but rather provides better speed trajectories for the stopping cars.

Progression Quality

Now, we focus on another performance measure, progression quality, which is well characterized by the POG value (Fig. 4.23). The improved near-optimal speed trajectories derived using the queue tail predictions from our model, exhibit smoother deceleration patters keeping the vehicle in motion for a longer time. In some cases, this time difference is sufficient to postpone the vehicle's arrival to the queue tail until the phase switch, increasing the Percent-on-green and improving the progression quality at the intersection.

Considering the intersection 1 with the low traffic demand, progression quality can hardly be improved by the queue estimation methods, since the number of incoming vehicles is in-

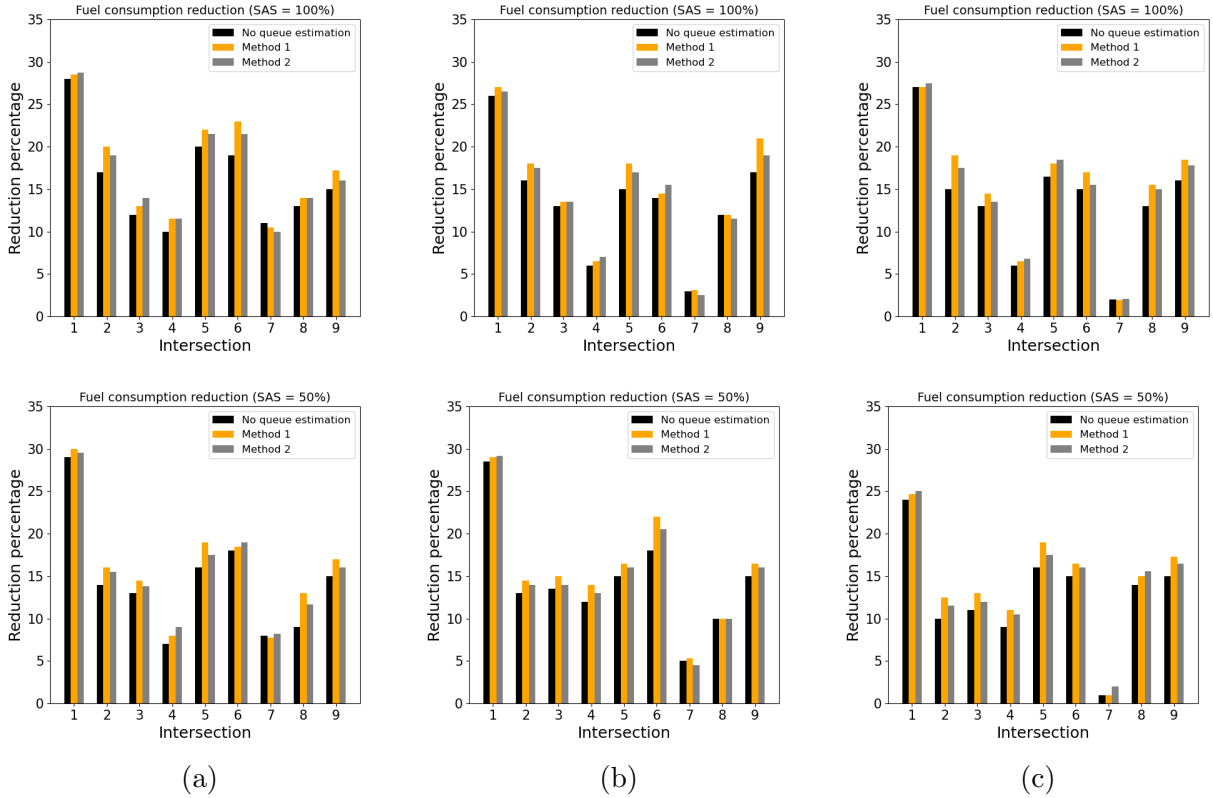


Figure 4.21: Fuel consumption reduction for no queue estimation scenario and for both estimation methods. (a) Fixed known acceleration. (b) Random known acceleration. (c) Random unknown acceleration.

sufficient for forming long enough queues. The queue tail position, as was shown in Fig. 4.20, is usually located relatively close (1-3 vehicle-lengths) to the intersection stop-line, making the new speed trajectory almost identical to the one built without the queue estimation. Intersection 5, on the other hand, benefits more from the implementation of our prediction models. Although, the average progression quality improvement is relatively low (≈ 0.05 and ≈ 0.03 for methods 1 and 2 respectively), high values of peak POG changes (up to 0.25 and 0.16 for methods 1 and 2 respectively) suggest significant improvement of traffic propagation at the intersection.

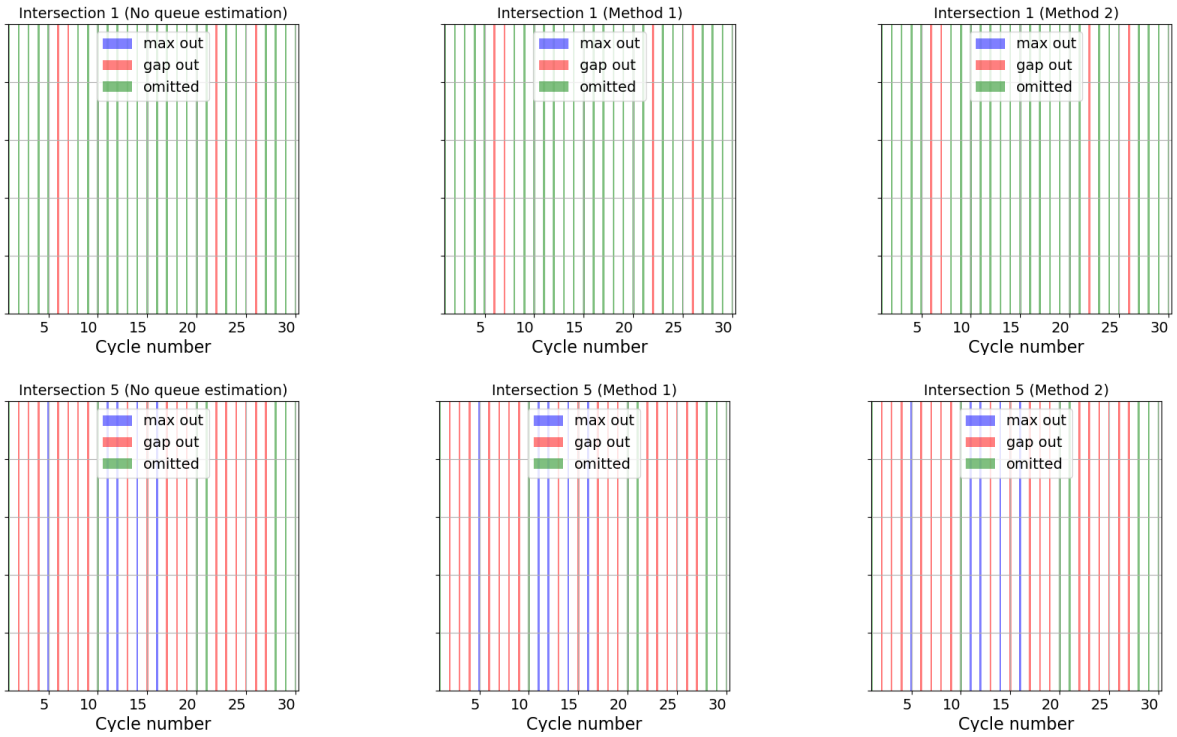
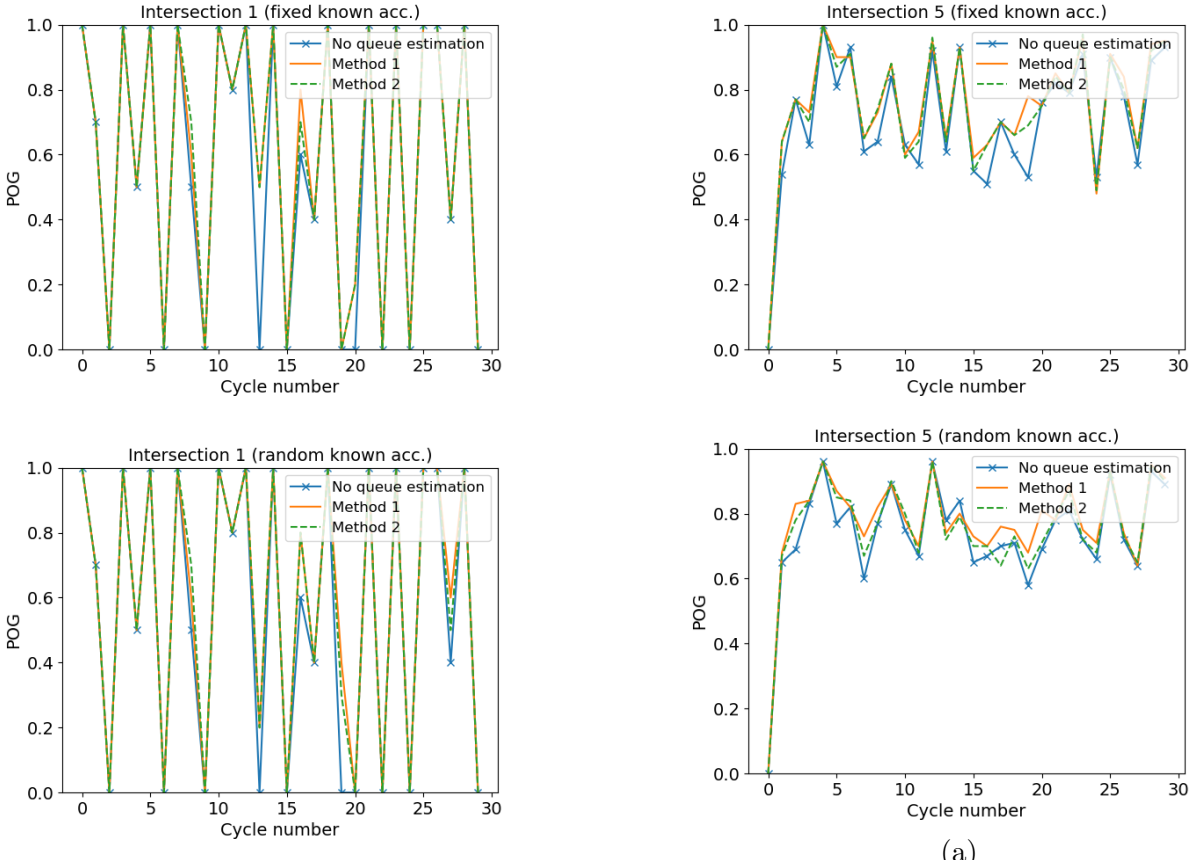
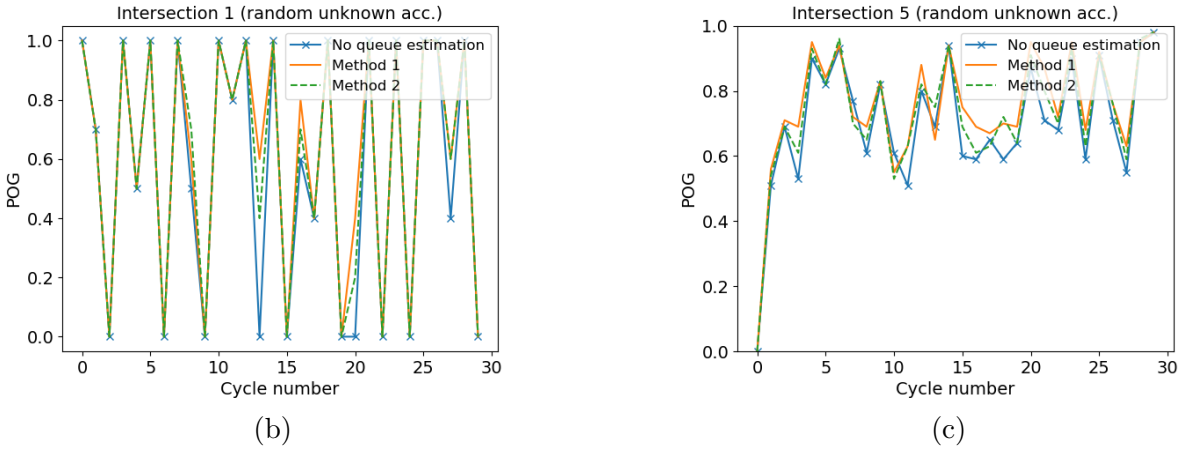


Figure 4.22: Phase utilization comparison for no queue estimation scenario and for both estimation methods. (a) No queue estimation. (b) Method 1. (c) Method 2.



(a)



(b) (c)

Figure 4.23: Percent-on-Green (POG) for progression quality comparison with no queue estimation and both prediction methods implemented. (a) Intersection 1 - Low demand. (b) Intersection 5 - Moderate demand.

Chapter 5

Parking Reservation

5.1 Introduction

With the continuing growth of number of vehicles in cities, transportation systems suffer from congestion, over-utilization and inefficient usage of resources. In addition to flow-related issues, such as exceeding road capacities, traffic shock-waves, etc., curb utilization plays a significant role in stressing the network. Insufficient parking space forces ordinary cars to cruise around in search for parking, slowing down traffic. On the other hand, commercial vehicles (delivery trucks, taxis, etc.) follow different behavioral patterns and more often resort to double-parking, thus blocking the road and creating a bottleneck effect (Fig. 5.7).

One of the possible parking management strategies is *online reservation*. Several systems were proposed for allowing drivers to make online reservations for parking spots [75], [76], [77], [78]. These models are designed to inform drivers about parking availability and pricing via the Internet, SMS or any other type of wireless communication. These models alone, however, do not provide sufficient tools for influencing drivers' decisions and managing parking demand distribution for congestion mitigation. In addition to a reservation mechanism, reference [79] introduces a system that dynamically adjusts parking prices to prevent fragmentation during the allocation of the parking slots, which is likely to improve the efficiency of traffic management.

Dynamic parking pricing itself is an extensively studied (e.g. in [80], [81], [82]) and widely implemented mechanism for curb utilization management. San Francisco was a study ground for a demand-responsive parking pricing project (SFpark) conducted from 2011 until 2013 and evaluated in [83]. Over the course of two years, SFMTA made 10 on-street rate adjustments to keep the occupation level at around 80%. Many side benefits, such as greenhouse gas emission reduction, traffic improvement, peak hour congestion decrease, etc. were reported as a result of dynamic pricing. Similar projects of smart parking include the SeaPark in Seattle and LA ExpressPark program in Downtown Los Angeles.

An extensive data-driven study of the spatio-temporal characteristics of curbside parking is presented in [84]. Using GPS location data, parking transaction data and block-face supply

data, the work estimates parking demand and analyzes parking occupancy. It further models parking demand using a Gaussian mixture model to cluster spatially close regions with similar demand and occupancy profiles for more effective parking policy implementation.

Existing literature on delivery vehicle behavior modelling and its impact on traffic conditions can be divided into three groups: analytic models, simulation-based studies and discrete-choice models. Among the analytic studies is the paper [85] – a generalization of [86] – which describes a downtown parking model with delivery truck behavior and analyzes traffic congestion with respect to the number of parking spots allocated for commercial vehicles and parking pricing.

The simulation-based study [87] analyzes the impact of double-parking on average travel time via $M/M/\infty$ queuing model and micro-simulation model. Another paper,[88], simulates city traffic and delivery vehicles' parking with respect to varying curb allocation to loading/unloading zones and analyzes the environmental impact of the presence of such zones. The common limitation of both papers is that other policies, such as parking enforcement and dynamic pricing, were not considered. The papers [89], [90], and [91] analyze urban logistics, parking policies and their impact on the system from a strategic point of view. The first two study logistics companies' sensitivity toward parking pricing, availability of loading/unloading bays and probability of finding them free via preference data collection. The third work focuses on the receivers' attitude towards off-hours delivery strategies and the policy which uses Urban Distribution Centers.

The work [92] combines a simulation model capable of evaluating the traffic impact of parking policy changes on truck parking choice and discrete parking choice model. A binary logit model for determining the probability of parking at a location is estimated from truck driver surveys and commercial vehicle parking observations. Drivers' willingness to occupy specific locations are predicted based on the distance from their final destinations and parking spot types (loading zone or on-street). Agent-based simulation software is used to evaluate parking space allocation. This work, however, does not consider parking pricing, enforcement and illegal parking.

The work [93] models commercial vehicle driver parking choice behavior in the presence of three possible parking options: loading zones, carpark areas and illegal on-street parking. By applying different parking management strategies, such as dynamic pricing, parking capacity variation, etc., and considering the proposed choice model, the study estimates the financial and ecological impact on the transportation system. However, overall traffic congestion improvement in the area of interest is not specifically studied.

In our work we put aside the idea of dynamic parking pricing for curbside management due its impractical nature with respect to delivery vehicles. These vehicles occupy parking spots for relatively short periods of time, do not pay parking fees and may find it expedient to double park in the absence of a rapid law enforcement system. Instead, we focus on methods which place physical limitations on delivery vehicles' parking behaviour. Using an intelligent partitioning of day-to-day operation time we are able to indirectly affect delivery vehicle arrival times and encourage drivers to follow an optimal schedule.

Our model builds a terminal utility function based on the drivers' utilities, available

partitions and city's preferences regarding demand distribution and delivery vehicles arrival times. By solving a series of optimization problems we obtain an optimal partitioning that maximizes social utility of the system at equilibrium.

The proposed system can be implemented in a parking reservation app to obtain a centralized information tool that connects all drivers, displays parking availability, structures and synchronizes parking requests and collects data (driver's preferences) for further analysis and model improvement. In addition, cities can equip busy areas with cameras to enforce the planned parking schedule.

5.2 Time-slot modelling

Delivery truck drivers, unlike ordinary car owners, must follow specific work schedules, which limits possible delivery times. Corporate rules and regulations establish certain operation hours, outside of which no delivery can be processed (e.g. 6 am - 10 pm). We assume each business day can be viewed as an isolated and independent structure that has no direct effect on the subsequent day. Delivery vehicles should be able to complete their deliveries within the current day and proposed schedules should not be transferred to the following days. Therefore, we focus on scenarios that yield feasible solution to the partitioning problem, providing suitable reservation periods for all participating agents.

The main objective of our system is reducing traffic congestion caused by double-parking. Therefore, we must ensure that no two delivery trucks arrive at the same parking location at the same time. Since delivery vehicles often do not pay for parking, well-known dynamic parking pricing methods are not effective in altering their behavior. Instead, we propose a new approach where we partition the operation hours into time slots, which we use as control knobs to influence delivery patterns. Drivers reserve a parking spot for a particular time-slot before making a delivery. Under the assumption of parking law enforcement implementation, violating the time-slot length limitation results in large fines to discourage delivery drivers from overstaying. We discuss how to design the optimal time slots in the following sections.

Reservation Model

Our reservation model is based on the following rules:

1. Time intervals for a parking spot can not overlap.
2. Only one vehicle can reserve a parking time slot at a time.
3. For each parking spot, a vehicle can reserve only one time slot at a time. To make a new reservation the current reservation must be cancelled or must have expired.
4. In the event of a double parking incident, the vehicle that does not have a reservation is penalized.

5. A time slot can only be reserved before the beginning of the corresponding time slot and, at any given time, a driver can see all of the available time slots for the rest of the operation hours.

The reason for the first two rules is prevention of double parking. The third rule prevents drivers from reserving several time slots at once when the arrival time is uncertain. The fourth rule states that we only penalize the vehicle that does not have a reservation regardless of the vehicle that double parks (if none of the vehicles have a reservation then the double parked car is penalized). The last rule allows for First-come-first-serve drive-ins. If a parking slot is unreserved and unoccupied after the beginning of a time-slot, any delivery vehicle or ordinary car can safely park there and not be forced away by a late reservation. In this model the decision variable of the designer is the length of each time slot. These time slots are adjusted before any reservations take place. Each driver is able to see the available time slots and their duration for the respective finite horizon (in this work we consider this to be around five to six hours).

Time Slots

As the designer, our objective is to create a set of time slots in the operation hours compliant with the proposed reservation model. Since time-slots do not overlap and leaving gaps in between consecutive time-slots results in under-utilization, a new time-slot must begin when another one ends (except the time-slot at the end of the operation hours). Moreover, since time slots are disjoint and their union is equal to the operation hours, we call each set of time slots a *partition* of operation hours. To keep the problem tractable, we assume that each time slot is a multiple of some predefined interval of length δ units of time, which we refer to as δ-interval. This assumption is reasonable, since arrival time estimation is unlikely to be precise due to inaccurate traffic state assessment, delays at other delivery locations, etc.

Suppose the operation hours are divided into m consecutive δ-intervals with starting times, which we refer to as δ-times, enumerated with the index $k \in \{0, 1, \ldots, m-1\}$. Therefore, if the operation hours start at time t_0, then k refers to the δ-time $(t_0 + k\delta)$. To indicate whether a δ-time marks the start of a new slot, we define the binary vector $y = [y_0 \quad y_1 \quad \ldots y_{m-1}]^T$ such that, for $k = 0, 1, \ldots, m-1$,

$$y_k = \begin{cases} 1, & \text{if the } k^{th} \text{ } \delta\text{-time is the beginning of a new slot} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

Note that the cardinality of the vector y is the number of distinct time-slots. Denoting this number as l, we partition $\{0, 1, \ldots, m-1\}$ into disjoint intervals (t_j^s, t_j^e), $j = 0, 1, \ldots, l-1$, where t_j^s and t_j^e are starting and ending δ-times in the time-slot j.

5.3 Driver Behavior, Assumptions and Equilibrium

Our objective is to find a partition of the operation time that maximizes the social utility (5.7) at the equilibrium. Since we are concerned about the social welfare of the system, which depends on drivers' preferences, city's parking requirements and traffic state, developing suitable utility functions is an essential step in posing an optimization problem.

Driver utility

We assume that drivers act selfishly to maximize their own utility functions. We further assume that drivers make reservations as soon as they know the time that they need to park and this information is available to the designer.

The following set of approximate parameters reflects a driver's delivery inclinations:

- $\boldsymbol{x} \in \mathbb{R}$: preferred arrival time for a particular location;
- $\boldsymbol{p} \in \mathbb{R}$: processing time required to drop off a delivery item (taken to be deterministic, since drivers usually have a good estimate of the time they need);
- $\boldsymbol{r} \in \mathbb{R}$: reservation time, which is an approximate time when a driver thinks they will have a better estimate for their arrival time to be able to make a reservation in the app. We assume $r_i \neq r_j$ for all $i, j < n$, where n is the total number of vehicles.

The utility function of a driver i, referred to as general utility $U_i : \mathbb{R} \mapsto \mathbb{R}$, is a normalized function of the potential arrival time that monotonically increases until reaching the maximizer and monotonically decreases afterwards, sampled at the δ-times. A more complex and accurate shape of a utility function can be obtained once the system is deployed and sufficient operational data is gathered. Note that the general utility function is independent of the partition since it simply reflects driver's preference.

Cost function and Net Utility

We now define cost and utility functions for delivery vehicles based on the partitioning structure introduced above. Drivers would like to avoid being penalized for overstaying. Therefore, we derive a specific cost function that represents such penalty:

$$C_i(k, y) = \max_{j>k}[y_j g(p_i - (j - k))] + (1 - y_k), \tag{5.2}$$

where $\boldsymbol{g} : \mathbb{R} \mapsto \mathbb{R}$ is a step-function.

Here the first term represents the penalty incurred by overstaying and the second term mathematically represents impossibility to reserve non-existing time-slots.

After combining the general utility function $U_i(t)$ with the corresponding cost function $C_i(k, y)$, we obtain the net utility function of driver i:

$$U_i^n(k, y) = U_i(t_0 + k\delta) - C_i(k, y) \tag{5.3}$$

This net utility function represents a driver's actual preference space with respect to potential partitions. Note that zero values as well as negative values of $U^n(k, y)$ are essentially equivalent and both correspond to absolute inability or prohibition to arrive to the parking location at this specific δ-time k. Therefore, we focus on the positive utility values in the further analysis.

Equilibrium points

The equilibrium strategy profile is defined as the best response for all agents given the partitioning and the set of available time-slots at the time of reservation. That is, each driver books a time slot among the time slots which have not been reserved so far, that maximizes their utility. Therefore, equilibrium can be obtained by solving one of the optimization problems (Eq. 5.4, 5.5) for every partition and for every participating agent. Without loss of generality, we index delivery vehicles in the order of their reservation times; therefore $r_1 < r_2 < \cdots < r_n$.

Given the partition y, the optimal arrival time for driver i is obtained as follows:

- For the delivery truck with the highest priority ($i = 1$):

$$\begin{aligned} t_1^*(y) = \arg\max_k \quad & U_1^n(k, y) \\ \text{s.t.} \quad & U_1^n(k, y) > 0, \\ & t_0 + k\delta \geq r_1. \end{aligned} \tag{5.4}$$

- For delivery trucks with $i > 1$:

$$\begin{aligned} t_i^*(y) = \arg\max_k \quad & U_i^n(k, y) \\ \text{s.t.} \quad & U_i^n(k, y) > 0, \\ & t_0 + k\delta \geq r_i, \\ & t_0 + k\delta \neq t_j^*(y), \quad \forall j : j < i. \end{aligned} \tag{5.5}$$

If there is no feasible solution to an optimization problem given a partition (there are not suitable time-slots for the corresponding delivery vehicle), the utility of that vehicle given this partition is zero and no time-slot is occupied, leaving the decision space of the later drivers unchanged. In a more restrictive case, where each delivery truck must be accommodated, such partitions can be discarded from further analysis.

Above we have characterized drivers' best responses independent of traffic conditions and city's requirements. Selecting the optimal partition now might result in scenarios when multiple delivery trucks attempt package drops during rush hours, which have a significant negative effect on congestion level and curb over-utilization. Therefore, some adjustments to the model taking into account the city's penalty function $P : \mathbb{R} \mapsto \mathbb{R}$, which reflects traffic conditions throughout the day, congestion levels and willingness to allow trucks occupy

parking locations at specific times (Fig. 5.1) are required. This utility is not static, but rather it may vary day to day, month to month, season to season. Such flexibility allows for a more precise model tuning using the data obtained from curb activity monitoring and drivers decisions recorded in the app.

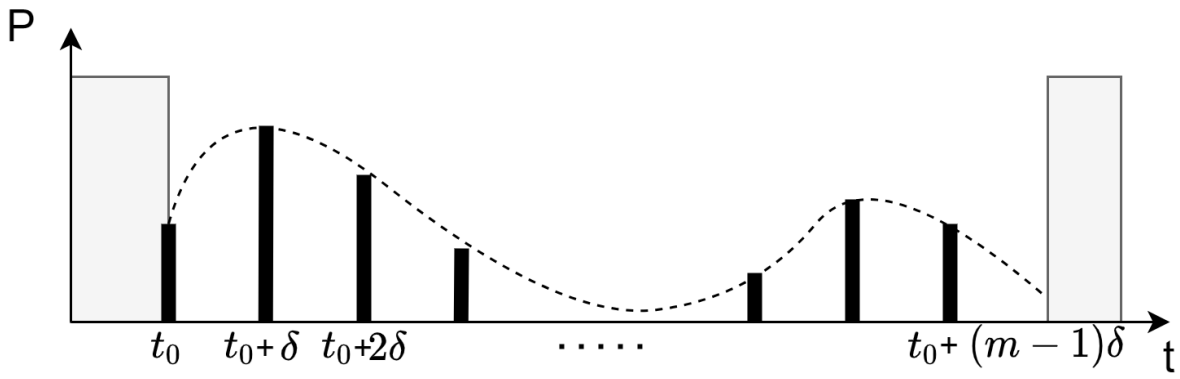


Figure 5.1: General form of the city's penalty function. To prevent excessive traffic load during morning and evening rush hours, these time periods are penalized more than the rest of the day.

The city's penalty function is further subtracted from the net utilities to construct terminal utility functions:

$$U_i^T(k,y) = U_i^n(k,y) - P(t_0 + k\delta) \tag{5.6}$$

As the designer, our objective is to select a partition of the operation time which achieves optimum at a corresponding equilibrium strategy profile. This selection is made via solving an optimization problem:

$$\begin{aligned} \max_{y} \quad & \sum_{i=1}^{n} (U_i^T(t_i^*(y), y) - \|(t_j^e - t_j^s) - p_i\|) \\ \text{s.t.} \quad & t_j^s = t_i^*(y) \end{aligned} \tag{5.7}$$

This formulation considers terminal utilities evaluated at the optimal arrival times t^* preserving the local equilibrium derived at the previous step. The second term is responsible for penalizing under-utilization. If the assigned time-slot significantly exceeds the vehicle's processing time, the parking location would be unoccupied but still reserved after the truck's departure, causing under-utilization. The optimal partition would be made visible to the drivers through the mobile app used for implementing the system.

5.4 Simulations

Sample problem simulation

In this section we demonstrate our model on a sample problem. For simulations we assume $\delta = 1$ unit of time and $t_0 = 0$. Assume the duration of operation hours is $m = 3$ units of time, which results in 4 different possible partitions: $y = [1 \quad 0 \quad 0]^T$, $y = [1 \quad 0 \quad 1]^T$, $y = [1 \quad 1 \quad 0]^T$, and $y = [1 \quad 1 \quad 1]^T$. Assume there are 2 trucks with general utility functions $U_1(t)$ and $U_2(t)$ respectively. In this example, we use a specific form of general utility function (Fig. 5.2) with parameters (k^l, k^r) characterizing the "flexibility" of a driver. Larger values of these parameters indicate less flexibility in arrival time. Here k^l is the left slope and $-k^r$ is the right slope of the function around the maximizer. If the maximizer lands on the first or the last δ-time, $k^l = 0$ or $k^r = 0$ respectively.

The complete sets of parameters (from Section 5.3) for trucks 1 and 2 are: $\{x_1 = 1, p_1 = 2, r_1, (k^l, k^r) = (0.5, 0.5)\}$ and $\{x_2 = 0, p_2 = 1, r_2, (k^l, k^r) = (0, \frac{1}{3})\}$ respectively. Note that $r_1 < r_2 \leq 0$.

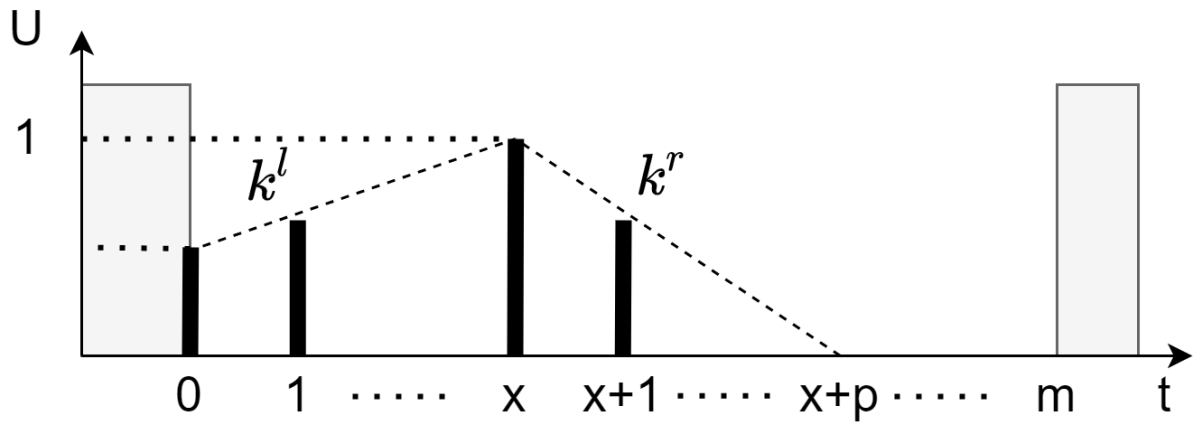
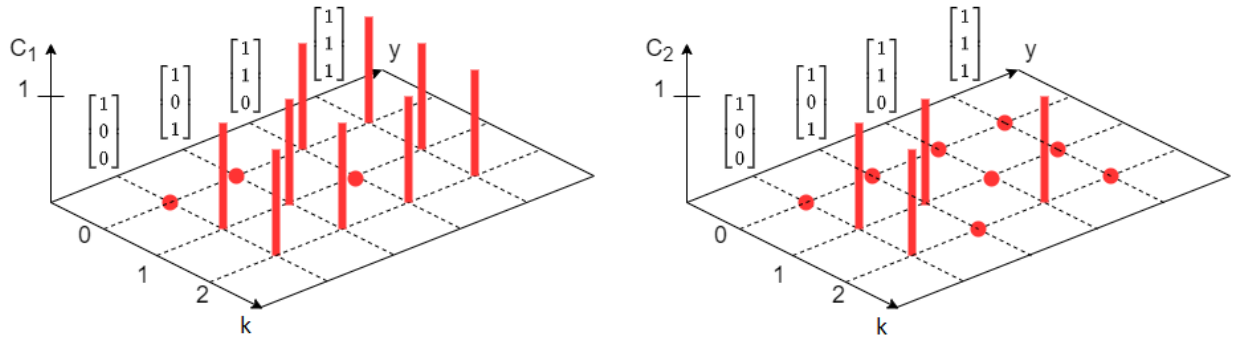


Figure 5.2: General utility function of a driver. The peak corresponds to the driver's desired delivery time and the parameters (k^l, k^r) represent the driver's flexibility.

The next step is to construct the cost functions from Eq. 5.2. Since $p_1 = 2$, the first delivery vehicle requires $y_i = 1, y_{i+1} = 0$ for some $0 \leq i \leq m - 1$ within the partition. Therefore, $C_1(k, y) = 1$, unless $y_k = 1$ and $y_{k+1} = 0$, where $C_1(k, y) = 0$ (Fig. 5.3a). On the other hand, the second truck is physically able to fit into any time-slot, thus, $C_2(k, y) = 0$ if $y_k = 1$ and $C_2(k, y) = 1$ otherwise (Fig. 5.3b).

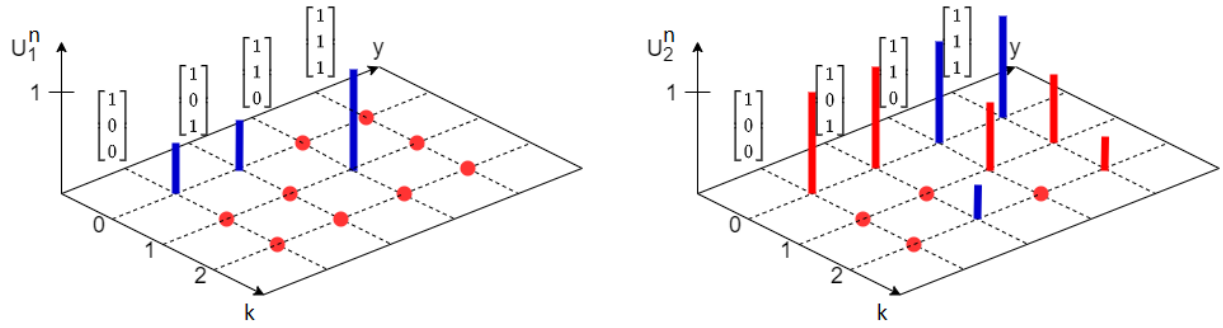
At this stage the net utility functions can be derived for both trucks. Applying Eq. 5.3, we notice that there are only three potential arrival times for the first truck within the time-partition space (Fig. 5.4a). The second delivery vehicle, however, has a significantly wider choice of suitable time-slots (Fig. 5.4b). Another notable detail is that some partitions



(a) (b)

Figure 5.3: Cost functions reflecting the overstaying penalty and mathematically representing impossibility to reserve non-existing time-slots. (a) Truck 1. (b) Truck 2.

cannot provide any suitable time-slots for one of trucks ($y = [1 \quad 1 \quad 1]^T$ for the first truck), and, therefore, are discarded from further consideration.



(a) (b)

Figure 5.4: Net utility functions for both delivery vehicles. Optimal arrival time $t^*(y)$ for each partition is highlighted blue. (a) Truck 1. (b) Truck 2.

Having both net utility functions derived, we can proceed to identifying the optimal arrival times $t_i^*(y)$ for $i = \{1, 2\}$ as functions of a partition via solving optimization problems (5.4) and (5.5). As mentioned before, we use a brute-force approach iteration over all partitions and arrival times for each truck and finding the equilibrium. Note that the partition $y = [1 \quad 0 \quad 0]^T$ yields no optimal arrival time for the second truck since the only arrival time with positive utility (at time $t = 0$) is reserved by truck 1 with higher priority. Such

partitions will not be considered either. We summarize the optimal arrival times in Tab. 5.1.

Assume further that due to the morning and evening rush hours it is undesirable to have many delivery trucks at those periods. Therefore, the city's penalty function ($P(t)$) has the shape presented in Fig. 5.5. Times 0 and 2 represent morning and evening times respectively.

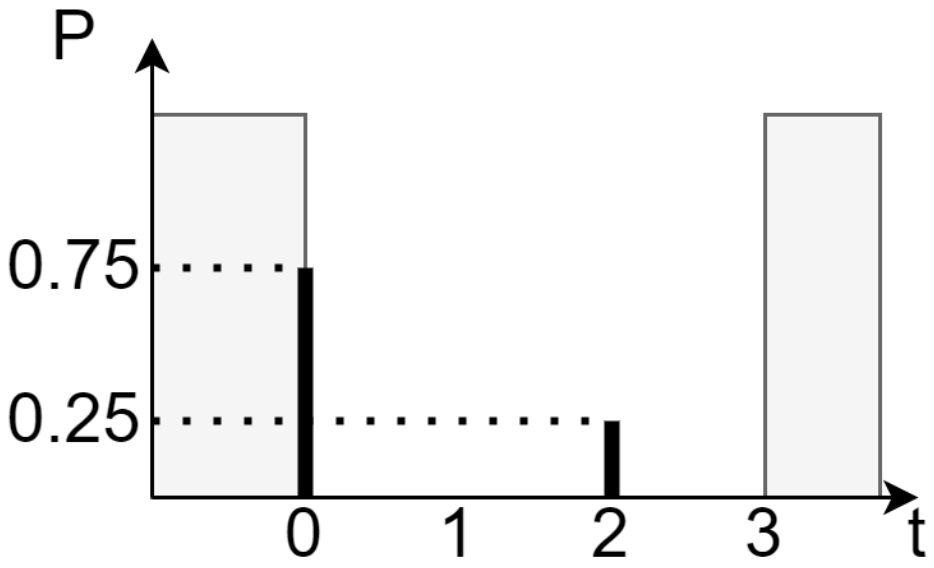


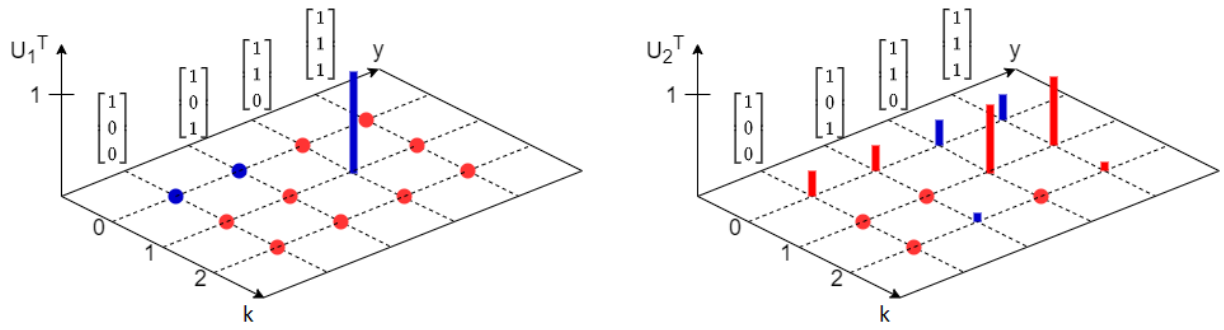
Figure 5.5: The city's penalty function used in the sample problem. Times 0 and 2 represent rush hours, and, therefore, are penalized more.

Lastly, a designer is able to derive terminal utility functions (Fig. 5.6) for both delivery vehicles using Eq. 5.6 and solve the optimization problem (5.7) to find the optimal partition that yields maximum social utility at equilibrium. In our scenario, the optimal partition is $y = [1 \quad 1 \quad 0]^T$ with the corresponding total utility of 1.25 units. According to the equilibrium arrival times, the first truck reserves a time-slot $(1, 3)$ and the second truck occupies the time-slot $(0, 1)$. All intermediate and final values are compiled in Tab. 5.1.

Bancroft Way Simulation

For a more realistic example, we chose to study Bancroft Way, Berkeley, CA, on the southern periphery of the UC Berkeley campus. Curb management is of major interest in this location due to heavy traffic and excessive delivery activity to nearby businesses, which adds to congestion. We simulate a single busy location (Fig. 5.7) in that area using historical traffic data and derive an optimal time-slot configuration to mitigate parking issues.

Since we are using an extensive search, we chose to implement a shorter period of interest (5-hour window with $\delta = 15$ minutes) for parking reservation. Therefore, the total number



(a) (b)

Figure 5.6: Terminal utility functions. Optimal arrival time $t^*(y)$ for each partition is highlighted blue. (a) Truck 1. (b) Truck 2.

Table 5.1: Optimal arrival times and corresponding utility functions values.

Partition (y)	$[1\ \ 0\ \ 0]^T$	$[1\ \ 0\ \ 1]^T$	$[1\ \ 1\ \ 0]^T$	$[1\ \ 1\ \ 1]^T$
$t_1^*(y)$	0	0	1	N
$t_2^*(y)$	N	2	1	1
$U_1^n(t_1^*(y), y)$	0.5	0.5	1	0
$U_2^n(t_2^*(y), y)$	0	$1/3$	1	1
$U_1^T(t_1^*(y), y)$	0	0	1	0
$U_2^T(t_2^*(y), y)$	0	$1/12$	0.25	0.25
Total Utility	0	$1/12$	1.25	0.25

of potential starting times (number of δ-intervals) is 20, which results in $2^{19} = 524288$ partitions.

According to the data collected from dash cameras installed on UC Berkley campus busses, a daily average number of distinct delivery trucks occupying the parking spot is 6, which will be reflected in our simulation. Moreover, the city's penalty function (Fig. 5.8) can be approximated from the same data via demand distribution recognition and congestion levels estimation. Utility function parameters for each driver are summarized in Tab. 5.2. Note that $r_1 < \cdots < r_6 \leq 0$.

According to simulation results (Fig. 5.9), we managed to accommodate all delivery vehicles avoiding arrival overlaps and potential double-parking. Moreover, the maximum social utility of 4.633 units at equilibrium is achieved, resulting in minimal potential deviation in drivers choices from the predicted behavior. In addition, the obtained partitioning demonstrates complete utilization of time-slots, opening the parking location for other possible purposes when not occupied.

Figure 5.7: A dashboard camera image from Bancroft Way, Berkeley, CA on the southern periphery of the UC Berkeley campus. The left and right lanes are blocked by delivery trucks, creating a bottleneck.

Table 5.2: Parameters defining truck specific general utility functions.

Truck #	p	(k^l, k^r)	x
1	2	$(4/15, 1/6)$	3
2	1	(0.2, 0.2)	9
3	1	$(9/140, 0.18)$	14
4	2	$(1/3, 0.5)$	4
5	3	(0.125, 0.25)	17
6	1	(0.2, 1)	10

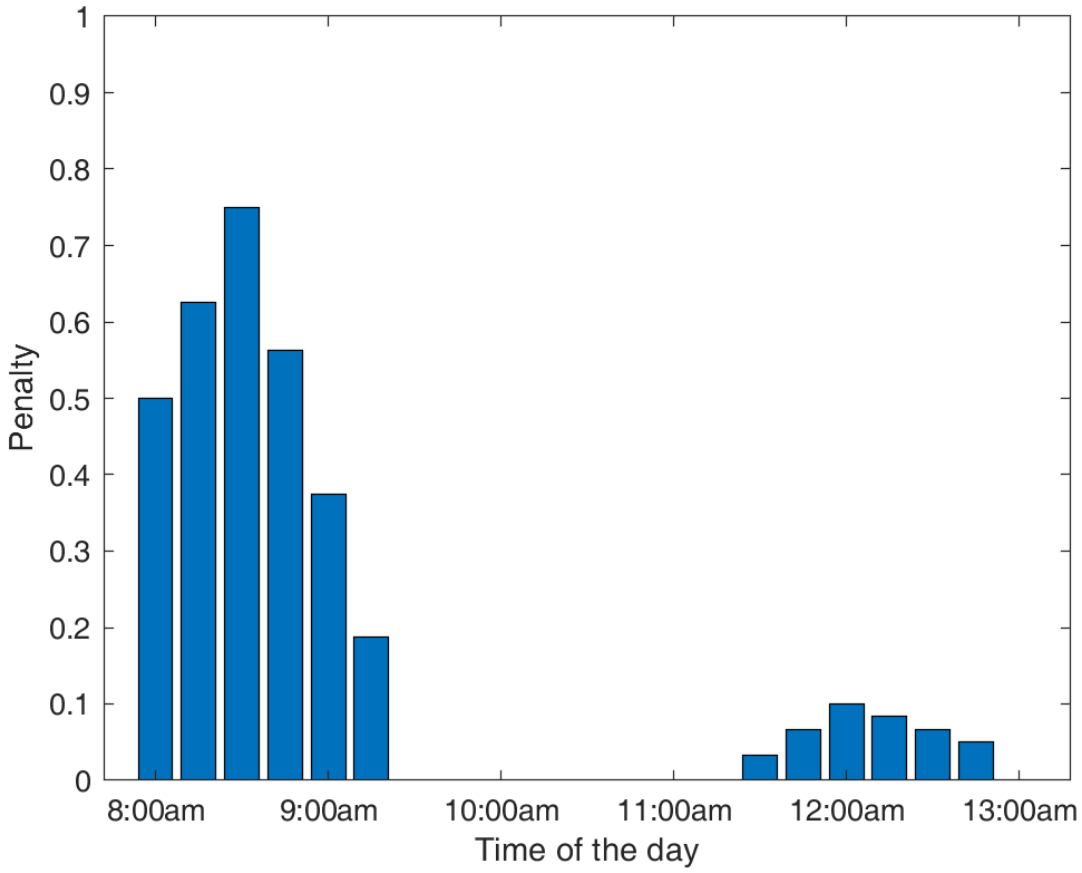


Figure 5.8: Bancroft Way city's penalty function. Morning and afternoon rush-hours are subject to a higher penalty for a delivery truck arrival time.

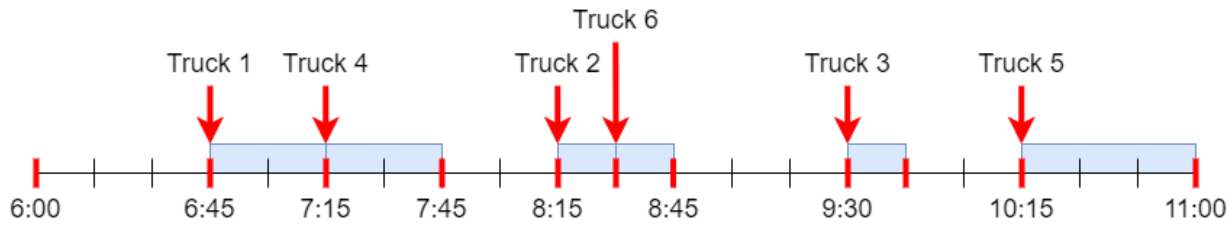


Figure 5.9: Bancroft Way optimal solution. Blue regions indicate time-slot utilization according to delivery vehicles' processing times.

Chapter 6

Curb Monitoring

6.1 Introduction

In the modern world with the development of transportation systems and improvements in logistics, commercial vehicles have earned their unique place in the transportation hierarchy. Taxi services, such as Uber and Lyft, making millions of trips daily contribute significantly to shaping the traffic states of networks. In the era of consumerism with growing production rates for all types of goods, ranging from food to high-tech components, the product distribution requires efficient transportation solutions involving fleets of freight trucks. With the increasing popularity of online shopping and internet ordering, the evolution of delivery services, such as Amazon, FedEx, UPS, DoorDash, GrubHub, etc., has never been more rapid. Being no different from ordinary cars, commercial vehicles require parking locations to be able to complete their tasks: dropping off passenger, delivering food/packages or unloading large items.

The current state of the parking system, however, is unable to keep up with the growing demand, resulting in parking deficit, and thus, in many local and global curb utilization inefficiencies, such as illegal parking, cruising in search of available spots, etc. In the absence of a rapid law enforcement system, commercial vehicles, due to the brief nature of their parking needs, may find it expedient to resort to double-parking or bus-lane occupation, causing traffic disturbance, congestion and hazardous situations for bus passengers. Various curb management models involving dynamic parking pricing ([80], [81]), parking reservation ([75], [76]), operation time partitioning (Chapter 5) have been developed to address the parking issue and improve curb utilization. However, the implementation of any control policy requires an extensive parking analysis to identify area-specific curb characteristics, spatiotemporal demand distributions, common congestion reasons and many other parameters. Therefore, the detailed and comprehensive traffic data collection and data analysis are essential for building a complete and informative representation of a transportation system.

Human-related data collection methods include surveys and questionnaires conducted by logistics companies' internal services or statistics-focused agencies. These methods can

provide companies' sensitivity towards various parking control policies ([89], [90]), common drivers' preferences on different delivery strategies ([91]), and individual agents' utilities for further curb management implementation. This approach, however, is subjective since the analysis of the obtained data is mostly related to structuring public opinions rather than deriving parking-related behaviour patterns. Therefore, human-related data collection must be accompanied by more substantial and precise monitoring methods to build a complete and comprehensive parking model of the network.

One of such methods, commonly used for curbside analysis involves static surveillance cameras installed on urban streets. The 24/7 video recording of a particular location enables an uninterrupted area monitoring regardless of weather conditions, operation time limitations, etc. Surveillance cameras often have a high video resolution which allows to capture distinctive vehicle characteristics, such as licence plate numbers, for the law enforcement purposes or vehicle tracking. Unfortunately, this approach does not scale well due to the limited monitoring area and a high maintenance cost for a single fixed camera. To discover the locations of interest worth monitoring, a large number of surveillance cameras must be installed around the city, which is both costly and effort-demanding.

To avoid the discussed issues while still being able to collect sufficiently detailed data, static camera approach can be replaced with a dashcam-related method. Having access to public (buses, taxis, commercial vehicles, etc.) and potentially private (personal cars) dashboard camera footage makes it possible to build an extensive traffic data-set comprised of thousands of hours of video recordings from multiple angles across a large region to identify parking trends, busy areas and curb usage patterns. The vehicle-point-of-view camera position allows to visualize and analyze the direct impact parking inefficiencies have on traffic agents. Although the completeness of the obtained data strongly depends on the penetration rate of probe vehicles, i.e. vehicles supplying dashcam footage, even infrequent monitoring of a particular area, as will be shown later, can be sufficient for a thorough analysis. In addition, this solution is more flexible, low-cost, and scalable, compared to static cameras, and thus is more preferable for a real-world implementation.

Several companies made attempts to develop a monitoring system for curb management purposes. One of these companies is Coord, who used augmented reality technology to capture the physical attributes of a scanned curb and make allocation suggestions for it. The proposed technology, however, was unable to either provide real-time data, or recognize moving objects, such as pedestrians and vehicles. Moreover, data collection is expected to be done manually via the mobile app, which can be problematic for large and dense areas. These limitations make the implementation of Coord's approach ineffective. Another company, Fehr and Peers, collaborated with various transportation authorities to study street-specific curb activity and provide analysis based on the data gathered. Their methods involved installing static video and photo cameras and manual processing of the obtained data (San Francisco Curb Study), which, as discussed earlier, do not scale well. In addition, Fehr and Peers relied on Uber's data regarding the pick-up and drop-off patterns of their vehicles, which has a limited application for the analysis of other commercial vehicles' behavior.

In this work we present a curb monitoring model based on a single-source dashcam footage

collection. To eliminate labor-demanding and inefficient manual data analysis, we developed a YOLOv5-based neural network (NN) for delivery vehicle recognition and classification. Further curb activity pattern identification is performed on the labeled and structured datasets provided by the NN. To demonstrate the performance of our approach in application to the real-world, we conducted a case study of the southern periphery of the UC Berkeley campus, Bancroft Way, Berkeley, CA.

6.2 Bancroft Way Case Study

To evaluate the performance of a curbside activity monitoring method it must be implemented in a busy street setting with high parking demand and a significant number of objects of interest (delivery vehicles). Bancroft Way on the southern side of UC Berkeley campus (Fig. 6.1) falls perfectly under this description. Many local business, university administrative buildings and an Amazon Drop-off location make Bancroft Way a popular destination for all sorts of delivery and freight trucks. The great parking demand together with the limited parking spot availability results in frequent incidents of parking violation, including double parking and bus lane occupation (Fig. 5.7). The absence of curb monitoring and managing systems makes it currently impossible to improve the parking situation.

To implement our model, we first need to establish the source of data collection that provides stable day-to-day curbside activity monitoring on Bancroft. In collaboration with the Berkeley Parking and Transportation Department we were given access to a Bear Transit Peripheral bus (Fig. 6.2) cruising around the campus from 7.30 am till 7.30 pm daily (except for weekends). The period of circulation is approximately 30 minutes, which might be a limiting factor for some types of activity analysis (e.g. parking duration identification). However, in the later sections we will demonstrate the sufficiency of the collected data for the recognition of many important behavioral patterns. The project was conducted over the course of 8 months, which resulted in over a thousand hours of dashcam video footage from around different points of campus peripheral, including several hundreds of hours of Bancroft Way recordings specifically.

First and foremost, to analyze the behavior of delivery trucks we must find a way to distinguish them from other types of vehicles on the dashcam recordings. Each video sample must be independently processed to identify whether it contains any valuable information. The amount of data produced by a single dashcam monitoring a small road region is extremely difficult and time-consuming to analyze manually. Increasing the number of active dashboard cameras and extending the application range of the approach would make the task impossible to perform in reasonable time manually. Therefore, vehicle recognition and classification must be automated to achieve the required performance level. In this project we focused on a neural network approach based on the YOLOv5 framework. According to the study [94], "You Only Look Once" (YOLO) object detection framework is reliable for identifying traffic objects even in congested traffic situations, which completely meets our requirements.

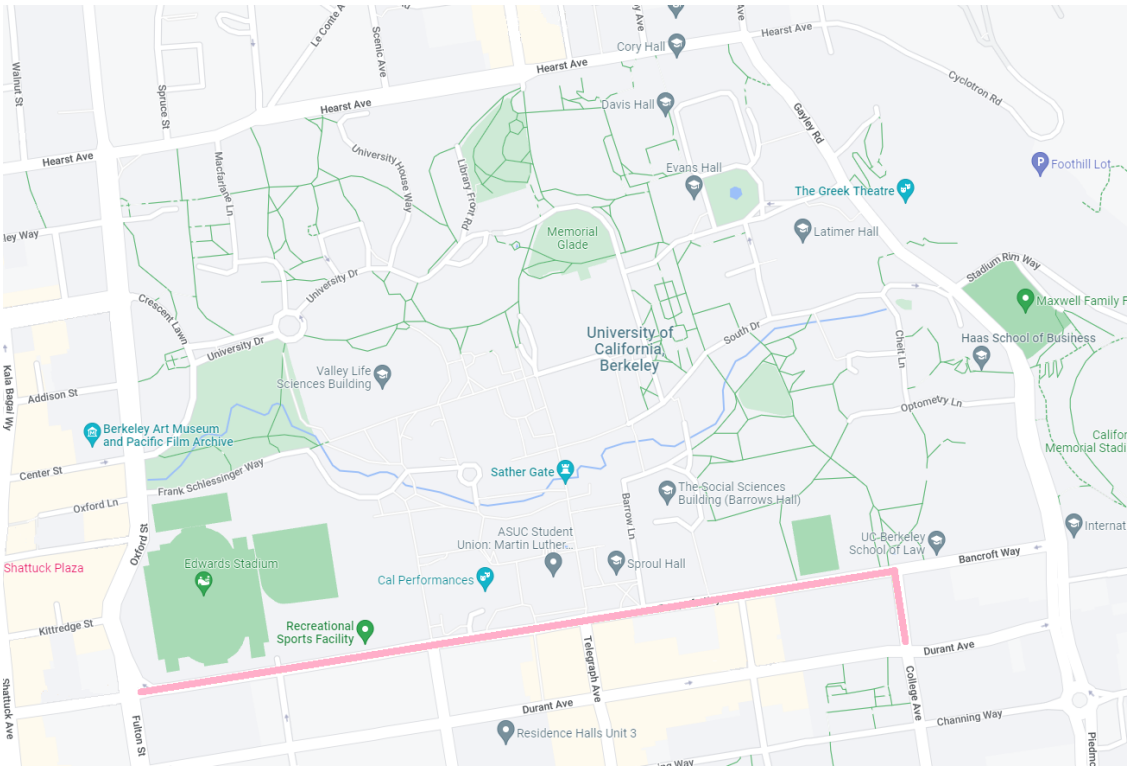


Figure 6.1: UC Berkeley campus map. The monitored segment of Bancroft Way on the southern side of the campus is highlighted pink.

Accurate classification of specific traffic objects requires an extensive neural network training and tuning. Being unable to find any pre-trained machine learning solutions for various types of delivery truck identification, we were forced to train our own neural network capable of classifying different vehicles. The accuracy of a NN training heavily depends on the size and quality of the training data-set. Large number of high resolution images with clear labeling is essential for any efficient machine learning models. Unfortunately, data-sets reflecting various types of delivery vehicles are not publicly available. All transportation-related data-sets, while identifying trucks as a separate vehicle type, do not further distinguish their affiliations with different companies. Therefore, for the training and validation purposes we created a labeled data-set consisting of images extracted from the collected dashcam footage.

The technical difficulty of building a training data-set corresponds to the lack of alternatives for the manual selection and labeling of relevant images. To find a video frame containing an object of interest, many video recordings must be reviewed and analyzed, which is extremely time-consuming. One effective shortcut that allows for a significant simplification of the process corresponds to filtering out video clips that do not contain any types of trucks. For that purpose, using a public Berkeley data-set, BDD100K ([95]), we

Figure 6.2: Bear Transit Perimeter bus cruising around UC Berkeley campus. Dashboard camera installed at the bus served as a single data collection source for the model.

trained another, more general, machine learning model capable of recognizing trucks and selecting corresponding footage for further manual review. Another benefit of this general neural network is the ability to pre-process and trim the raw dashcam footage before feeding it into our deliver vehicle classification model.

After training our YOLOv5 model on a custom training data-set, we were able to run the remaining dashcam footage through the NN, processing it frame by frame, to classify vehicles caught on the video. Each identified vehicle is outlined by a bounding box (Fig. 6.3) reflecting its relative location in the frame. The size and position of a bounding box are important parameters correlated with the distance to the object and its position on a street. In addition to the model-generated data, dashcam video clips provide the detailed metadata consisting of date, time, GPS coordinates and current speed of the vehicle carrying the dashcam (Bear Transit bus in our case). By combining the vehicle classification from the neural network with the camera-generated information, we were able to build a complete and comprehensive data-set of all delivery vehicle detections with the corresponding spatiotemporal characteristics and bus speed measurements for extensive curbside activity analysis. GPS coordinates also helped us to extract only Bancroft-related video clips through filtering.

Performance Metrics for Traffic Analysis

The first objective of our curb monitoring model is identifying "hot-spots", busy street regions with a high traffic density and a large number of delivery vehicle detections. These areas tend to be most problematic in terms of congestion and curb utilization due to high excessive parking demand and the resulting traffic flow disturbance. In case of limited monitoring and management resources hot-spots should have the highest control priority among

Figure 6.3: Vehicle detection and classification using YOLOv5 model. Bounding boxes include objects' labels and confidence levels. Dashcam provides additional metadata: date, time, GPS coordinates, and speed of the bus. Red vertical line is the reference point for the illegal bus lane occupation analysis. FedEx truck is located in the right-hand side of the frame suggesting bus lane parking.

the street locations to achieve the maximum traffic improvement. Using the generated classification data-set, more specifically spatial (GPS coordinates) and temporal (date and time) detection characteristics, it is possible to accurately determine the busy locations via data point clustering. In most cases hot-spots are temporary and can be easily recognised only within a limited period of time (rush hours, a grocery store's scheduled delivery times, etc.). On rare occasions, however, the discovered locations remain congested for the major part of the day, which might require a more detailed monitoring and thorough examination for an efficient regulation. Moreover, the hot-spot detection is essential for the curb management model, presented in Chapter 5, since it allows to distinguish a particular popular parking location to be reflected in the reservation app implementation.

Another component of the delivery vehicles' behavioral analysis is a trucks' arrival pattern recognition. Filtering by the model-generated labels, we can derive spatiotemporal detection distributions for each particular type of a delivery vehicle (Amazon, UPS, FedEx, etc.). Depending on the arrival tendency for different hours within a day or different days within a week, it is possible to predict the future demand distribution and anticipate potential congestion. Furthermore, arrival patterns specific for a particular company can provide valuable insights about their internal delivery policies, and help estimate individual driver's preferences for a more accurate utility function approximation and effective curb management. In addition, the detailed detection information allows us to identify relatively free periods of time with sparse arrivals and low parking demand, which can serve as the buffer zones in the process of redistributing delivery times for a more optimal curb utilization.

Another important deliverable of our model is the relative parking position of each individual delivery vehicle on the road (left curb, right curb). Of a particular interest is recognising whether or not a delivery truck occupies a bus lane or a bus stop. As mentioned earlier, bounding boxes constructed by the object classification model, are capable of providing the required estimations. Depending on the location of the box relative to the central vertical of the frame we were able to approximate the detected vehicle's position with respect to the bus. Since normally buses tend to utilize the special dedicated lane (usually rightmost lane), detecting a truck to the right of the bus almost certainly suggests the illegal bus lane occupation (Fig. 6.3). To verify the accuracy of the result, additional manual analysis can be applied. This information is valuable for analyzing illegal behavior on a company-to-company basis and monitoring their lenience towards such parking violations. Moreover, detecting bus lane occupation allows for a thorough investigation of the impact such parking choices have on traffic progression and buses' speed profiles. Furthermore, the violation recognition method can potentially produce special traffic reports to be transferred to the city's transportation police department. NN-based monitoring can be a powerful tool in the hands of the law enforcement system.

Lastly, we are interested in analysing the direct impact different parking behaviour has on the traffic flow. One of the metadata parameters provided by the dashboard camera is the instantaneous speed of the bus, which could serve as an accurate representation of the traffic progression. Unfortunately, the delay in dashcam speed measurements was too high, causing discrepancy with the ground truth. Therefore, we developed a different approach utilizing GPS coordinates. Instead of computing the current speed of the bus, we derived the average velocity within the specific road segments (Fig. 6.4). The breaking points for the segments were chosen considering the locations of the bus stops and the hot-spots. To avoid data corruption by long stopping delays, we kept the bus stops out of the segments; to capture the impact of the hot-spots on the velocity profiles we kept them in the segments. By combining the detection data with the average speed calculations, it is possible to analyze how the parking behavior of particular delivery vehicles affects the actual traffic propagation.

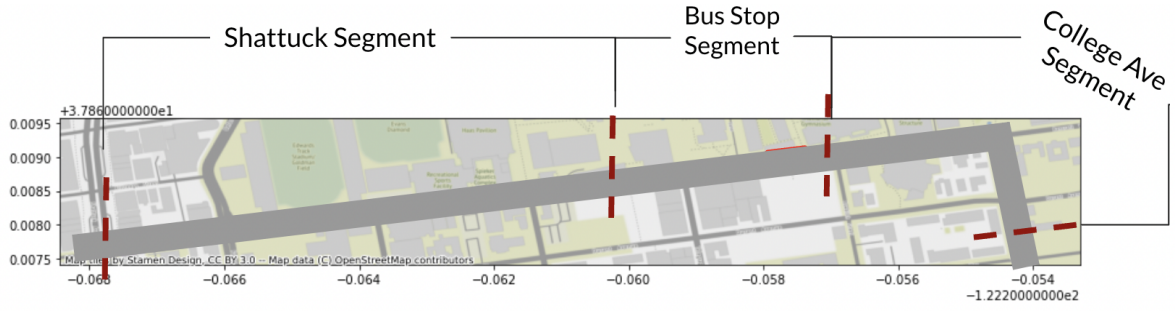


Figure 6.4: Bancroft Way segmentation for the average bus speed computation using GPS coordinates and travel time. Road regions are obtained based on the positions of the bus stops to avoid long stops within regions.

6.3 Data Analysis Findings and Conclusions

In this section we demonstrate the application of our curb monitoring model on Bancroft Way. The parking trends and patterns derived by the data analysis algorithms coincide with traffic observations.

Hot-spot detection

First, we focus on hot-spot identification based on the delivery trucks detection data. After creating the overall heatmap covering all the occurrences across all the considered delivery truck types, we identified the most busy location on Bancroft Way between the intersections with Bowditch Street and Barrow Lane (Fig. 6.5).

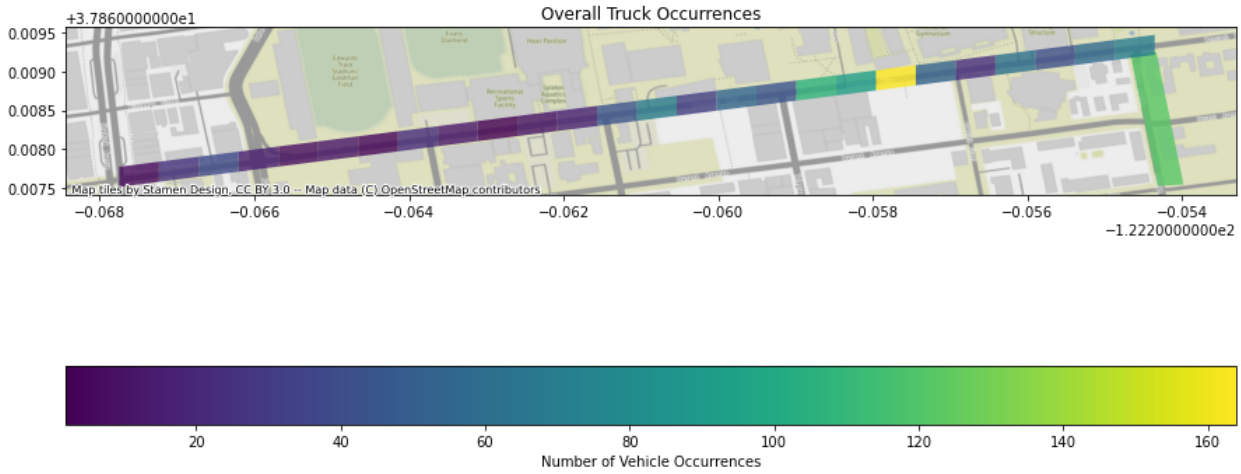


Figure 6.5: Heatmap of delivery vehicle detections on Bancrofw Way. The most dense area (hot-spot) is located between the intersections Bancroft-Telegraph and Bancroft-Bowditch.

This area is in close proximity to many local businesses and several university buildings, which all require frequent deliveries. A large number of available parking spots makes this location an easy and convenient parking destination (Fig. 6.6a). Another busy area is located down the Bancroft Way, between Barrow Lane and Telegraph Street. This is the closest parking street segment to the Amazon Drop-off location (Fig. 6.6b), which, in addition, has a convenient road pocket on the right-hand side of the road, allowing the parked vehicles to stay out of the traffic's way and avoid flow disturbance.

The third busy area is located around the intersection of Bancroft and College. Similar to the Bancroft and Telegraph hot-spot, this street location provides access to multiple campus buildings. Nearby cafes and housing also contribute to the high frequency of delivery vehicles detections by the bypassing Bear transit bus. All three locations were identified as problematic by bus drivers and the Transportation department, which is now supported by the analysis results.

(a) (b)

Figure 6.6: The Bancroft Way road segments between Bowditch Street and Telegraph Avenue identified as a hot-spot. Many local businesses, university buildings and the Amazon Drop-off location make this area a popular destination for delivery vehicles. (a) Bowditch - Barrow. (b) Barrow - Telegraph.

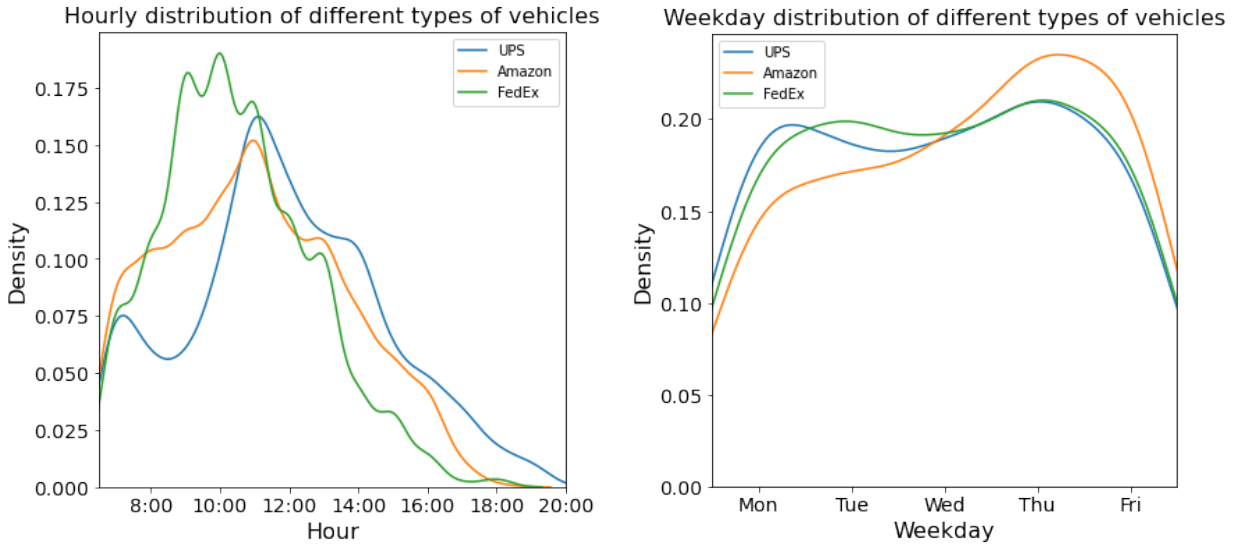
Delivery vehicle temporal distribution

The second important metric providing insights on different companies' delivery schedules and preferences is the temporal distribution of delivery vehicles detections. Based on the dashcam data analysis, we discovered several interesting hourly and daily delivery trends on Bancroft Way. First, we focused on the delivery vehicles' arrival patterns between different hours of a day. Since the Bear Transit bus has a fixed operation schedule (7.30am - 7.30pm), only that fraction of a day is considered in our analysis. According to the Fig. 6.7a, UPS and Amazon delivery trucks have similar detection distributions peaking at around 11am - 12pm and gradually decreasing towards the end of the day. FedEx vehicles, on the other hand, tend to complete their deliveries earlier in the morning, 9am - 11am, having the last delivery spike at around 11am.

For the day-to-day delivery trends across all considered delivery companies, based on the collected data compiled in Fig. 6.7b, both FedEx and UPS have somewhat uniform arrival distributions between different weekdays. A slightly different result is observed for Amazon vehicles. Approximately half of their weekly detections fall on Thursdays and Fridays. One possible reason for such uneven pattern could be related to the close proximity of the Amazon Drop-off location. To optimize the utilization of delivery trucks, Amazon might schedule the pick-ups of the returned packages from the Drop-off office on the last couple of days of the week, shift the arrival frequency accordingly.

Bus speed analysis

In order to analyze the impact delivery vehicles have on the average speed of the bus carrying the dashcam, we compare the velocity changes on each of the three segments of Bancroft with and without truck detections. Based on the dashcam data summarized in Fig.



(a) (b)

Figure 6.7: Temporal detection distribution for different types of delivery vehicles: Amazon, FedEx, and UPS. (a) Hourly distribution. (b) Weekday distribution.

6.8, we were not able to detect any significant slow-downs caused by delivery vehicles. The biggest difference we could measure does not exceed 3mph, which is insignificant. Further investigation (manual review of dashcam video clips) revealed that due to the infrastructure specifications of Bancroft (many crosswalks in busy areas), the delivery vehicles' impact on the average bus speed is overwhelmed by the delays caused by the pedestrian traffic. Although delivery trucks indeed slowed down the bus, deriving the numerical evidence is challenging at the current state of the project. To isolate the correlation between the parking patterns and the traffic propagation, an extended traffic analysis is required.

Bus lane occupation

The last component of our curb monitoring model is related to the bus lane occupation by delivery vehicles. As it was mentioned in the previous sections, this illegal parking behavior can potentially cause the most damage not only to the traffic state of the system, but also to the safety of bus passengers. According to the statistical breakdown from the collected data (Fig. 6.9), FedEx trucks are the most frequently detected delivery vehicles on the bus lane by a large margin.

It is important to note that the obtained outcome does not necessarily correspond to a large number of violations from delivery vehicles. Bus lane occupation estimation is probably the most uncertain algorithm considered in this study. First, having only one data source, which monitors each location for a brief period of time, we are unable to identify whether a particular vehicle is in motion or parked. Therefore, detecting a delivery truck on the right-

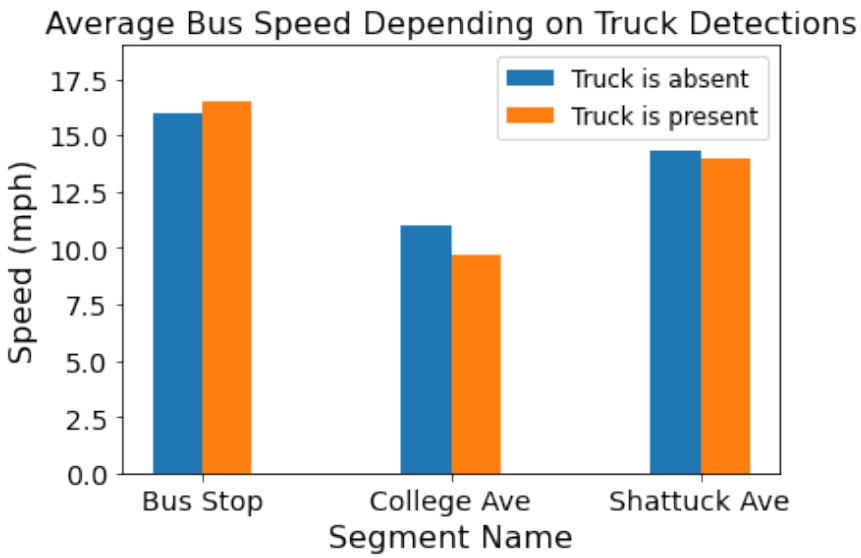


Figure 6.8: Impact of the delivery vehicle detection on the average bus speed. The analysis was conducted for three road segments: College Ave, Bus Stop, and Shattuck Ave.

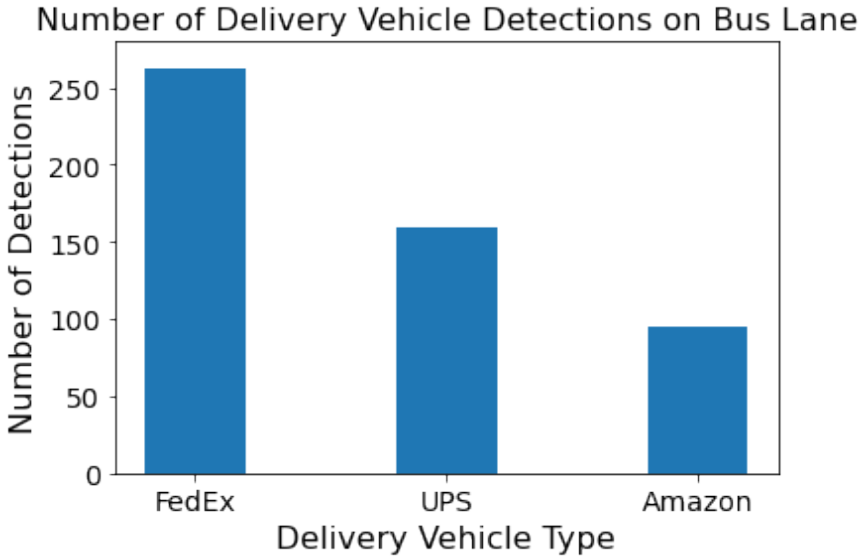


Figure 6.9: Bus lane occupation analysis for each delivery vehicle type: FedEx, Amazon, and UPS.

hand side of the video frame does not guarantee a parking violation. Second, the assumption that the bus travels only on the bus lane whenever it is available is quite restrictive and may cause some false-negative violation detections. For example, when the bus chooses the left-

most lane, any vehicle on the right would be considered occupying+ the bus lane, even if this is not the case. Lastly, not every road segment has a special dedicated bus lane (e.g. Fig. 6.6a), which makes the bus lane violation physically impossible at this location. Further model development is required to optimize the proposed analysis method.

Chapter 7

Conclusion and Future Directions

In this dissertation we presented various optimization methods and control algorithms for transportation system management and showed how communication protocols, navigation and reservation apps can be used to implement these models. The main focus of our work was on eliminating local and global inefficiencies caused by non-optimal resource utilization and infrastructure limitations. We first discussed the network-level control methods and their impact on a transportation system state. In Chapter 2 we showed how the Braess route elimination procedure applied to an extended graph representation of a network accounting for queues can affect the system's social delay when implemented in navigation apps. In order to improve local road utilization, we then turned our attention to the link-level management methods. In Chapter 3 we proposed the platoon formation algorithm capable of improving traffic progression on roads and reducing travel delay of connected vehicles. Focusing on traffic progression at intersections and fuel consumption, in Chapter 4 we developed the queue estimation algorithm based on the vehicle labeling procedure for the real-time phase length prediction and implementation in Speed Advisory Systems. Finally, to address the congestion issues related to curb utilization and double-parking, Chapters 5 and 6 discussed the parking reservation system for delivery vehicles based on operation time partitioning and the curb monitoring system utilizing dash cam footage for traffic analysis respectively.

The complete implementation and deployment of a Smart city that can successfully operate and effectively manage the transportation network still requires the development of numerous intelligent control systems. Considering the models presented in this dissertation, we now discuss several potential research topics extending our work.

Network Modeling and Congestion Estimation

Transportation modeling methods presented in Chapters 2 and 4 cover only a fraction of possible traffic inefficiencies and congestion scenarios. Eliminating a single inefficiency, such as Braess paradox, while reducing network delay, cannot guarantee the optiminality of the derived system. Extending our traffic management models to account for various congestion-related phenomena would significantly improve the traffic state of the system.

Furthermore, queue modeling and estimation methods discussed in our work did not consider either spatial or temporal queue propagation. For the former case, spillbacks are particularly interesting to explore and incorporate in the network modeling. Being caused by a severe congestion, spillbacks are capable of extending their impact far beyond the original lanes they formed at, which makes it challenging to build their mathematical representations.

For the temporal queue propagation, a promising research direction corresponds to the extension of our queue estimation model, presented in Chapter 4, to the over-saturated intersection scenario. The proposed labeling method must be accompanied by some additional flow prediction algorithm capable of keeping track of the traffic state at the intersection. Incorporating both under-saturated and over-saturated scenarios in a single unified estimation model can drastically simplify required calculations and provide a powerful tool for the Speed Advisory System's performance.

Implementation of Parking Management Techniques

Developing a complete and effective curb management system is a complex and multi-faceted task consisting of many layers. In Chapter 6 we presented a preliminary activity monitoring model that serves as a proof of concept for active data collection and analysis. Eliminating various limitations regarding the project implementation could help us to build a more accurate and broad curb monitoring system. First, the custom training data-set built for our machine learning model consists of the minimal number of labeled images required for a successful training. This leads to numerous delivery vehicle misclassifications when processing the data and can potentially corrupt the analysis. Training the neural network on a larger data-set will most definitely improve the accuracy and robustness of the model and reduce the number of incorrect classifications.

Another potential direction for improvement corresponds to increasing the number of the data collection sources (active dash cams) enabling a more detailed and frequent monitoring of a larger area. The improved granularity and reliability of the collected data would allow us to extend our machine learning models and extract additional features about the detected delivery vehicles. In particular, we are interested in identifying whether a delivery truck is parked or moving.

Having access to the combined video footage from several dash cams, we can build an intelligent vehicle tracking model capable of locking on each individual truck and deriving its specific parking patterns and preferences for further analysis. Increasing dash cam resolution would also contribute to a more accurate vehicle identification and potential parking enforcement.

The next step after building and perfecting the curb monitoring system is incorporating the obtained activity data into curb management models. In particular, the partitioning model for parking reservation from Chapter 5 requires the knowledge of the drivers' utility functions to derive the parking equilibrium. This information can be approximated from the detailed parking pattern analysis for various delivery vehicles and further tuned based on the system response to the applied parking policies. Demand distribution and delivery

vehicles' arrival schedules will allow the city planner to construct proper penalty and cost functions for an accurate representation of the parking state.

Bibliography

[1] M. Burov, C. Kizilkale, A. Kurzhanskiy, and M. Arcak, "Detecting braess routes: An algorithm accounting for queuing delays with an extended graph", in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 2125–2130.

[2] M. Burov, N. Mehr, S. Smith, A. Kurzhanskiy, and M. Arcak, "Platoon formation algorithm for minimizing travel time", in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–6.

[3] M. Burov, A. Kurzhanskiy, and M. Arcak, *Speed advisory system using real-time actuated traffic light phase length prediction*, 2021. eprint: `arXiv:2107.10372`.

[4] M. N. Smith, "The number of cars worldwide is set to double by 2040", *World Economic Forum*, Jun. 22, 2016. [Online]. Available: `https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040`.

[5] U. S. Department of Transportation; Federal Highway Administration; Office of Operations, "2018 urban congestion trends. improving operations, improving performance", Feb. 2019. [Online]. Available: `https://ops.fhwa.dot.gov/publications/fhwahop19026/index.htm`.

[6] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions", in *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times*, 2011, pp. 282–291.

[7] K. Su, J. Li, and H. Fu, "Smart city and the applications", in *2011 international conference on electronics, communications and control (ICECC)*, IEEE, 2011, pp. 1028–1031.

[8] D. Braess, "Über ein paradoxon aus der verkehrsplanung", *Unternehmensforschung*, vol. 12, no. 1, pp. 258–268, 1968.

[9] T. Roughgarden, "On the severity of Braess's paradox: Designing networks for selfish users is hard", *Journal of Computer and System Sciences*, vol. 72, no. 5, pp. 922–953, 2006.

[10] P. Varaiya, "Smart cars on smart roads: Problems of control", *IEEE Transactions on automatic control*, vol. 38, no. 2, pp. 195–207, 1993.

[11] D. Swaroop, "String stability of interconnected systems: An application to platooning in automated highway systems", Ph.D. dissertation, University of California, Berkeley, 1994.

[12] S. W. Smith, Y. Kim, J. Guanetti, A. A. Kurzhanskiy, M. Arcak, and F. Borrelli, "Balancing safety and traffic throughput in cooperative vehicle platooning", in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 2197–2202.

[13] S. W. Smith, Y. Kim, J. Guanetti, *et al.*, "Improving urban traffic throughput with vehicle platooning: Theory and experiments", *IEEE Access*, vol. 8, pp. 141 208–141 223, 2020.

[14] N. Wan, A. Vahidi, and A. Luckow, "Optimal speed advisory for connected vehicles in arterial roads and the impact on mixed traffic", *Transportation Research Part C: Emerging Technologies*, vol. 69, pp. 548–563, 2016.

[15] Y. Zhu, X. Ye, J. Chen, X. Yan, and T. Wang, "Impact of cruising for parking on travel time of traffic flow", *Sustainability*, vol. 12, no. 8, p. 3079, 2020.

[16] J. D. Murchland, "Braess's paradox of traffic flow", *Transportation Research*, vol. 4, no. 4, pp. 391–394, 1970.

[17] L. J. Leblanc, "An algorithm for the discrete network design problem", *Transportation Science*, vol. 9, no. 3, pp. 183–199, 1975.

[18] C. Fisk, "More paradoxes in the equilibrium assignment problem", *Transportation Research Part B: Methodological*, vol. 13, no. 4, pp. 305–309, 1979.

[19] N. Stewart, "Equilibrium vs system-optimal flow: Some examples", *Transportation Research Part A: General*, vol. 14, no. 2, pp. 81–84, 1980.

[20] M. Frank, "The braess paradox", *Mathematical Programming*, vol. 20, no. 1, pp. 283–302, 1981.

[21] R. Steinberg and W. I. Zangwill, "The prevalence of Braess' paradox", *Transportation Science*, vol. 17, no. 3, pp. 301–318, 1983.

[22] S. Dafermos and A. Nagurney, "On some traffic equilibrium theory paradoxes", *Transportation Research Part B: Methodological*, vol. 18, no. 2, pp. 101–110, 1984.

[23] A. Taguchi, "Braess' paradox in a two-terminal transportation network", *Journal of the Operations Research Society of Japan*, vol. 25, no. 4, pp. 376–389, 1982.

[24] I. Milchtaich, "Network topology and the efficiency of equilibrium", *Games and Economic Behavior*, vol. 57, no. 2, pp. 321–346, 2006.

[25] P. Cenciarelli, D. Gorla, and I. Salvo, "Graph theoretic investigations on inefficiencies in network models", *arXiv preprint arXiv:1603.01983*, 2016.

[26] X. Chen, Z. Diao, and X. Hu, "Excluding braess's paradox in nonatomic selfish routing", in *International Symposium on Algorithmic Game Theory*, Springer, 2015, pp. 219–230.

[27] S. Fujishige, M. X. Goemans, T. Harks, B. Peis, and R. Zenklusen, "Matroids are immune to Braess' paradox", *Mathematics of Operations Research*, vol. 42, no. 3, pp. 745–761, 2017.

[28] Y. Ji, W. Mao, and X. Zhang, "Detecting braess paradox links with a mixed integer linear programme", *International Journal of Industrial and Systems Engineering*, vol. 17, no. 3, pp. 275–284, 2014.

[29] U. S. B. of Public Roads, *Traffic assignment manual for application with a large, high speed computer*. US Department of Commerce, Bureau of Public Roads, Office of Planning, 1964, vol. 2.

[30] R. Kucharski and A. Drabicki, "Estimating macroscopic volume delay functions with the traffic density derived from measured speeds and flows", *Journal of Advanced Transportation*, vol. 2017, 2017.

[31] M. Beckmann, C. B. McGuire, and C. B. Winsten, "Studies in the economics of transportation", Tech. Rep., 1956.

[32] J. Macfarlane, "When apps rule the road: The proliferation of navigation apps is causing traffic chaos. it's time to restore order", *IEEE Spectrum*, vol. 56, no. 10, pp. 22–27, 2019. DOI: 10.1109/MSPEC.2019.8847586.

[33] E.-Z. Madeleine, B. Dafflon, F. Gechter, and J.-M. Contet, "Vehicle platoon control with multi-configuration ability", *Procedia Computer Science*, vol. 9, pp. 1503–1512, 2012.

[34] H. Liu, X. Kan, S. E. Shladover, X.-Y. Lu, and R. E. Ferlis, "Impact of cooperative adaptive cruise control on multilane freeway merge capacity", *Journal of Intelligent Transportation Systems*, vol. 22, no. 3, pp. 263–275, 2018.

[35] M. Michaelian and F. Browand, "Field experiments demonstrate fuel savings for close-following", 2000.

[36] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa, "Overview of platooning systems", in *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

[37] C. Nowakowski, S. E. Shladover, X.-Y. Lu, D. Thompson, and A. Kailas, "Cooperative adaptive cruise control (cacc) for truck platooning: Operational concept alternatives", 2015.

[38] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons", *IEEE Transactions on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, 2000.

[39] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet", *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[40] M. Saeednia and M. Menendez, "Analysis of strategies for truck platooning: Hybrid strategy", *Transportation Research Record*, vol. 2547, no. 1, pp. 41–48, 2016.

[41] E. S. Kazerooni and J. Ploeg, "Interaction protocols for cooperative merging and lane reduction scenarios", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 1964–1970.

[42] H. H. Bengtsson, L. Chen, A. Voronov, and C. Englund, "Interaction protocol for highway platoon merge", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 1971–1976.

[43] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. L. Cigno, "Supporting platooning maneuvers through ivc: An initial protocol analysis for the join maneuver", in *2014 11th Annual conference on wireless on-demand network systems and services (WONS)*, IEEE, 2014, pp. 130–137.

[44] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control: Definitions and operating concepts", *Transportation Research Record*, vol. 2489, no. 1, pp. 145–152, 2015.

[45] E. Chan, "Overview of the sartre platooning project: Technology leadership brief", SAE Technical Paper, Tech. Rep., 2012.

[46] T.-S. Dao, C. M. Clark, and J. P. Huissoon, "Distributed platoon assignment and lane selection for traffic flow optimization", in *2008 IEEE Intelligent Vehicles Symposium*, IEEE, 2008, pp. 739–744.

[47] ——, "Optimized lane assignment using inter-vehicle communication", in *2007 IEEE Intelligent Vehicles Symposium*, IEEE, 2007, pp. 1217–1222.

[48] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi, "Signalguru: Leveraging mobile phones for collaborative traffic signal schedule advisory", in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011, pp. 127–140.

[49] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time", *IEEE transactions on control systems technology*, vol. 19, no. 3, pp. 707–714, 2010.

[50] C. Wang and S. Jiang, "Traffic signal phases' estimation by floating car data", in *2012 12th International Conference on ITS Telecommunications*, IEEE, 2012, pp. 568–573.

[51] V. Protschky, C. Ruhhammer, and S. Feit, "Learning traffic light parameters with floating car data", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 2438–2443.

[52] S. A. Fayazi, A. Vahidi, G. Mahler, and A. Winckler, "Traffic signal phase and timing estimation from low-frequency transit bus data", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 19–28, 2014.

[53] S. A. Fayazi and A. Vahidi, "Crowdsourcing phase and timing of pre-timed traffic signals in the presence of queues: Algorithms and back-end system architecture", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 870–881, 2015.

[54] V. Protschky, S. Feit, and C. Linnhoff-Popien, "Extensive traffic light prediction under real-world conditions", in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, IEEE, 2014, pp. 1–5.

[55] V. Protschky, K. Wiesner, and S. Feit, "Adaptive traffic light prediction via kalman filtering", in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, IEEE, 2014, pp. 151–157.

[56] S. Ibrahim, D. Kalathil, R. O. Sanchez, and P. Varaiya, "Estimating phase duration for spat messages", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2668–2676, 2018.

[57] J. Sun and L. Zhang, "Vehicle actuation based short-term traffic flow prediction model for signalized intersections", *Journal of Central South University*, vol. 19, no. 1, pp. 287–298, 2012.

[58] S. Chen and D. J. Sun, "An improved adaptive signal control method for isolated signalized intersection based on dynamic programming", *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 4–14, 2016.

[59] S. Coogan, C. Flores, and P. Varaiya, "Traffic predictive control from low-rank structure", *Transportation Research Part B: Methodological*, vol. 97, pp. 1–22, 2017.

[60] C. M. Day, D. M. Bullock, H. Li, *et al.*, "Performance measures for traffic signal systems: An outcome-oriented approach", Tech. Rep., 2014.

[61] C. Sun, X. Shen, and S. Moura, "Robust optimal eco-driving control with uncertain traffic signal timing", in *2018 annual American control conference (ACC)*, IEEE, 2018, pp. 5548–5553.

[62] G. Comert and M. Cetin, "Queue length estimation from probe vehicle location and the impacts of sample size", *European Journal of Operational Research*, vol. 197, no. 1, pp. 196–202, 2009.

[63] ——, "Analytical evaluation of the error in queue length estimation at traffic signals from probe vehicle data", *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 563–573, 2011.

[64] G. Comert, "Simple analytical models for estimating the queue lengths from probe vehicles at traffic signals", *Transportation Research Part B: Methodological*, vol. 55, pp. 59–74, 2013.

[65] K. Tiaprasert, Y. Zhang, X. B. Wang, and X. Zeng, "Queue length estimation using connected vehicle technology for adaptive signal control", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2129–2140, 2015.

[66] X. J. Ban, P. Hao, and Z. Sun, "Real time queue length estimation for signalized intersections using travel times from mobile sensors", *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1133–1156, 2011.

[67] M. Ramezani and N. Geroliminis, "Queue profile estimation in congested urban networks with probe data", *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 6, pp. 414–432, 2015.

[68] H. Zhang, H. X. Liu, P. Chen, G. Yu, and Y. Wang, "Cycle-based end of queue estimation at signalized intersections using low-penetration-rate vehicle trajectories", *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3257–3272, 2019.

[69] A. D. May, "Traffic flow theory-the traffic engineers challenge", *Proc. Inst. Traffic Eng*, pp. 290–303, 1965.

[70] A. Sharma, D. M. Bullock, and J. A. Bonneson, "Input–output and hybrid techniques for real-time prediction of delay and maximum queue length at signalized intersections", *Transportation Research Record*, vol. 2035, no. 1, pp. 69–80, 2007.

[71] G. Vigos, M. Papageorgiou, and Y. Wang, "Real-time estimation of vehicle-count within signalized links", *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 1, pp. 18–35, 2008.

[72] H. X. Liu, X. Wu, W. Ma, and H. Hu, "Real-time queue length estimation for congested signalized intersections", *Transportation research part C: emerging technologies*, vol. 17, no. 4, pp. 412–427, 2009.

[73] J.-Q. Li, K. Zhou, S. E. Shladover, and A. Skabardonis, "Estimating queue length under connected vehicle technology: Using probe vehicle, loop detector, and fused data", *Transportation research record*, vol. 2356, no. 1, pp. 17–22, 2013.

[74] M. R. Shahrbabaki, A. A. Safavi, M. Papageorgiou, and I. Papamichail, "A data fusion approach for real-time traffic state estimation in urban signalized links", *Transportation research part C: emerging technologies*, vol. 92, pp. 525–548, 2018.

[75] K. Inaba, M. Shibui, T. Naganawa, M. Ogiwara, and N. Yoshikai, "Intelligent parking reservation service on the internet", in *Proceedings 2001 Symposium on Applications and the Internet Workshops (Cat. No. 01PR0945)*, IEEE, 2001, pp. 159–164.

[76] H. Wang and W. He, "A reservation-based smart parking system", in *2011 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, IEEE, 2011, pp. 690–695.

[77] W. Qing-Feng and W. Qing-Gang, "Research on public parking reservation system based on sms", in *ICTIS 2011: Multimodal Approach to Sustained Transportation System Development: Information, Technology, Implementation*, 2011, pp. 636–643.

[78] G. Yan, W. Yang, D. B. Rawat, and S. Olariu, "Smartparking: A secure and intelligent parking system", *IEEE intelligent transportation systems magazine*, vol. 3, no. 1, pp. 18–30, 2011.

[79] S. Yang and S. Qian, "A non-sensor solution for effective and inexpensive parking management: Payment, reservation, and dynamic pricing", *Provisional Patent*, 2018.

[80] Z. S. Qian and R. Rajagopal, "Optimal parking pricing in general networks with provision of occupancy information", *Procedia-Social and Behavioral Sciences*, vol. 80, pp. 779–805, 2013.

[81] D. Mackowski, Y. Bai, and Y. Ouyang, "Parking space management via dynamic performance-based pricing", *Transportation Research Procedia*, vol. 7, pp. 170–191, 2015.

[82] C. Lei and Y. Ouyang, "Dynamic pricing and reservation for intelligent urban parking management", *Transportation Research Part C: Emerging Technologies*, vol. 77, pp. 226–244, 2017.

[83] SFMTA, "SFpark. Pilot project evaluation", *Project evaluation*, 2014.

[84] T. Fiez and L. Ratliff, "Data-driven spatio-temporal analysis of curbside parking demand: A case-study in seattle", *arXiv preprint arXiv:1712.01263*, 2017.

[85] A. Amer and J. Y. Chow, "A downtown on-street parking model with urban truck delivery behavior", *Transportation Research Part A: Policy and Practice*, vol. 102, pp. 51–67, 2017.

[86] R. Arnott and E. Inci, "An integrated model of downtown parking and traffic congestion", *Journal of Urban Economics*, vol. 60, no. 3, pp. 418–442, 2006.

[87] J. Gao and K. Ozbay, "Modeling double parking impacts on urban street", in *Transportation Research Board 95th Annual Meeting*, vol. 12, 2016.

[88] S. Iwan, K. Kijewska, B. G. Johansen, *et al.*, "Analysis of the environmental impacts of unloading bays based on cellular automata simulation", *Transportation Research Part D: Transport and Environment*, vol. 61, pp. 104–117, 2018.

[89] E. Marcucci, V. Gatta, and L. Scaccia, "Urban freight, parking and pricing policies: An evaluation from a transport providers' perspective", *Transportation Research Part A: Policy and Practice*, vol. 74, pp. 239–249, 2015.

[90] V. Gatta and E. Marcucci, "Behavioural implications of non-linear effects on urban freight transport policies: The case of retailers and transport providers in rome", *Case Studies on Transport Policy*, vol. 4, no. 1, pp. 22–28, 2016.

[91] L. dell'Olio, J. L. Moura, A. Ibeas, R. Cordera, and J. Holguin-Veras, "Receivers' willingness-to-adopt novel urban goods distribution practices", *Transportation Research Part A: Policy and Practice*, vol. 102, pp. 130–141, 2017.

[92] M. Nourinejad, A. Wenneman, K. N. Habib, and M. J. Roorda, "Truck parking in urban areas: Application of choice modelling within traffic microsimulation", *Transportation Research Part A: Policy and Practice*, vol. 64, pp. 54–64, 2014.

[93] G. D. Chiara, L. Cheah, C. L. Azevedo, and M. E. Ben-Akiva, "A policy-sensitive model of parking choice for commercial vehicles in urban areas", *Transportation Science*, vol. 54, no. 3, pp. 606–630, 2020.

[94] A. Sarda, S. Dixit, and A. Bhan, "Object detection for autonomous driving using yolo [you only look once] algorithm", in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, IEEE, 2021, pp. 1370–1374.

[95] F. Yu, H. Chen, X. Wang, *et al.*, "BDD100K: A diverse driving dataset for heterogeneous multitask learning", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.