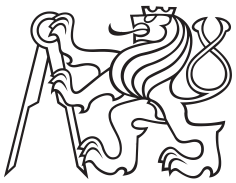
Master Thesis



Czech Technical University in Prague

F6

Faculty of Transportation Sciences
Department of Applied Mathematics

Local level routing to reduce travel times in urban networks

André Maia Pereira

Supervisor: Prof. Ing. Ondřej Přibyl, Ph.D.
Field of study: Technology in Transportation and Telecommunications
Subfield: Intelligent Transport Systems
November 2018

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Transportation Sciences
Dean's office
Konviktská 20, 110 00 Prague 1, Czech Republic

K611..... **Department of Applied Mathematics**

MASTER'S THESIS ASSIGNMENT

(PROJECT, WORK OF ART)

Student's name and surname (including degrees):

André Maia Pereira

Code of study programme code and study field of the student:

N 3710 – IS – Intelligent Transport Systems

Theme title (in Czech): **Snížení doby jízdy v městské dopravní síti pomocí lokálního směrování**

Theme title (in English): Local level routing to reduce travel times in urban networks

Guides for elaboration

During the elaboration of the master's thesis follow the outline below:

- Review the state-of-the-art on automated vehicle routing. Try to find existing local level routing approaches.
- Formulate the problem taking into account specifics of the MAVEN project (SPaT, lane-dependent queues, platooning, etc.).
- Propose an algorithm for local level routing solving the problem as stated in the previous section.
- Evaluate the proposed algorithm using SUMO micro-simulator.

Graphical work range: Will be specified by the supervisor

Accompanying report length: Will be specified by the supervisor

Bibliography:

K. Faez and M. Khanjary, "UTOSPF with waiting times for green light consideration," IEEE Int. Conf. Syst. Man Cybern., no. October, pp. 4170–4174, 2009.

J. Pan, I. S. Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic rerouting for lower travel time," IEEE Trans. Veh. Technol., vol. 62, no. 8, 2013.

Master's thesis supervisor: **prof. Ing. Ondřej Přibyl, Ph.D.**

Johan Olstam, M.Sc., Ph.D.

Date of master's thesis assignment: **March 22, 2017**
(date of the first assignment of this work, that has be minimum of 10 months before the deadline of the theses submission based on the standard duration of the study)

Date of master's thesis submission: **November 30, 2018**

a) date of first anticipated submission of the thesis based on the standard study duration and the recommended study time schedule
b) in case of postponing the submission of the thesis, next submission date results from the recommended time schedule

L. S.

v. r.

prof. RNDr. Miroslav Vlček, DrSc.
head of the Department of Applied Mathematics

doc. Ing. Pavel Hrubeš, Ph.D.
dean of the faculty

I confirm assumption of master's thesis assignment.

André Maia Pereira
Student's name and signature

Prague .....May 29, 2018

Acknowledgements

First of all, I would like to thank my supervisors, Ondřej Přibyl for his guidance and opportunities he offered to me in Prague, and Johan Olstam for his support in Norrköping where the initial ideas of this thesis started to be developed.

Next, I would like to dedicate this thesis to Petr and László for their endless support, motivation and understanding during my studies.

This work has been supported by the European Union thanks to project MAVEN (Managing Automated Vehicles Enhances Network), which is funded by the EC Horizon 2020 Research and Innovation Framework Programme, under the Grant Agreement No. 690727.

Declaration

I hereby submit for the evaluation and defence the master's thesis elaborated at the CTU in Prague, Faculty of Transportation Sciences.

I have no relevant reason against using this schoolwork in the sense of § 60 of Act No. 121/2000 Coll. on Copyright and Rights Related to Copyright and on Amendment to Certain Acts (the Copyright Act).

I declare I have accomplished my final thesis by myself and I have named all the sources used in accordance with the Guideline on ethical preparation of university final theses.

Prague, 30. November 2018

André Maia Pereira

Name and signature

Abstract

The thesis explores the concept of local level routing, where vehicles receive either travel times or route advice based on accurate short-term traffic predictions to reroute themselves through a network. We focus on developing a system to provide time-dependent road travel times considering information of signal timings, queue lengths, unexpected events and vehicle arrivals, in which traffic controllers model the movement of vehicles and share their knowledge about the traffic amongst themselves as a centralized or distributed system. Our motivation is to reduce travel times, avoid congestion and balance the load of vehicles throughout the network. In order to achieve this, we first analyse how similar route guidance systems estimate the cost of traversing edges and select optimal routes. Then, we study possible solutions for the optimal route problem, specially shortest-path algorithms, and we define the necessary inputs for the chosen algorithm type. After that, we research methods for estimation of discrete time-dependent travel times and decide to propose an event-based traffic theoretic model that models the driving behaviour of vehicles and deals with the restrictions of communication bandwidth between traffic controllers, the necessity of prediction, and short running time in larger networks, while feeding a deterministic queueing model that ensures estimations within a planning horizon. We describe the functionality of our proposed system and evaluate it through simulations of a real-world scenario for different penetration rates of vehicles able to reroute, comparing it against the possible best and worst cases as well as the case of vehicles using only hourly travel times. Finally, we discuss the performance and future work of the proposed local level routing system.

Keywords: local level routing, route guidance, travel time estimation, event-based traffic modeling, deterministic queueing model, automated vehicles, connected vehicles, urban traffic networks, SUMO simulation

Supervisor: Prof. Ing. Ondřej Přibyl, Ph.D.
Na Florenci 25, Praha 1, 110 00

Abstrakt

Diplomová práce se zabývá konceptem místního směrování, kdy vozidla dostávají pokyn buď podle kritéria cestovního času, nebo podle kritéria trasy na základě přesných krátkodobých dopravních podmínek tak, aby se přesměrovala skrz síť možných tras. Soustřeďujeme se na vývoj systému, který je založen na kritériu cestovního času při zohlednění časového intervalu dopravní signalizace, délky kolon, silničních uzavírek a času příjezdů, kdy kontrolní jednotky modelují pohyb vozidel a sdílejí informace o stavu dopravy mezi sebou ve formě centralizovaného nebo distribučního systému. Naším cílem je snížení cestovního času, odstranění dopravního přetížení a optimalizace proudu vozidel skrz síť možných tras. Abychom toho dosáhli, nejprve analyzujeme, jak systém vedení obdobných tras odhaduje náklady na průjezd trasou a jak vybírá optimální trasu. Následně studujeme možná řešení pro optimalizaci trasování, zejména algoritmy nejkratší cesty a dále definujeme nezbytné vstupy pro ten který typ algoritmu. Poté zkoumáme metody pro odhad diskrétního časového intervalu v závislosti na vybrané denní době cesty a vybíráme provozní teoretický model založený na událostech, který vytváří chování vozidel a zároveň zohledňuje omezení v komunikaci mezi řídícími jednotkami; potřebu predikce a časově krátkého zprocesování ve větších sítích. Navržený model je také zdrojem pro deterministický model kolon, který zajišťuje odhady v rámci plánovaného horizontu. Popisujeme funkcionalitu námi navrženého systému a hodnotíme ho na základě simulací odrážejících scénáře reálného provozu pro různé míry penetrace vozidel, která jsou schopná přesměrování a srovnáváme nejlepší a nejhorší možné případy spolu s případy vozidel používajících pouze jednotlivé hodinové intervaly. Na závěr diskutujeme dosažený výkon a očekávanou další práci na vývoji našeho systému místního směrování.

Klíčová slova: místní směrování, vedení trasy, odhad jízdního času, modelování provozu založeného na událostech, deterministický model čekání, automatizovaná vozidla, propojená vozidla, městské dopravní sítě, simulace SUMO

Překlad názvu: Snížení doby jízdy v městské dopravní síti pomocí lokálního směrování

Contents

Figures

Tables

Chapter 1

Introduction

In the European Union (EU), congestion costs more than 1% of its Gross Domestic Product (GDP), which is more than the EU budget. The transport sector is growing and one key issue is to increase the efficiency of its already existing infrastructure [European Commission, 2012]. Intelligent Transportation Systems (ITS) have already been transforming the transport sector, integrating information and telecommunication technologies with traffic engineering to alleviate traffic congestion, reduce emissions, improve safety, and provide reliable traffic management. Recently, Connected Vehicle (CV) technology and Cooperative Intelligent Transport Systems (C-ITS) represent an evolution of ITS in which road users and traffic managers share information that was previously not available in order to coordinate their actions [European Commission, 2016]. In the future, *Cooperative Automated Vehicles* (CAV) will transfer the driving functions from a human driver to a computer as well as cooperate with other road users and infrastructure to better plan and inform its decisions, aiming to improve mobility while increase safety and reduce negative environmental impacts ([Olia et al., 2016], [European Commission, 2016]).

In this cooperative environment, vehicles communicate through short-range and long-range communication channels once they are within certain range from other vehicles, so called Vehicle-to-Vehicle (V2V), or infrastructure, known as Vehicle-to-Infrastructure (V2I). Typically, the communication is between a vehicle's On-board Unit (OBU) and Roadside Units (RSUs), which are installed on the infrastructure (i.e. streets), allowing the exchange of messages in a fast, secure, and reliable way [Olia et al., 2016]. Vehicles send Cooperative Awareness Message (CAM) messages containing its state and parameters (actual speed, position, acceleration capability, etc.), while the infrastructure sends MAP messages with static road topology, Signal Phase and Timing (SPaT) for dynamic traffic light, (SRM, SSM) related to priority and preempted access of special vehicles, (PVD, PDM) when probe vehicle data, and (IVI) for in-vehicle information. Moreover, both may send Decentralized Environmental Notification Message (DENM) for dissemination of event-driven safety information [Festag, 2014]. On the top of that,

Traffic Controllers (TC), which are components of the infrastructure, may communicate among themselves while vehicles exchange messages between each other. When it comes to the benefits of such cooperative environment, authorities are interested, in particular, to the minimisation of the network's total travel time and congestion with minimum investment on new infrastructure, while vehicles aim to find their optimal route [Watling and van Vuren, 1993]. Therefore, efficient routing of vehicles from a specific source to certain destination is a needed aspect of cooperative automated vehicles, and route guidance may provide the desired congestion and travel time reduction [Schmitt and Jula, 2006].

Local level routing is a specific type of route planning and its concept is illustrated in Figure 1.1 considering a network with junctions connected by edges. The route choice of vehicles may use information of *vehicle arrivals*, corresponding to when and which vehicle will be arriving on the edge (including the ones already on the edge); *unexpected events*, e.g. representing edges not available to be used or with reduced speed; and *queue length* as well as *signal timings* (begin and duration of green phases) for calculating delays, given the junction's capacities of discharge. The cost of traversing each edge is estimated throughout a planning horizon and provided it in a way that users will either use it as they desire (i.e. the optimal route is responsibility of the user); or in the form of the optimal route advice for a specific vehicle.

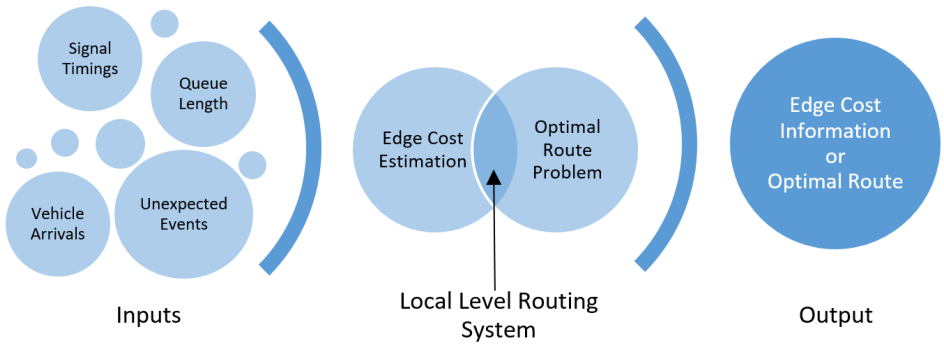


Figure 1.1: Example of inputs and outputs of a local level routing system.

A *local level routing system* is a type of route guidance system and a Traveller Information System (TIS). Originally, route guidance systems were invented to facilitate driving in unfamiliar places and compute the shortest routes in a static map without real-time traffic information. Nowadays, drivers also use route guidance systems to get information of real-time traffic conditions and suggestions on alternative routes to avoid congestion, road pricing, and parking availability [Liang and Wakahara, 2014]. In addition, by influencing the driver/vehicle decisions, these systems may affect the transportation demand [Herbert and Mili, 2008]. Under the new C-ITS environment, local level routing systems find optimal vehicle routes using short-term predictions based on local information exchanged between vehicles and infrastructure. Although it is recommended that the planning horizon

should be longer than the duration of trips [Krishnamoorthy, 2008], the focus is on accurate estimations as travel times in urban networks are more dependent on the signal timings and current traffic conditions.

1.1 Research Objectives

According to [Watling and van Vuren, 1993], dynamic route guidance must respond to variations of travel demand (e.g. peak and non-peak hours, incidents, etc.), and model the behaviour of drivers with and without routing information, in order to simulate the effects of such variations propagated throughout the network. This is very important since the benefits of route guidance systems are realised the most at these situations. Additionally, in a network composed by traffic controllers modelling and (possibly) controlling the traffic of a set of edges and their respective junctions, the scalability of the system (linked to the possible covered area, running time and planning horizon length) is essential [Schmitt and Jula, 2006]. However, the communication bandwidth between traffic controllers is limited, what requires a flexible system that can be either centralized or distributed according to the restrictions of the network. Therefore, the main objective of this thesis is to design, develop, and evaluate a local level routing system that uses as low as possible communication bandwidth of traffic controllers and models Cooperative Automated Vehicles (CAVs) and non-CAVs while use information of signal timings, queue lengths, vehicle arrivals, and unexpected events to provide routing information. In order to achieve this, the following list summarizes the research objectives.

1. **Optimal route algorithms and edge cost estimation models:** the first objective is to explore optimal route algorithms and choose a suitable and efficient one while research edge cost estimation models. Then, either select or develop edge cost estimation models that input the information available as well as estimate what is not available (e.g. non-CAVs data), and output in the format needed for the chosen optimal route algorithm.

2. **Local level routing system:** the second objective is to design a local level routing system using the chosen edge cost estimation models and optimal route algorithm, while developing efficient algorithms for each sub-system.

3. **Evaluation in a real-world scenario:** the third objective is to evaluate the proposed system's performance in a real-world scenario and compare it against suitable cases, besides defining the next research directions to achieve a system capable of real-world operation.

1.2 Thesis Structure

The thesis is organised as follows. Chapter 2 presents an overview of related work reviewing the state-of-the-art of route guidance systems that have characteristics of a local level routing system. Chapter 3 explores the algorithms for the optimal route problem, in which we narrowed the research to the shortest-path problem, and define the type as well as the input necessary for the chosen algorithm. Chapter 4 discuss the queueing and traffic theoretic models found in the literature that use the available information for the estimation of travel times. In Chapter 5, we present and describe the proposed local level routing system. Chapter 6 presents the simulation experiment we set to evaluate our proposed system and also the performance results. Finally, Chapter 7 concludes the thesis with a summary of achievements and outlooks for future research.

Chapter 2

Related Work

Throughout this chapter we will review already proposed systems that deal with route guidance of vehicles offering certain planning horizon considering real-time and/or estimated information. Most of the literature that focus on the problem of routing vehicles considering traffic lights aims to propose an integrated solution for both routing and signal control problems, in other words, the chosen routes influences the signal control and the signal plans the suggested routes. Although this control loop is usually desired, the common and current practice is that those problems are often solved separately without interaction between each other, specially due technological limitations, e.g. minimum area of application and penetration rate, needed communication bandwidth and computational power, what may turn such integrated solutions infeasible at the moment. Therefore, some authors approach this integration by developing systems that are able to route vehicles given short-term predictions of traffic signal plans (or using historical plans) without the input of vehicle routes into the traffic controller, in order to achieve a more realistic routing under the constraints aforementioned.

The term local level routing (also called *micro-routing* in [Leistner et al., 2012]) is not common in the literature. Usually, *route guidance* leads to the local level routing concept. However, route guidance represents any system that guide drivers from its origin to destination through a roadway network [Schmitt and Jula, 2006], while local level routing focus only on the part of adjusting (if needed) a previous defined route due to the dynamism of urban networks. Therefore, a local level routing system is a *dynamic guidance*, which can be categorized, according to [Khanjary and Hashemi, 2012], [Schmitt and Jula, 2006] and [Herbert and Mili, 2008], based on its architecture, information provision, decision-maker, and planning horizon, as illustrated in Figure 2.1.

The **information provision** defines which information is provided for route decision making. When *descriptive guidance*, vehicles receive information about traffic and/or road conditions with no routing advice (what may not result in network optimal results). On the other hand, *prescriptive guidance* aims to give optimal route advice but no information about traffic network conditions (if route compliance is 100% then it should lead to optimal

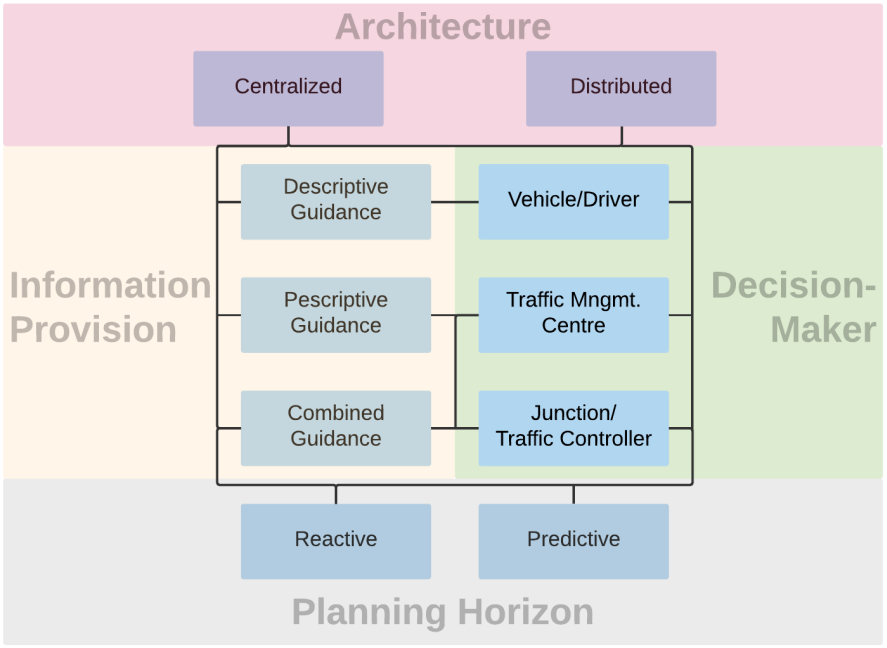


Figure 2.1: Classification of dynamic route guidance systems based on four major categories: architecture, information provision, decision-maker and planning horizon.

results). The *combined guidance* suggest routes to vehicles at the same time it is informed traffic conditions.

The **decision-maker** defines who is responsible for calculation of the optimal route. It can be either the *vehicle/driver*; *Traffic Controller (TC)* that model and controls the traffic on certain edges; or a *Traffic Management Centre (TMC)*, which monitors and coordinate the traffic for the whole network.

The **architecture** describes who is responsible for traffic data collection and provision. It is either: *centralized*, a TMC collects all the traffic data, compute the decision variables (e.g edge travel times), and provide the information for routing (robust and reliable, but computationally costly, complex and some individual users may experience longer travel times for network optimality); or *distributed*, in which each traffic controller collects local traffic data, compute its decision variables and share amongst themselves (when possible), as well as provide routing information for nearby vehicles (what may cause only local optimality and not a robust system on the network level).

The **planning horizon** is the anticipation degree of future conditions. *Reactive*, also known as feedback routing, is less complex and based only on current traffic conditions (few cases also time to green) in the network, but at the cost of greater vulnerability to congestion and incidents; while *predictive*, also called proactive routing, is based upon an iterative model that predicts

future conditions based on current and historical traffic information, what can offer high quality guidance if the traffic pattern is repetitive and the algorithm for such anticipation type (called anticipatory or hybrid) is able to handle unexpected events.

We should mention that other features of route guidance systems are also presented in the literature. [Schmitt and Jula, 2006] also differentiate systems between deterministic and stochastic, stating that while some deterministic approaches deals with uncertainties and use heuristics, the stochastic approaches use mathematical expectation of time invariant conditions (e.g. mean and variance of travel time), or the changing nature of the traffic with respect to time (e.g. mean travel time and variance as a function of the time of day). Although this distinction is important, we leave this as a characteristic of the approach used to solve the routing problem. Another example are the infrastructureless systems in [Khanjary and Hashemi, 2012], we don't dig into such systems because we consider local level routing as a cooperation between road infrastructure and road users.

Most of the approaches we will discuss provide *combined guidance* with at least small degree of prediction and the optimal route is decided by *junctions* in *distributed systems* or by a *Traffic Management Centre* (TMC) in *centralized systems*. Therefore, the main difference between each proposed system is how the authors approach the problem of defining the edge costs (e.g. the cost to transverse a road) and which algorithm is used to find the optimal route. We can classify the literature of such systems based on their observed features pattern (summarized in Table 2.1).

Classification	Author	Approach	Edge Cost Variables	Optimal Route
Centralized Systems	[Yamashita et al., 2005]	Greenshield V-K relationship with multi-agent simulation	Density	Not specified
	[Pan et al., 2012], [Pan et al., 2013]	Greenshield V-K relationship with macroscopic simulation and breadth first search	Density	Entropy balanced k-shortest paths
	[Yang et al., 2006], [Ximin, 2006]	Cellular automata with dynamic traffic assignment and (traffic lights) hybrid genetic algorithm	Flow	Simulation-based IOA
	[Liang and Wakahara, 2014]	Supply-demand model with / macroscopic simulation	Flow and detector occupancy	Dijkstra
	[Li et al., 2015]	Triangular Q-K relationship with Lagrangian-relaxation based optimisation	Flow and traffic light delay	K-least-cost paths
	[Ma et al., 2002], [Chen et al., 2006]	Non-convex control model with dynamic traffic assignment and optimal control	Free flow travel times, green times, cycle time, saturation flow	Dynamic traffic assignment
	[Rahman and Kaiser, 2016]	Fuzzy neural network	User preference, distance, traffic signal delay, road type and flow	Fuzzy neural network
	[Chen and Hu, 2012]	Bi-level framework user equilibrium with dynamic traffic assignment	Flow	Dynamic traffic assignment
Distributed Systems	[Khanjary et al., 2011]	Cellular automata with mesoscopic simulation and fuzzy logic	Street priority and density	Dijkstra
	[Tatomir and Rothkrantz, 2006]	Virtual "propagation" delay with ant colony	Average speed by sampled travel times	Ant colony
	[Faez and Khanjary, 2009]	Open shortest-path first protocol with Dijkstra and (traffic lights) finite-state machine	Average speed and next junctions time to green	Dijkstra
	[Chai et al., 2017]	Least expected time path and (traffic lights) modified max pressure control	Travel time and its probability, traffic light delay, driver's preference	Non-adaptive hyperpath shortest path
	[Kampen, 2015], [Taale et al., 2015]	Back-pressure control	Route saturation flow, edge capacity, travel time	Path size logit choice model
	[Leistner et al., 2012]	Queue model with microscopic simulation	Flow, traffic light delay, saturation flow	Minimum-time path
	[Lei and Ozguner, 1999]	Dynamic traffic assignment and optimal control	Queue length, flow, free flow travel time	Dynamic traffic assignment
	[Yang and Miller-Hooks, 2004]	Adaptive routing considering signal control with least expected time path, and (traffic lights) two-state continuous time Markov chain	Signal probability, measured travel time distribution	Adaptive hyperpath shortest path

Table 2.1: Classification and approaches of related work.

2.1 Centralized Systems

A centralized system collects data and centralize it into a server that process it, monitors and usually forecast the dynamics of the whole network as well as control (or just predict) the path planning of each vehicle. Although in well controlled situations these systems may allow network-wide optimal results, their main issues are the scalability, complexity and effectiveness under prediction with many stochastic factors (e.g. actuated or adaptive traffic lights, assumptions or vehicle/driver decision to follow route suggestions).

A proactive route guidance system is proposed in [Liang and Wakahara,

2014], using the rolling horizon approach (a planning horizon divided into discrete time intervals), at the beginning of each time interval the following three phases are conducted repeatedly: (1) detecting and predicting congestion, (2) selecting vehicles for rerouteing and computing alternative routes and (3) pushing route guidance to drivers. Two models for predicting the traffic amount on roads edge in urban traffic network are presented. This first model (spatio-temporal correlation) predicts the traffic amount that enters and leaves edge considering the split rate of traffic flows at the junctions, the ratio of green time over the edge travel time and the departure/arrival traffic amount on each edge, while the travel time is given by the average speed (based on loop detectors' occupancy and vehicle counts). The second model (spare road capacity) first estimates the maximum inflow (average speed over the sum of average vehicle length and minimum vehicle's gap) and outflow (same as the inflow but multiplied by the green time ratio) on each edge at each time interval, then the traffic flow is predicted using the inflow of the edge from upstream as well as their average traffic amount during certain time interval. A vehicle is set to be rerouted if it is before a certain number of upstream junctions of a congested or "will-be-congested" edge, or if it is intend to use this edge afterwards. After that, the optimal route is calculated through the Dijkstra shortest-path algorithm using current travel time on each edge.

[Li et al., 2015] solves the problem of optimal route guidance for vehicles with specific origins-destinations and preferred departure/arrival times, where a set of waiting times at signals in the suggested routes for each vehicle influences the flows and, therefore, the schedule and signal timings. The problem is decomposed into two less computationally complex subproblems (signal control and route guidance) through Lagrangian relaxation, coupled only by the flow balance constraints used in minimum-cost flow problems and least-cost path problems. The solution finds a time-dependent least-cost path for green times within a new time-dependent network, referred as "space-phase-time hypernetwork" which includes only allowable phase-time arcs, and maintains the linearity of both subproblems without loss of signal control or routing flexibility.

A hierarchical coordination system with a traffic signal control sub-system and route guidance sub-system is proposed in [Ma et al., 2002]. The optimal problems for every sub-system are firstly solved parallel and independently, then based on their results a coordinate solution is obtained by solving a correlative model, in which a road edge is divided into two parts: free-driving, calculated by the distance and speed; congested, defined by the Webster delay formula given the cycle time, green splits, and saturation flow. The route guidance sub-system uses a convex dynamic assignment model that finds optimal flow rates which will bring the shortest travel time for every vehicle under the restriction of flow conservation and FIFO (First-In-First-Out). The signal control sub-system use a optimal control optimization that minimizes the total delay time in all directions of a junction, giving the flow rate and green time for all directions. [Chen et al., 2006] later applied this approach

in a multi-agent system with nested hierarchical structure, which included 5 agents: traffic control and route guidance cooperative agent (CPA), TSC strategy level agent (TSC-SLA), TSC Executing level agent (TSC-ELA), DRG strategy level agent (DRG-SLA) and DRG Executing level agent (DRG-ELA). In their system, strategy level agents determines the cooperative strategies and cooperative coefficients, while the executive level agents must feed back the information about local road conditions and have capability to offer decision-making. The TSC-SLA update traffic control strategies and finds optimal control based on traffic flows, while the DRG-ELA adjusts routed vehicles routes based on real-time dynamic traffic conditions, and the DRG-SLA accomplishes dynamic route guidance.

The route guidance problem is also presented in a bi-level framework in [Chen and Hu, 2012], solving the problem of flow equilibrium for signal control and route choice (traffic assignment). The upper level computes the signal setting parameters given flow distributions (including cycle length and green splits) aiming to minimize total delay; the lower level solves the user equilibrium dynamic traffic assignment flows under resulting traffic control policies from the upper level. The flow distribution in the network is a function of signal control and route control, while the solution procedure defines the evolution of flow distributions under the interaction of signal and route control. To model the behaviour of traffic controllers under observed flows (e.g. if a junction has actuated traffic lights) the authors use uses an approximate macroscopic method that evaluates if a current green phase should be extended or not, given its maximum green time and the arrival times at stop line of vehicles at the end of the simulation time interval. One of the problems raised by the authors is the computational burden of the Dynamic Traffic Assignment.

An application in [Yamashita et al., 2005] is able to predict congestion and reroute vehicles to semi-optimal routes through macroscopic simulation. The model inputs vehicles current positions, destinations, as well as selected routes by an information server, while the network is represented by road edges between junctions that are divided into several blocks, each of them has length of the free flow speed during one simulation step. A prospective traffic volume is transmitted to drivers, this information is a product of the expected travel time (ETT) and total passage weight (TPW). The expected travel time (ETT) is calculated by the sum of the passage time that a vehicle will experience on each block; while for the total passage weight is the sum of a passage weight of each vehicle passing on the edge, which is the representation of the accuracy a vehicle will pass certain edge, calculate by the division of the number of next edges to transverse by the total number of edges to the destination.

Similarly to the passage weight idea, the authors in [Pan et al., 2012] and [Pan et al., 2013] presented several approaches that also use the Greenshield's V-K relationship. However, the common procedure is that the selection of vehicles to be rerouted is based on the breadth-first search (BFS) on

the inverted network graph, from the congested segments till the furthest distance (in number of segments) of vehicle can be away of the congested segment. In general, they estimate the total number of vehicles that have their path assigned passing certain segment in a time window; this number represents a decision variable for each approach: on the entropy-balanced kSP (EBkSP), the probability of vehicles on the segment and route vehicles to the lowest popular edges; the flow-balanced kSP (FBkSP) assigns a vehicle to the path that minimizes the impact of traffic flow; A* shortest path with repulsion (AR*) that consider both the travel time and the paths of the other vehicles (as a repulsive force) in the computation of the shortest path. The authors raised issues related to privacy, scalability, and low overhead for future research.

[Ximin, 2006] uses a hybrid genetic algorithm with cellular-automata simulation to estimate travel time and optimize signal timings. In their iterative simulation and assignment procedure, the traffic flow (and consequently travel time) is given by a Cell Transmission Model that includes a mixed-integer linear program; the signal settings are optimized by a hybrid genetic algorithm with delay minimization policy (the delay is the number of vehicles that will not be able to go to next cell), where it first optimizes globally and then locally using a pattern search function, while the decision variables are cycle, phase duration, and offset. The algorithm is based on the IOA-simulation, which updates alternatively the signal setting for fixed flows and solve the traffic equilibrium problem for fixed signal settings until the solutions of the two problems are considered to be mutually consistent.

A Fuzzy Neural Network (FNN) in presented in [Rahman and Kaiser, 2016], where estimates the path selection probability and the delay of each path while, for each origin-destination pair, it considers linguistic values from low to high of user preference, the distance, signal point delay, road type (highway or not) and traffic flow. The total route delay is calculated by the sum of: 1) straight path delay, which is the time needed to cover a specific distance at certain average speed; 2) cross section delay, the time that a vehicle takes to cross a crossing-point junction; and 3) signal point delay, the average time a vehicle will experience at each signalized junction. Specifically on the calculation of the average delays at junctions, the authors first calculate the delay time for a particular signalized junction as the sum of the average delays plus its variance; then the variance is estimated using a Taylor Series equation with the degree of saturation as its random variable, the average is given by a modified Webster Delay Model that accounts not only the delay due to uniform and random arrivals, but also an adjustment term empirically estimated by linear regression of observed traffic data. Based on the input features the FNN controller outputs the probability of each available routes between the source-destination pair, this is done via a learning process using a hybrid of back propagation and least-squares estimation that changes the input and output membership function parameters. One concern over this approach is that it uses one fixed reference of signal control parameters (e.g. cycle length, degree of saturation, etc.), what is not realistic given that each

junction may have different parameters and even be influenced by nearby junctions. However, this approach differs from the common research line and could be explored for improvements.

2.2 Distributed Systems

Distributed systems have independent traffic controllers that collect local information and process it with usually some degree of prediction. Usually, they only inform vehicles (and sometimes also other connected traffic controllers) about local traffic conditions. When route suggestion is addressed, the path is the outgoing edge from each junction or set of next road edges to reach a very near destination. Although such systems may avoid congestions to the next road edge (or within certain coverage area of connected traffic controllers), network-wide optimal results are likely to not be achieved, and the accuracy and range of prediction (if exists) influence their overall performance. On the other hand, due to its distributed nature, complex systems may scale for local subnetworks (traffic network of connected traffic controllers).

[Faez and Khanjary, 2009] improved their distributed dynamic route guidance system based on wireless sensor networks and the Open Shortest Path First Protocol (OSPF) introduced in [Faez and Khanjary, 2008]. The authors estimate each edge travel time based on the local speed measured by sensors, as well as the waiting time for green time according to the traffic load on edge and the time a vehicle will arrive at the traffic light and the signal state. The authors claim the main disadvantage of this system is that it needs many sensors and access points all over the traffic network to be deployed in order to detect vehicles and communicate.

Later, the same authors proposed in [Khanjary et al., 2011] a combination of distributed traffic signal control information and route guidance system, in which the route guidance estimates travel time for each edge, based on the average speed and information of green timings from hierarchical fuzzy signal controllers. Such controllers consider the priority of an edge (based on its buildings), the density and average speed of vehicles in each edge using a Unidirectional Selective Cellular Automata (USCA) algorithm, as well as the queue length of stopped vehicles behind junctions. Each traffic controller exchange information with others to find the optimal routes to all other junctions using Dijkstra's shortest-path algorithm.

A hierarchical routing system is introduced in [Tatomir and Rothkrantz, 2006], where a large network in divided into sectors that have junctions situated at the border between sectors and which have connection with other sectors. Each junction has a routing table containing a probability that expresses the goodness of choosing a junction as next junction for every possible final destination, and the one with highest probability is suggested to vehicles. Moreover, each sector has a virtual junction (with its routing table) for routing between sectors. The system consists of timetable updating system

(TUS) and route finding system (RFS). For the TUS, regularly, vehicles send their travel times along their path that is used to estimate the average speed. This average speed is used for the RFS that uses a Hierarchical Ant Based Control algorithm (H-ABC). One disadvantage of the proposal is that a delay due congestion during the route is distributed, and if only one edge is congested the sampled travel time could not detect it. The authors say this can be avoided if vehicles would send updated information at every junction, but the amount of data and communication required can be a problem and a compromise should be made. On the other hand, the distributed algorithm is able to route vehicles in complex networks and is highly robust against failures of distributed parts due to its hierarchical levelling system.

[Yang and Miller-Hooks, 2004] propose an efficient label-correcting algorithm called adaptive routing with signal control (ARSC), which determines the adaptive least expected time (LET) hyperpaths (acyclic subnetworks) in signalized stochastic time-varying (STV) networks, where actual signal timings (i.e. added delays) due signal operations are known deterministically or probabilistically. When approaching a junction, the driver decides the next junction given the traffic signal indication and the revealed arrival time according to hyperpath pointers for each departure time, which contains the delays due signal operations, from all origins to a given destination. The durations of edge available and unavailable time (i.e. red and green signals) for a given movement are assumed to be exponentially distributed, and a two-state continuous time Markov chain (CTMC) computes the probability of the availability of an outgoing edge for a given movement through a signalized junction for all departure times. Their adaptive approach avoids imprecise computations of the delay at the intermediate junctions because no assumption about the arrival times at intermediate locations is made until travel has been completed, what computed the correct durations of the expected hyperpath times. One minor issued related to this approach is that delays due to deceleration for a red signal or start-up and queue clearance lost times for a green signal are not explicitly considered, though can be modelled by extending the delay (or penalty) incurred during a red signal phase.

[Chai et al., 2017] presents a hyperpath-based stochastic shortest path dynamic routing guidance that can constantly update travellers' knowledge of edge travel time considering both current and historical information, as well as signal timings provided by some proposed adaptive signal control strategies. Using a Bayesian edge travel time updating scheme, each edge can have the delay for each signal phase, k possible costs and travel times where each of these are associated with a probability, while drivers preference is considered by giving weights to older and new information of travel times. In order to retrieve the shortest path to each next edge for any time step, every junction has two different vectors: one with the next junction to take at every time step; and another with the cost from current junction to destination at every time step. Their dynamic routing algorithm's computational complexity is a challenge, the authors suggest the route guidance per origin-destination and choosing the update interval as a trade-off between performance and

complexity.

A variation of the Max Pressure concept, called Back-Pressure, is explored in [Kampen, 2015] and [Taale et al., 2015]. The principle is used for route guidance, in which the task is to determine the ratio to direct traffic to following edges, being the route pressure based on how filled up the route is. A path size logit choice model calculates this ratio of routes based on the product of route pressure and route service rate (route saturation flow), considering also edge capacity and travel time. The system works by traffic controllers being fed by route guidance information to improve the estimate turn probabilities, and then those controllers decide their traffic signal phases that eventually will impact vehicle routes. The principle is not optimal for route guidance, being the main issues the definition of a representative route pressure, and the service rate for routes.

[Leistner et al., 2012] proposes an algorithm that gives route advices to vehicles at junctions where alternative routes begin. The first junction (where vehicles decide which route to take) estimate route travel times by gathering information about signal timings and based on a queuing model along each route. In addition, they also consider the previous advice to add vehicles that received advice into the proper traffic light queues and improve the queue prediction. The travel time is estimated by the free flow travel time along the route, and the estimated time instant the vehicle will be able to cross the stop line (service time), which is the delay (number of vehicles to be serviced before the considered vehicle over the saturation flow), the moment the traffic light gets green and the acceleration loss.

A combination of multi-destination routing and real-time traffic light control is introduced by [Lei and Ozguner, 1999], where every junction in the network is assigned a cost-to-go according to the queue length at the junction and the cost-to-go to all downstream junctions. The cost function to a single destination is the minimum sum of the delay due congestion and the free flow travel time. This gives the optimal flow entering every edge and queueing at junctions, based on the capacity of the edge and the cost on each edge to the same destination. This latter cost is the free flow travel on the edge plus the average cost of each junction to the destination, given available capacity under the optimal flow. One advantage of this approach is that average costs are associated to the topology of the network to the destination, so it accounts the cost of a vehicle being assigned to a edge, and the queues in the downstream junctions that will cause delay to the cars which will arrive at the junctions whatever their destinations.

Chapter 3

The Optimal Route Problem

In the previous chapter, Chapter 2, we have reviewed the state-of-the-art of route guidance systems which are similar to the desired local level system to be proposed. We also identified that there are two main characteristics that differentiate each system. One of them is the optimal route problem, which will be discussed on the remaining of this chapter. Although we have also seen in Chapter 2 that *Dynamic Traffic Assignment* (DTA) is a common approach for the optimal route problem, we will not focus on it as DTA is mainly used for solving both signal control and routing problems together, and due its required prescriptive guidance as well as high penetration rates of vehicles equipped with rerouteing devices. In addition, DTA and *simulation-based* approaches have issues related to running time, while *route-choice* requires knowledge (or estimation) of each vehicle's Origin-Destination (O-D) pair, what might be available. However, we discuss DTA and route choice in Appendix B.3. Approaches using *hyperpaths* are usually stochastic and we will not study them, we will later explain in this chapter the reason of ditching stochastic algorithms. On the other hand, and in this chapter, we will focus on the traditional way of finding the optimal route, the so called *shortest-path problem*, exploring possible solutions and their algorithms as well as the required inputs that will have to be provided by the edge cost estimation models, which will be discussed in Chapter 4.

The problem of finding the "shortest" path from certain source s to a destination d, (s, d) or O-D pair, on a directed graph G [Herbert and Mili, 2008] is illustrated in Figure 3.1.

A generic version of the labelled directed graph $G = (V, E, C)$ seen in Figure 3.1 can be interpreted as:

V : $\{s = v_1, v_2, ..., v_n = d\}$ the set of n junctions (V because it is usually called vertices or nodes) that represents the path decision points on the map of interest. These decision points can be intersections, highway entry and exit, or lane merge and split points.

E : $\{e_1, e_2, ..., e_m\}$ the set of m edges (commonly also arcs or links), where each edge can be denoted by its pair of junctions (v_j, v_{j+1}) representing

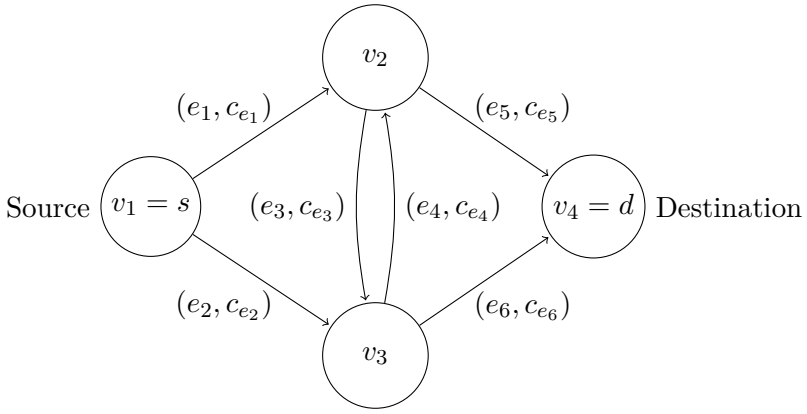


Figure 3.1: Example of a graph.

its source junction v_j and the direction junction v_{j+1}. Usually, an edge is a segment of road between two adjacent decision points, i.e. one-way streets and one direction of roads and motorways.

C : $\{c_{e_1}, c_{e_2}, ..., c_{e_m}\}$ is the cost (weight) to transverse the edge $e_k = (v_j, v_{j+1})$, being this cost time-varying or static. The cost of a path is the sum of the respective weight at arrival of each edge belonging to the path.

P : $\{(s, v_{j-1}), (v_{j-1}, v_j), ..., (v_{j+1}, d)\}$ the set of edges preceding a finite sequence of junctions that are visited at most once and belongs to the so called optimum path between the O-D pair (s, d), which represents the path with minimal sum of costs (weight).

L : $\{l_{v_j}, l_{v_{j+1}}, ..., l_n\}$ is the set of the minimum cost to visit each junction of the graph.

According to [Fu et al., 2006], most shortest-path algorithms follow a standard procedure, when it is assumed that the algorithm starts from the origin junction, seen in Algorithm 2 in the Appendix. Such algorithms label the minimum cost of visiting a junction and store the preceding edge which has this minimum accumulated cost. In reality, the *shortest-path* problem is a very broad problem, it has different classifications in the literature as well as applications, each solution for a specific type of shortest-path problem has its own characteristics. A complete taxonomy that classifies the various shortest-path problems into multiple high-level branches can be found in [Madkour et al., 2017], while Figure 3.2 shows the characteristics of shortest-path algorithms suitable for a local level routing system. There are three main types of solutions and after choosing one we must define either the data structure to use (in case of exact or heuristics algorithms), or the procedure (when meta-heuristics). After that, we narrow the possible algorithms as in Chapter 2 we have already defined the local level routing as a dynamic type of routing problem (routing guidance), in which the edge travel time

is commonly used as the cost (weight), usually one of the four situations according to [Fu, 2001]:

- constant edge travel time;
- time-dependent edge travel time;
- stochastic edge travel time; and
- stochastic time-dependent edge travel time.

The information of travel times can be obtained real-time, from historical data during a day, a week or a season, or even be predicted [Dong, 2011]. Based on this different travel times as inputs, and considering that unexpected events may occur in the network, we would need to look into stochastic, time dependent and dynamic algorithms.

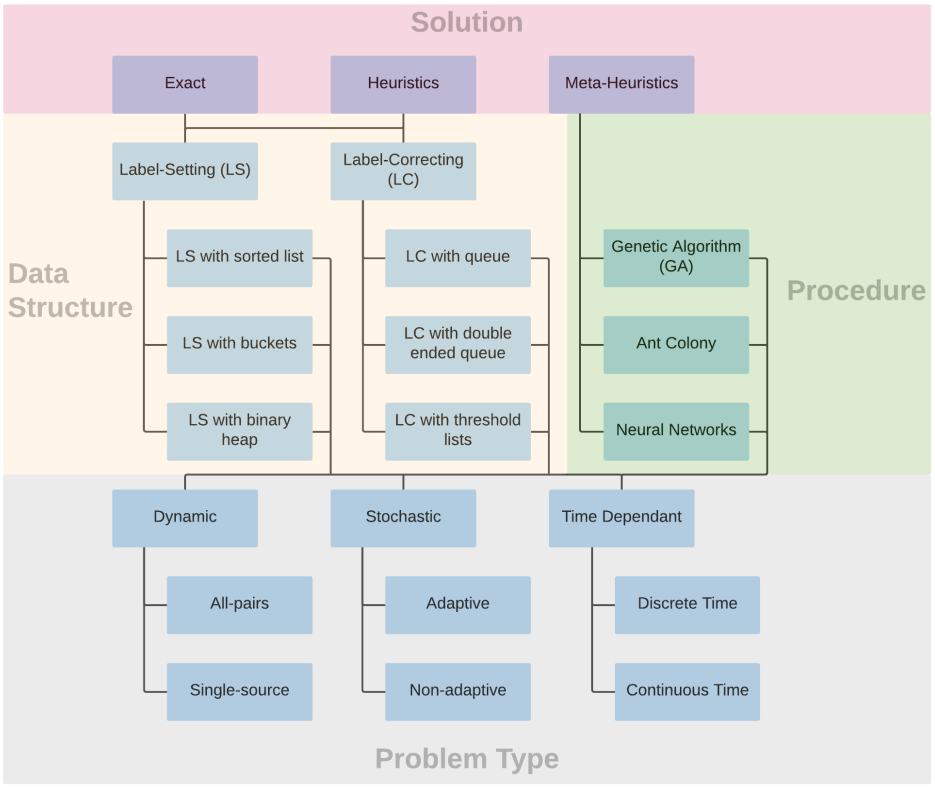


Figure 3.2: Characteristics of shortest-path algorithms for a local level routing system.

Exact algorithms search on the graph and find the shortest-path as long as the path exists, but the algorithm itself has an inherent long running time when it works [Dong, 2011][Madkour et al., 2017]. Yet, some techniques can be implemented to aid reducing the computational cost, like hierarchical maps [Herbert and Mili, 2008]. There are several algorithms belonging to this

category, the efficient Dijkstra algorithm [Dijkstra, 1959] (finding the shortest path between two junctions), and the Bellman-Ford algorithm [Bellman, 1955] (finding the shortest path from a single source to all other junctions) that is slower but allows negative edge's costs. When it is desired to find a k number of shortest paths between an O-D pair, k-shortest path algorithms are used [Eppstein, 1998]. **Heuristics** are specific for a particular problem. The algorithm finds a sub-optimal (approximate) solution within a short time by reducing the search space or decompose the search problem using any prior information ([Dong, 2011],[Madkour et al., 2017]). A widely used algorithm is the A* (goal-directed search) [Peter E. Hart et al., 1968], which is an extension of the Dijkstra's algorithm but limiting its search space using a heuristic function that has the effect of giving priority to junctions that are (supposedly) closer to the destination junction. The main differences between each exact or heuristics algorithms is which data structure is used to scan the eligible junction set and how junctions are identified and selected for examination. The following list classify exact algorithms into two main categories, according to [Fu et al., 2006].

- *Label-Setting (LS) algorithms* order the scan eligible junction set based on the current path costs from the source junction to the junctions in the junction set, and the junction with smallest label is selected for examination while concurrently identifying the shortest path to the junction. The algorithm may finish when the label of the destination junction is calculated, this is usually referred as *one-to-one* search mode (find shortest paths between two specific junctions) or *all-pairs* (when run for each O-D). Some variations of such algorithms are related to which data structure is used for the ordered scan eligible junction set: LS with sorted list; LS with buckets; and LS with binary heap.

- *Label-Correcting (LC) algorithms* uses a list structure to manage the scan eligible junction set that needs to be examined. Therefore, they cannot provide the shortest path between two junctions before the shortest path to every junction in the network is defined, what characterizes a *one-to-all* search mode (find shortest paths between the source and all other junctions). Some variations of such algorithms are related to the operation policy of the list structure: LC with queue; LC with double ended queue; and LC with threshold lists.

These exact and heuristics algorithms are useful only for the shortest-path problem (or for problems that can be modelled as the shortest path problem). Another category of algorithms that can also solve the optimal route problem is the **meta-heuristics**. Meta-heuristic algorithms have a generic way to solve the search problem and they are applicable not only for routing, but also on many different problems. Some examples are o the following list.

- *Genetic Algorithm (GA)* is a global optimization and search technique based on the principles of genetics and natural selection, in which a

population composed of many individuals (e.g. solutions) evolve under specified selection rules to a state that maximizes their fitness to the optimal solution [Mitchell, 1998].

- *Ant Colony* is a probabilistic technique where ants randomly look for food, and once they find it they return to their colony laying down pheromone trail that are likely to be followed by other ants and reinforced in case they find food [Dorigo and Stutzle, 2004].

- *Neural Networks* are inspired by the way biological nervous systems (e.g. synaptic connections that exist between the brain neurones), where a large number of highly interconnected neurones first learn examples (for instance an input pattern and output characteristics) and then adjust its configurations to predict the output based on the inputs [Gupta et al., 2003].

Dynamic algorithms allow edge updates (insertion or deletion of edges from the graph) and perform query operations (compute costs) efficiently in a real-time manner, being: *all-pairs* when besides handling insert/delete, and update operations, algorithms answer the total cost queries between any O-D pair on the graph; or *single-source*, in which algorithms compute the update and query operations for the total costs from the source junction to a given target junction. Another possibility are **stochastic** algorithms, they account the uncertainty of edges cost by modelling them as random variables and aim to find the shortest paths based on the least expected costs. Such algorithms can be: *adaptive*, if they determine only the best next junction at the each decision point (e.g. junction) of the shortest path, based on the edge costs at arrival on each edge along the path; or *non-adaptive*, when determining the whole shortest path at certain time instant. **Time-dependent** algorithms process graphs that have edges associated with a function that can be: *discrete time*, if assumes the edge weight functions defined over a finite discrete time window, what reduces the problem to computing shortest paths over a static network per time window (usually a time-expanded network); or *continuous time*, where the edge cost are defined by a continuous function (usually piecewise linear) [Madkour et al., 2017].

From this point, when we use the term cost (or weight) of an edge, we will refer to the travel time on the edge, what leads to the *minimum-time path problem* (also known as *fastest-path problem* or *earliest-arrival time problem*). This is because the time-dependent travel costs shortest path problem, even if travel times are static, is NP-hard [Dean, 2004] due to its similarity in nature to non-FIFO minimum-time path problems. This is also showed in [Batz and Sanders, 2012], where a multi-label A* search was applied to combine time-dependent travel time to time-invariant approximate costs proportional to driving distance (e.g. energy consumption, tolls, etc.). In addition, although including stochastic travel times may be desirable, this comes with the price of increased size of the exchanged messages between traffic controllers, more computations, and the need of assumptions as well prediction of probabilities

regarding traffic light states and vehicle travel times. Under adaptive and actuated signal control, signal timings would depend heavily on the prediction of vehicle arrivals, what could be complex to accurately forecast when signal control is not chained to the route of vehicles and when there is no total overview of traffic flows. Moreover, if As the information about signal timings and traffic is updated quite regularly as well as reliably predicted for some short time ahead, we don't need to include a stochastic element. Therefore, we will focus only on *deterministic time-dependent edge travel times* as edges' cost, but still considering the stochastic route choice into the calculation of travel times (i.e. turning probabilities or route choice given perceived travel times). This time-varying travel time applies to drivers at the time they arrive the edge. On this way we still capture some short-term time-varying random flows but using only the current expected signal timings provided by the traffic controllers.

The ability to give the solution within short time is fundamental for a route guidance application, being *discrete time* the chosen input for the shortest path algorithm in order to reduce the complexity [Dean, 2004]. However, these discrete models can miss the state of the network between any two discrete-time instants if they are too long, while error can be accumulated if the departure time from a junction (i.e. query time) is between two intervals not captured in the model [Demiryurek et al., 2011]. Moreover, the chosen algorithm must have the FIFO (First-In-First-Out) property, meaning that waiting at junctions is not allowed and moving objects exit from an edge in the same order they entered the edge. Vehicles are not expected to wait voluntarily at junctions as well as the time-dependent problem without this property is NP-hard in continuous time ([Dean, 2004], [Chen and Tang, 1998]), while FIFO efficient polynomial-time solution algorithms are available [Dean, 2004]. *Exact algorithms* are acceptable if a vehicle has already a complete network-wide route and just need to find the shortest path to the last junction of its route that is within certain range (i.e. sub-network of on-line data) from the source junction. However, the myopic approach of using exact algorithms in a sub-network may discard better alternative routes that have a different last junction outside the range (from the source junction) with on-line information. The use of *heuristics* is recommended only when running the algorithm in a large network, though it might lead to optimal routes.

3.1 Shortest Path in Discrete FIFO Time-Dependent Networks

The *time-dependent shortest path problem* considers that the network characteristics change over time (e.g. due recurrent and predicted non-recurrent congestion) and such changes are known in advance. It was first introduced in [Cooke and Halsey, 1966], in which the main assumption is that future travel time along each edge is a function of the departure time along an edge,

defined certain time before.

[Dean, 2004] studied theoretical properties and provided a simple framework of solution algorithms which unifies previous research on time-dependent shortest path problems. The author identified 16 different variants of this problem, and model them using arc arrival time functions for finding the *earliest arrival* from source junction s to a destination d, instead of arc travel time functions for least travel time usually seem in the literature. However, as most of the variants are similar or highly symmetric in time, he reduced these variants into 2 fundamental problems (using "wildcard notation" *):

- $EA_{s,*}(t)$: the earliest arrival from source junction s to all other junctions $v_{j+1} \in V \setminus s$ in the graph G departing at time (i.e., query time) t; and
- $EA_{s,*}(*)$: the earliest arrival from source junction s to all other junctions $v_{j+1} \in V \setminus s$ in the graph G for every possible departure time $T = \{0, 1, .., t_{max}\}$ (finite-length window of time from $t = 0$ until a planning horizon at $t = t_{max}$). Although this latter problem will not be analysed as our local level routing system will not have control of vehicle's departure time, a description of the algorithm can be found in Appendix A.2.

The inputs for the time-dependent shortest path problem are a directed network $G = (V, E)$, and a non-decreasing arrival time function $a_{v_j,v_{j+1}}(t) \geq t$ for every edge $(v_j, v_{j+1}) \in E$, while $n = |V|$ and $m = |E|$. The $a_{v_j,v_{j+1}}(t)$ corresponds to the time of arrival at v_{j+1} if a vehicle departs from v_j at time t (and consequently the travel time along the edge (v_j, v_{j+1}) can be found by $a_{v_j,v_{j+1}}(t) - t$). In addition, due to FIFO property, any vehicle departed from v_j after an arbitrary t_0 cannot arrive earlier at v_{j+1} than the one(s) departed at t_0. As outputs, $P(s, d)$ denote the set of paths between a source junction s and a destination junction d, and the function $EA_{s,d}(t) = \min\{a_p(t) : p \in P(s, d)\}$ gives the earliest arrival time at junction d if one leaves s at time t, where $a_p(t)$ is the path arrival time function of a path p.

3.1.1 Earliest Arrival for Fixed Departure Time

[Dean, 2004] states that the $EA_{s,*}(t)$ problem is quite similar to the static shortest path problem. He presents a pseudocode for a modified variant of label-setting (i.e. Dijkstra), seen in Algorithm 1, and another for a label-correcting static shortest path, found in the Appendix 3. Regarding the running times, for the label-setting algorithm, it depends on the implementation of the priority queue, S, being the best upper bound $O(m + n \log n)$ achieved when S is a Fibonacci heap; while the label-correcting algorithm depends on the implementation of the set Q, in which a polynomial running time of $O(mn)$ is achieved using FIFO queue. These modified shortest path algorithms have the same performance as the static algorithms, where

$p_{v_{j+1}}$: is the preceding edge $e_k = (v_j, v_{j+1})$ on the shortest path to junction v_{j+1},

$l_{v_{j+1}}$: is the cost, commonly referred as the "distance label", of visiting the junction v_{j+1},

c_{e_k} : is the cost to transverse edge k, where $e_k = (v_j, v_{j+1})$, and

S : is the scan eligible junction set which manages the junctions to be examined during the search procedure.

Algorithm 1 Pseudocode to solve the earliest arrival from source s to a destination d using a label setting (Dijkstra) algorithm [Dean, 2004].

Input: $t, V,\ E,\ a_{v_j,v_{j+1}}(t)\ \forall\ e_k = (v_j, v_{j+1}) \in E$
Output: $EA_{s,v_{j+1}}(t)$
  $EA_{s,v_j}(t) = \infty \quad \forall\ v_j \in V \setminus \{s\}$
  $EA_{s,s}(t) = t$
  $S = V$
  while $S \neq \emptyset$ **do**
    Select a junction $v_j \in S$ minimizing $EA_{s,v_j}(t)$
    $S = S \setminus \{v_j\}$
    for each junction v_{j+1} connected to v_j, where $e_k = (v_j, v_{j+1}) \in E$ **do**
      $EA_{s,v_{j+1}}(t) = \min\{EA_{s,v_{j+1}}(t), a_{v_j,v_{j+1}}(EA_{s,v_j}(t))\}$
    end for
  end while

3.1.2 Speed-up Techniques

According to [Dreyfus, 1969], the time-dependent fastest path problem (in FIFO networks) can be solved by modifying the Dijkstra's algorithm, in which we showed in Algorithm 1 a version presented in [Dean, 2004]. While this simple modified (time-dependent) Dijkstra visits all network junctions reachable from s in every direction until destination junction d is reached, a time-dependent A* algorithm (for instance, the one proposed in [Chabini and Lan, 2002]) can significantly reduce the number of junctions to be visited compared to Dijkstra. The idea is to use a monotonic (i.e. consistent) heuristic function $h(v_j)$ that directs the search towards the destination junction d. One important aspect is that $h(v_j)$ must be less than or equal to the actual distance between an intermediate junction v_j and d. A good estimation of the distance to d by the heuristic function efficiently drives the A* search, while if $h(v_j) = 0$ then the algorithm behaves exactly like Dijkstra's algorithm. In time-dependent road networks, a lower-bound estimative of $h(v_j)$ (that never overestimates the travel time) can be the Euclidean distance between v_j and d divided by the maximum speed among the edges in the entire network. However, this estimative is a very loose bound, what can lead to insignificant reduction of the search space. Although there are several other speed-up techniques in the literature (see Appendix A.3), we will not go deeper into this topic as A* is already a good option of heuristics and similar to the exact algorithm presented in Algorithm 1.

Chapter 4

Edge Cost Estimation

In Chapter 2 we reviewed several approaches of systems similar to the local level routing. Even though most of them share the same characteristics, they always differ in two aspects: how they solve the optimal route problem, which we explored in Chapter 3; and how they estimate the costs on each edge. For this latter problem, it is important to direct our research on possible solutions based on the available information, we don't need to forecast flow from upstream connected junctions, neither predict signal timings as they will be known. Thus, the main task is to explore ways how to predict edge travel time (as we defined in Chapter 3 that this will be the edge cost) along a planning horizon with the knowledge of the signal timings and queue length at junctions, in addition to traffic arrivals (i.e arrival profile and vehicles already on the edge). In this chapter we research how the travel time along an edge is modelled and why we will explore existing queueing models to support a new traffic theoretic model we will propose for modelling vehicles.

The edge travel time (valid within certain time interval) is the representation of the time spent by each vehicle travelling along an edge during certain time interval, and should be the input into the optimal route algorithm (i.e. shortest-path algorithm). Generally, travel time is a distribution that is often approximated by the arithmetic mean of the travel time of vehicles that traverse the edge during a given time interval [Krishnamoorthy, 2008]. On urban roads, vehicle arrivals at junctions are mainly in platoons, while due to traffic signals and their coordination vehicle travel time distribution tends to be gamma, log-normal (normal with positive skewness) ([Dong, 2011], [Liobaite and Khokhlov, 2016]), multi-modal (two or more distinct peaks) [Krishnamoorthy, 2008], or, for platoons, normal or binomial [FHWA, 2018]. As seen in Figure 4.1, the first peak of the distribution corresponds to vehicles that get minimal or no signal delay (no "cycle failure") and they don't need to slowdown due a queue in front of them (e.g. vehicles from upstream arriving after some other vehicles). At the second valley some vehicles can still cross without having to wait for another cycle but they got some delay due the dissipation of the queue at the time they arrive (e.g. first vehicles arriving from upstream), and the last peak the vehicles that will get red signal at the

time they arrive.

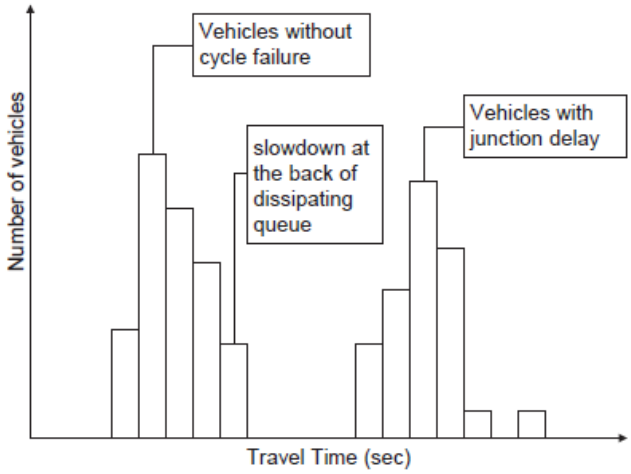


Figure 4.1: Idealised travel time distribution on an urban edge, adapted from [Krishnamoorthy, 2008].

The relationship between flow and travel times look as the curve in Figure 4.2, where congestion/queueing not only affects travel time but also traffic flow. [Srinivas Peeta et al., 2015] presented a dynamical system for a general time-dependent edge travel time function (referred as edge performance function), which can be found in Appendix B.

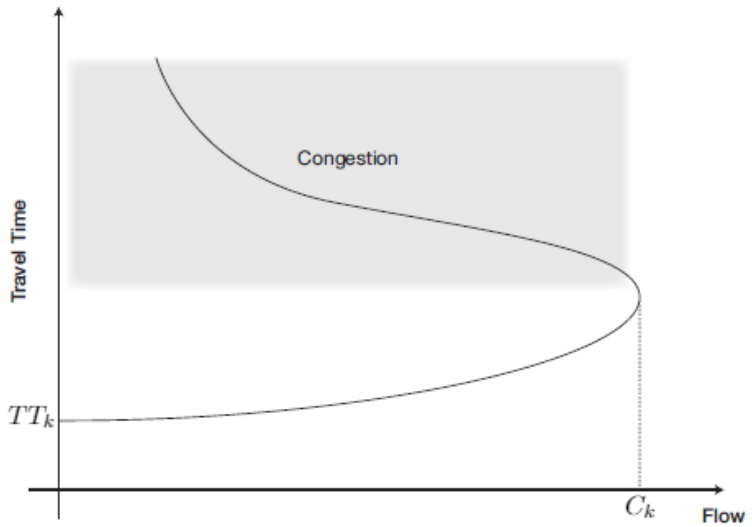


Figure 4.2: Relationship between edge travel time and edge flow, adapted from [Srinivas Peeta et al., 2015].

There are three different types of travel time data formats [Chrobok et al., 2000]:

- historical;

- current; and
- predictive.

Currently, it is common that route recommendations are based on **current travel times**. However, using current travel times might be effective due the fact that traffic conditions may change a lot since the collection of the informations until the time drivers actually arrive on the edge (though some stability of travel times are expected in short periods of time). Specially in urban roads where travel times depend not only of real measurements, but also the signal timings and the coordination of them between junctions [Krishnamoorthy, 2008]. Therefore, some anticipation of future events is necessary to provide **predictive travel times**, which are dependent on the capabilities of the system, route recommendation policy, driver response to the recommendations, and other unknown factors. Usually, at least at certain point, travel time prediction models combine **historical travel times** data with current and/or predicted travel times. As this combination (commonly) involves a large amount of data, recurring patterns can be predicted with good precision, but at cost of insensitivity to current traffic conditions [Watling and van Vuren, 1993]. In practice, future travel times can be estimated *a priori* with uncertainty, which is due in part to variations of driver behaviour, heterogeneous/mixed traffic environment and other random events, e.g. accidents [Yang and Miller-Hooks, 2004]. Travel time prediction models can be distinguished into two main approaches (statistical or analytical) according to [Wu et al., 2004], or classified by the conceptual foundation of their parameters [Treiber and Kesting, 2013], as illustrated in Figure 4.3.

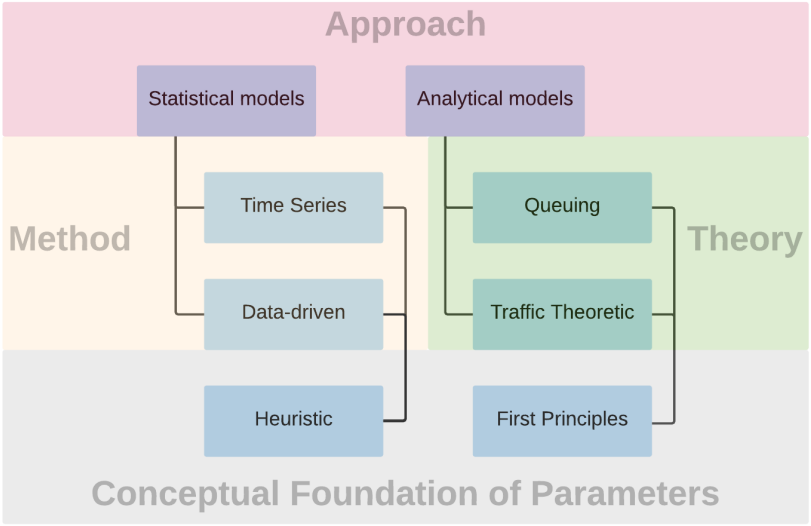


Figure 4.3: Classification of travel time prediction models.

Statistical models (inductive approaches in [Hoogendoorn and Bovy, 2001]) are based on the correlation between time-variant historical and current traffic variables such as travel times, speeds, and volumes as input. The

main idea of traffic forecasting in statistical models is that they can both reconstruct deterministic traffic motion and predict the random behaviours caused due unexpected factors by fitting real data. These methods can be classified, according to [Chen et al., 2014] into the following list.

- *Time series models*, in which the main classes are Kalman filter models [Ojeda et al., 2013] and auto-regressive integrated moving average (ARIMA) models [Ermagun and David Levinson, 2017].
- *Data-driven methods*, such as neural networks [Anderson and Bell, 1997], support vector regression (SVR) [Chun-Hsin Wu et al., 2003] and k-nearest neighbour (k-NN) models [Tak et al., 2014].

Analytical models (deductive approaches in [Hoogendoorn and Bovy, 2001]) predict travel times using physical laws, and usually require either dynamic O-D matrices or the junction's arrival profile and capacity to discharge as input. [Krishnamoorthy, 2008] categories travel time analytical models according to the two first items of the following list.

- *Traffic theoretic models*, where the predicted travel times are calculated based on the simulation of the chosen traffic flow model.
- *Queuing models* estimate delays and hence travel times for signalised junctions based on queue estimation and signal timings.
- [Hoogendoorn and Bovy, 2001] also includes the *intermediate approaches*, where mathematical model-structures are first developed which real data is then fitted.

[Treiber and Kesting, 2013] points out that one important factor related to the model is about its conceptual foundation of its parameters. **Heuristic models** are generally used when the model have no intuitive meaning about the form of an unknown function, and use simple mathematical statements (e.g. regression techniques), in which coefficients play the role of the model parameters by fitting data into it. On the other hand, parameters of **first principles** models are reflected by postulates, for instance, driver behaviour is determined by desired values of speed, acceleration, deceleration, time gap, and minimum gap.

As we have information of vehicle arrivals (at least statistically from upstream junctions), the estimated queue length and short-term signal timings, we don't need to rely on statistical models that are based on historical data. However, such models could be useful for the time-dependent travel times which are beyond our local level routing planning horizon, what is out of scope of this thesis. Therefore, we apply *analytical models* to specify the relationship between the information related to the traffic and signal timings with the travel time, based on *first principles* of traffic flow theory.

4.1 Traffic Theoretic Models

Traffic flow models aim to describe the behaviour of the complex traffic flow system by modelling the dynamics of vehicles and drivers in terms of mathematical equations, usually in the form of dynamical systems. As illustrated in Figure 4.4, mathematical equations and the theory of traffic flow define the model, by which, once calibrated, best fit with the traffic data and can be used to predict traffic flow or other applications (e.g. predict travel time) [Treiber and Kesting, 2013].

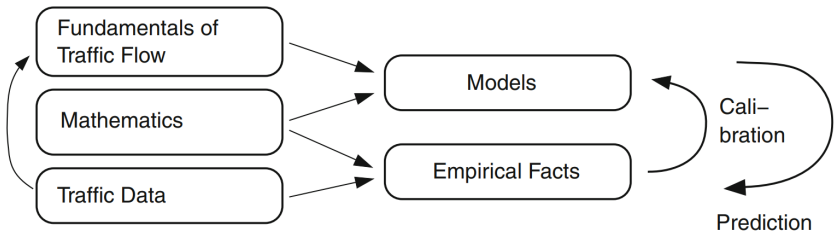


Figure 4.4: Traffic flow modelling [Treiber and Kesting, 2013].

The chosen theory of traffic flow needs to have sufficient description of the traffic depending on its application purpose. The following list presents how traffic models are classified according to [Hoogendoorn and Bovy, 2001] and [Treiber and Kesting, 2013].

- **Scale of the independent variables:** it is a natural classification based on the time-scale, space-scale, and state variables. *Discrete state variable models* describe specific traffic patterns to simplify models. *Continuous state variable models* describe most of the traffic patterns (density, flow, speed or occupation).
  - *Discrete space models* usually discretize the space into cells, where occupation and/or state variables, at every time step, of the cells is not only determined by its values at the previous time step but also state/occupation of the neighbouring vehicles or cells. The accuracy of the numerical solution is only restricted by numerical rounding errors.
  - *Discrete time models* creates discrete time steps, Δt (model parameter and sometimes discrete by events, i.e. when a vehicle enters of leave a cell), while the state variables (e.g. speed) usually remain continuous.
  - *Continuous space models* use space not divided into cells.
  - *Continuous time models* describe continuous changes over time of the traffic system's state, the mathematically exact solution is obtained in the limit $\Delta t \to 0$. When using *Partial Differential Equations* (PDE), both location x and time t are continuous and serve as the independent variables (e.g. local speed), while *Coupled*

Ordinary Differential Equations have continuous state variables (e.g. speed of a vehicle).

- **Representation of the processes:** models can be *deterministic*, in which same equation with same inputs will result on same results defined by exact relationships; or *stochastic*, where random elements (also known as *noise terms* or *stochastic term*) describe aspects of the traffic flow which are unknown, immeasurable, impossible to model.
- **Operationalisation:** *analytical* (also known as *traffic stream models*) represent static solutions given sets of equations describing pairwise relations between state variables (e.g. density, flow, speed or occupation) when the road conditions and traffic state remains the same on a segment. *Simulation models*, represent dynamic solutions when the traffic state and road conditions change during the time on a segment.
- **Scale of application:** it indicates the area that will be described by the model, e.g. a single junction, an arterial, an entire traffic network, etc;
- **Level of detail:** it considers the representation of the traffic flow characteristics [Ni, 2016].
  - *Picoscopic*, where individual drivers and vehicle-units are modelled considering how a driver operates his/her vehicle in the driving environment. The system is based on control theory and detailed representation of the interaction between each object in the model, consisting in a driver-vehicle-environment closed-loop control system.
  - *Microscopic*, in which the model describes space-time motion of vehicles as well as their individually interactions (chain of drivers' decisions). Roads and lanes are simplified into lines while vehicles behave as particles, and driver's longitudinal and lateral control are separate but simpler models.
  - *Mesoscopic*, where the behaviour of a group of drivers is specified based on an time evolution equation for the probability distribution of vehicle's position and velocity along the road considering microscopic values (e.g. acceleration, interaction between vehicles, lane-changing).
  - *Macroscopic*, traffic flow is characterized in term of flows and group of drivers, describing the dynamics of macroscopic variables (e.g. density, velocity, and flow) using partial differential equations generating the average state of road stretch.

Regarding the scale of the independent variables, in Chapter 3 we have defined that we need to estimate discrete time-dependent travel times in order to reduce the computational burden. Additionally, as these travel times are valid for edges, we should use both **discrete space and time models**.

The representation of the processes may be both deterministic or stochastic, though **deterministic** is more desirable to reduce the complexity of the model and aid to the stability of the system. Considering the fact that we need to provide some prediction along a planning horizon, and a deterministic model is the first choice, **analytical models** enable us to create longer planning horizon and more predicable results on an event-basis. The scale of application should support macro-scale (not only junctions or arterials) but should have quite higher level of detail as aggregated measures may fail to evaluate **microscopic** interaction between vehicles.

We need to explore solutions to calculate the edge travel time considering mainly the information of the inflow profile, queue length and signal timings. The arrival time of vehicles at a signalized junction plays a big role on defining the travel times due signal timings, what tell us that microscopic and picoscopic models are the first choice due the high level of detail. However, we are not that interested in so much detail provided by picoscopic, as they are more suitable to model the impacts of driver support system on the vehicle dynamics and driving behaviour [Hoogendoorn and Bovy, 2001]. The problem of microscopic models is that they are all simulation models. They need to be continuously updated, what in a considerable medium size network with many vehicles could be very slow as we need to provide some prediction time during the planning horizon, too complex when the goal is only the estimation of travel times, and require the information about queue length and vehicles going from one traffic controller area to another to be shared instantly. The remaining models (mesoscopic and macroscopic) seem also suitable. However, although macroscopic models are a well-studied topic that may aid the prediction of flows profiles, and discussed in Appendix B.1.1, their low level of detail may not provide enough information what we need. For instance, they fail to estimate delay due queue clearance, as it is a function of the vechicle's position in the queue and they focus on the average speed and density. Additionally, truly mesoscopic models are still under improvement, and many conventional mesoscopic models in fact share some microscopic and macroscopic characteristics at the same time [Ni, 2016]. The other option is the queueing models, which will be discussed in Section 4.2. They can model queue clearance delay and reduce the calculations per each traffic signal phase (compared to constant updates of simulation models). Therefore, considering that the information from Cooperative Automated Vehicles (CAVs), queue length estimation, and communication between TCs may enable us to have the knowledge of how many vehicles are at the junction at certain time and when certain amount will come in the future, we can use such information for a new traffic theoretic model to be proposed. The goal is to only estimate vehicle's position, speed and time at certain important events where it could produce delays. This would manage to avoid too many calculations and it would serve as an input for a queueing model to calculate delays due signal plans and the modelled traffic.

4.2 Queuing Models

Queuing theory is probably the most straightforward approach to model traffic dynamics. Due to the simplicity of queueing models, after the first work by [Webster, 1958], such models have been widely used for junction control and analysis. In queuing theory, a (virtual) queue of vehicles starts when the inflow to a bottleneck (e.g. red traffic light or non-protected left-turn) is larger than the bottleneck capacity. This can be represented in the following equation, according to ([Hoogendoorn and Knoop, 2012], [FHWA, 2018]):

$$dq_k = Q_k(t)dt - C_k(t)dt, \tag{4.1}$$

where q_k is the number of vehicles in the queue (while dq_k the change of the number of vehicles in the queue, $Q_k(t)$ the flow (inflow) in $vehicles/s$ into the bottleneck, $C_k(t)$ the capacity of the bottleneck (and also the outflow in $vehicles/s$ from it), for edge k at time t. One important aspect of it is that the capacity $C_k(t)$ is time dependent. While the inflow is given by the pattern of the arrival of vehicles, the capacity can be defined by the saturation flow (in $vehicles/s$) at junction. One disadvantage of the queuing theory is that the queues have no spatial dimension, and they do not have a proper length as they are counted in number of vehicles and not by the space they occupy [Hoogendoorn and Knoop, 2012]. This type of queue is called vertical queue, while the other type is the horizontal queue specifying certain length as distance, they are both illustrated in Figure 4.5.

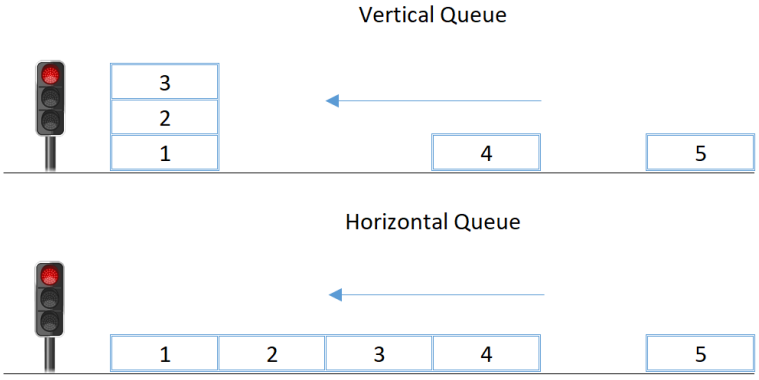


Figure 4.5: Queue types.

Then, the simplest way to define a travel time function TT_k using queuing models is applying the point-queue model, where the second term represents the time to discharge the entire queue $q_k = q_k(t)$ at time t [Srinivas Peeta et al., 2015]:

$$TT_k = TT_k^0 + \frac{q_k(t)}{C_k(t)}. \tag{4.2}$$

However, both the arrival $Q_k(t)$ and capacity $C_k(t)$ of the bottleneck have

some randomness process. For instance, some drivers have a shorter reaction time than others, leading to a shorter headway [Hoogendoorn and Knoop, 2012]. This leads to how queueing models compute delays at junctions, often having the first two components [FHWA, 2018], but also a third according to [Bureau of Public Roads, 1950] and illustrated in Figures 4.6 and 4.7, resulting in $d_k = d_k^u + d_k^o + d_k^q$ the following list.

- *deterministic* (or *uniform*), denoted d_k^u, founded on the fluid theory of traffic flow (which we describe in section B.1.1), where demand (inflow) and supply (capacity) flows are continuous variables that vary over time (periods in Figure 4.6), and its value is due the congestion caused by the flow to capacity ratio.

- *stochastic* (also *random*, *overflow* or *incremental*), denoted d_k^o, based on adaptations of the steady-state queuing theory, which accounts the distributions of the arrival rate and service time to reflect the random (non-uniform) properties of traffic flow (e.g. temporary cycle failures due to demand approaching supply).

- *initial queue*, denoted d_k^q, which considers the delay to all vehicles due to an initial queue within a period of time in analysis. This is because usually the calculation of delays accounts the signal plan beyond one cycle. As we are going to see later (and illustrated in Figure 4.6), the common practice it to divide the period into intervals of constant inflow and between such intervals there can be some remaining estimated queue if the inflow is higher than the capacity of the signal.

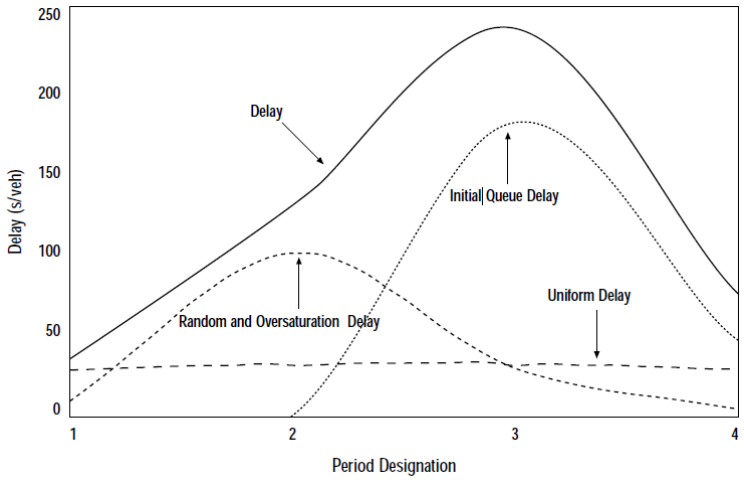


Figure 4.6: Delay model components for multiple-period analysis, adapted from [Bureau of Public Roads, 1950].

From now until the end of this section, we will not denote the subscript k when defining variables. This is due the fact that queueing models are specific to a particular junction (suppose junction $v_{j+1} \equiv j+1$) and hence the incoming

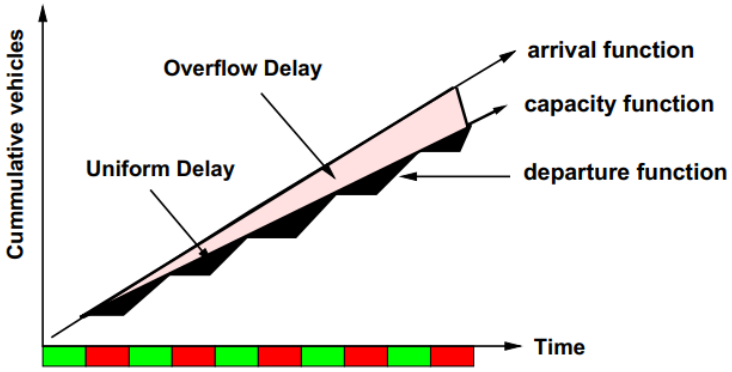


Figure 4.7: Functions of delay model, adapted from [Mathew, 2014].

edge to it (assuming $k = (j, j+1) \equiv e_k = (v_j, v_{j+1})$) is already implicit to the variables of queueing models. For instance, Figure 4.8 illustrates the process of queue development given uniform arrival and departure rates at a signalized junction, where the deterministic component of the cyclic delay d_k^u (the area under the queue profile diagram) can be easily estimated under the following assumptions.

1. a zero initial queue at the start of the green phase.
2. a uniform arrival pattern at the arrival flow rate $Q = Q_k(t)$ throughout the cycle c.
3. a uniform departure pattern at the saturation flow rate $S = S_k(t)$ (in $vehicles/s$) while a queue is present, and at the arrival rate when the queue vanishes.
4. arrivals do not exceed the signal capacity $C = C_k(t)$.

The effective green time g, seen in Figure 4.8, is the fraction of the green time during one cycle, c, (given by the green split $\boldsymbol{\lambda}$) where outflows achieve the saturation flow S. Usually, it is defined by the total green time minus an initial start-up/acceleration lost time (2-3 seconds) plus the clearance interval (2-4 seconds). The signal capacity,

$$C = S\left(\frac{g}{c}\right), \tag{4.3}$$

represents the amount of vehicles the signal can discharge during the effective green time (in $vehicles/s$), where g/c is the effective green to cycle ratio.

In reality, as the arrival and capacity can vary over time, while the arrival might be uniform, the queue development with some stochastic component can look like in Figure 4.9. The more the inflow reaches the capacity the more the likelihood of "cycle failures", in which vehicles are not able to discharge in one cycle and create the so called *overflow* queue of vehicles (initial queue for next cycle). We should notice that this initial queue is only

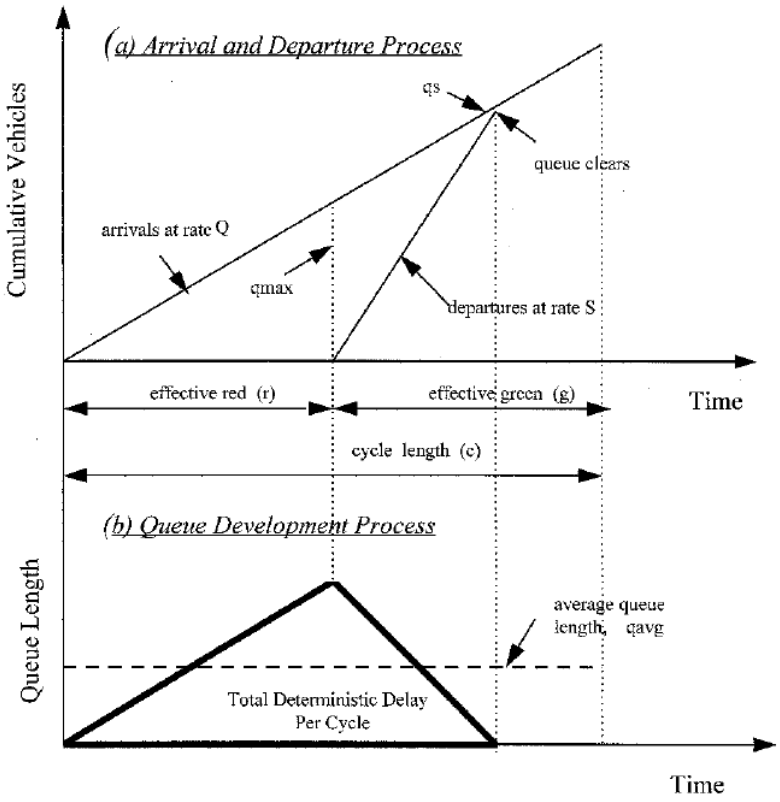


Figure 4.8: Deterministic component of queuing models with uniform arrival rate, adapted from [FHWA, 2018].

due a random phenomenon, depending on which cycle happens to experience higher-than-capacity flow rates, which causes an additional delay which must be considered. The following list presents how the modelling of the queue development varies with the queue model approach [FHWA, 2018].

- **Steady-state queuing models**: Delays are calculated based on statistical distributions of the arrival and departure processes, assuming that, after sufficient time, the state of the system is independent from its initial state and the elapsed time (stochastic equilibrium).
- **Time-dependent queuing models**: It is assumed stationary arrival and departure processes (which are not necessarily under stochastic equilibrium) that can be approximated by some mathematical function (step-function, parabolic, or triangular functions), and its average delay and queues are valid over the period of time in which the arrival are applicable.
- **Deterministic (oversaturation) models**: They model arrival and departure rates as a deterministic function of time, and calculate the average delay per arriving vehicle during certain time interval [Rouphail M and Akcelik, 1992].

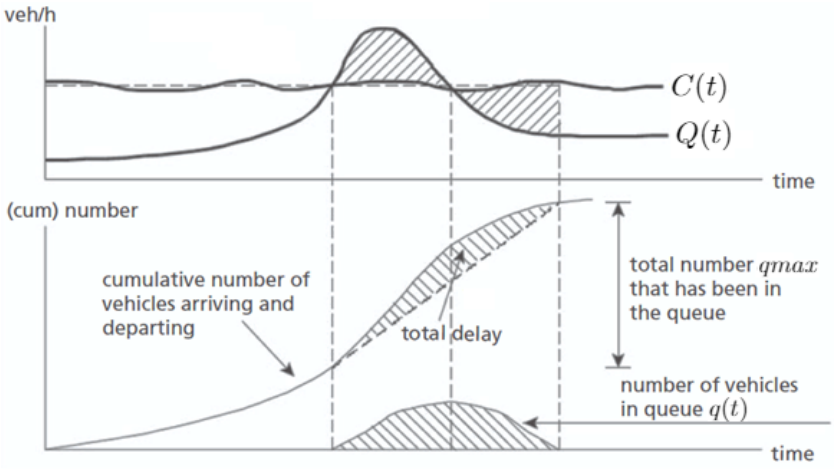


Figure 4.9: Total delay of queuing models with non-uniform arrival rate, adapted from [Hoogendoorn and Knoop, 2012].

The first issue with steady-state queuing models is that the equations assume fixed signal control and fixed signal capacity, what is not the situation in many cases of urban environments. The second issue is that once the average inflow Q exceeds the average capacity rate C, the stochastic equilibrium cannot be achieved (called *oversaturated* conditions, while the opposite situation is known as *undersaturated* condition). Actually, such conditions may occur already when inflow approaches signal capacity. Additionally, steady-state is valid only for random arrivals (when vehicles don't interact to each other and therefore no platooning) at the signal (e.g. Binomial and Poisson headway distributions). This is important and discussed in Appendix B.2.3, because such random arrivals do not reflect the influence of upstream traffic signals and control, being useful only for isolated junctions. The choice between time-dependent queuing models and deterministic models are based on the input information we will have for the queueing model. The traffic theoretic model should be able to model individual vehicles flowing between junctions and provide queue prediction (based on the arrival and departure of vehicles). Therefore, relying on models that use step-functions to model the arrival process is not useful once we have already the arrival process modelled by the traffic theoretic model. The assumption for deterministic models is that the fluctuations of the queue length due randomness can be neglected as they use flow rates for the estimation of the queues. However, if we substitute the queue prediction from flow rates to the prediction of the traffic theoretic model, we can overcome this limitation and still determine a reasonable delay specific for each time window of the planning horizon. In the remaining of this chapter we will focus on **deterministic models**, while the discussion related to other queuing models can be found in Appendix B.2.1 (steady-state) and in Appendix B.2.2 (time-dependent).

4.2.1 Deterministic (Oversaturation) Models

A simple deterministic model explicitly considers the delay in two terms, d_u representing a uniform delay, while the second, d_o, corresponding the overflow delay based on the saturation degree, x, and a period of aggregation, $lenrge$ ([Catling, 1977] apud [FHWA, 2018]):

$$d = d_u + \frac{lenrge}{2}(x - 1), \tag{4.4}$$

in which x is the degree of saturation given by

$$x = \frac{Q/S}{g/c}. \tag{4.5}$$

([May Jr and Keller, 1967] apud [FHWA, 2018]) proposed a deterministic modelling approach to calculate delay and queues, originally for an unsignalized bottleneck. Their approach can be modified for signalized junctions, as it follows:

$$A(t) = \int_0^t Q(\tau)d\tau, \tag{4.6}$$

$$D(t) = \int_0^t C(\tau)d\tau, \tag{4.7}$$

$$q(t) = q(0) + A(t) - D(t), \tag{4.8}$$

$$C(\tau) = \begin{cases} 0 & if\ signal\ is\ red \\ S(\tau) & if\ signal\ is\ green\ and\ q(\tau)\ >\ 0 \\ Q(\tau) & if\ signal\ is\ green\ and\ q(\tau)\ =\ 0,\ and \end{cases} \tag{4.9}$$

$$d = \frac{1}{A(lenrge)} \int_0^{lenrge} q(t)dt, \tag{4.10}$$

where

$lenrge$ is the time in which the arrival and departure processes are stationary,

$A(t)$ is the cumulative number of arrivals from beginning of cycle starts until t,

$D(t)$ is the departures under continuous presence of vehicle queue from beginning of cycle starts until t, and

d is the average delay of vehicles queuing during the time period $[0, lenrge]$.

Such approach (illustrated in Figure 4.10) is similar to the delay term of Equation 4.2, but here the average delay is over the time period $[0, lenrge]$ (e.g. one cycle) at the time the delay on Equation 4.2 is the queue clearance time at time instant t. While Equation 4.8 is equivalent to Equation 4.1. Additionally, as Equation 4.4 shows, at low saturation degree, the delay

is almost the same as the uniform component d_u of the delay d, while at very high saturation degree the equation can describe the delay $d \approx d_o$ (the overflow component of delay) because the small participation of the uniform component of delay. This issue also applies for other deterministic models, i.e. Equations 4.2 and 4.10. Therefore, deterministic queueing models tend to underestimate queues and delays, being reasonable only when ($x \ll 1$, very low flow) or ($x \gg 1$, highly oversaturated, i.e. > 1.4).

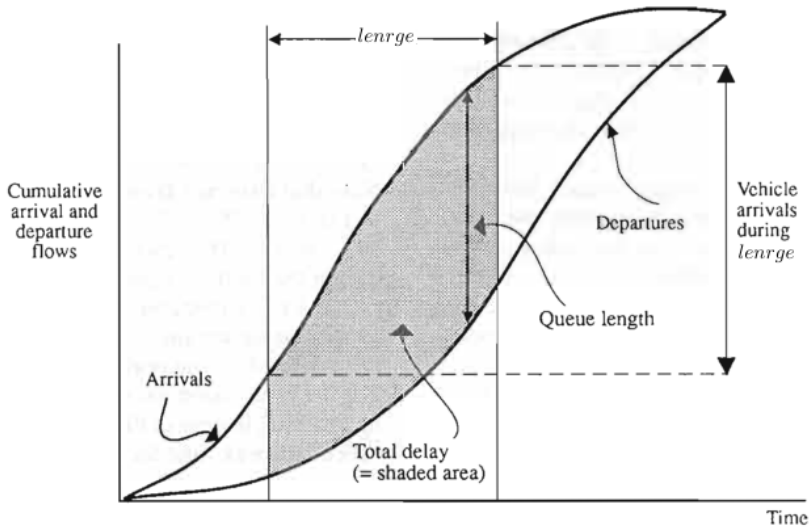


Figure 4.10: Delay on deterministic oversaturated model, adapted from [Rouphail M and Akcelik, 1992].

The problem of the model proposed by [May Jr and Keller, 1967] is that it uses the flow and traffic signal capacity to estimate the number of arrivals and departures, what neglects fluctuations of the queue length due randomness. However, once we are able to model vehicle arrivals and departures for each edge using a traffic theoretic model, we only need the delay calculation of this queueing model (Equation 4.10). On this way we can estimate the delay at certain time range based on the predicted queue which will have already considered the random factors.

Chapter 5

Proposed Local Level Routing System

The idea of local level routing is that the knowledge of traffic light plans in advance may enable vehicles to get a green wave on one route alternative, or if a queue is about to grow beyond the capacity of one cycle of the traffic light, it routes to another alternative. These traffic lights are controlled by Traffic Controllers (TCs) that may control one or more traffic lights and even communicate with other controllers in the same Traffic Network (TN), as seen in Figure 5.1.

Cooperative Automated Vehicles (CAVs) are expected to have embedded On-board Units (OBUs) that are able to store historical discrete travel times or compute them for all edges in the network (if they use functions instead of discrete values), and also get on-line information with better estimation of travel times (TTs) from TCs through Roadside Units (RSUs) once they are inside its communication range. Moreover, such vehicles may send information related to their state (position, speed, etc.), capabilities (e.g. acceleration and deceleration) and planed route. Therefore, we can define the local level routing problem as how to:

1. link signal plans and unexpected events information with the information of current and predicted vehicles on each lane, in order to estimate if a vehicle may get green light or how long it would delay if it arrives an edge at a certain time;
2. choose between individual route advices with suitable route or information of time dependant edge travel times for all vehicles; and
3. consider the properties of the network, if we have a traffic network controlled by one traffic controller, or just a group of junctions with their own traffic controller that communicates (or not) with other TCs.

From this point, we need to understand the behaviour of two groups: *cooperative automated* vehicles; and *modelled* vehicles. Both of them may have a *global destination*, let's say in the other side of the city, but always a *local destination*, which is within the area controlled by the traffic controller they are at certain moment. Once they obtain information of the TN's travel

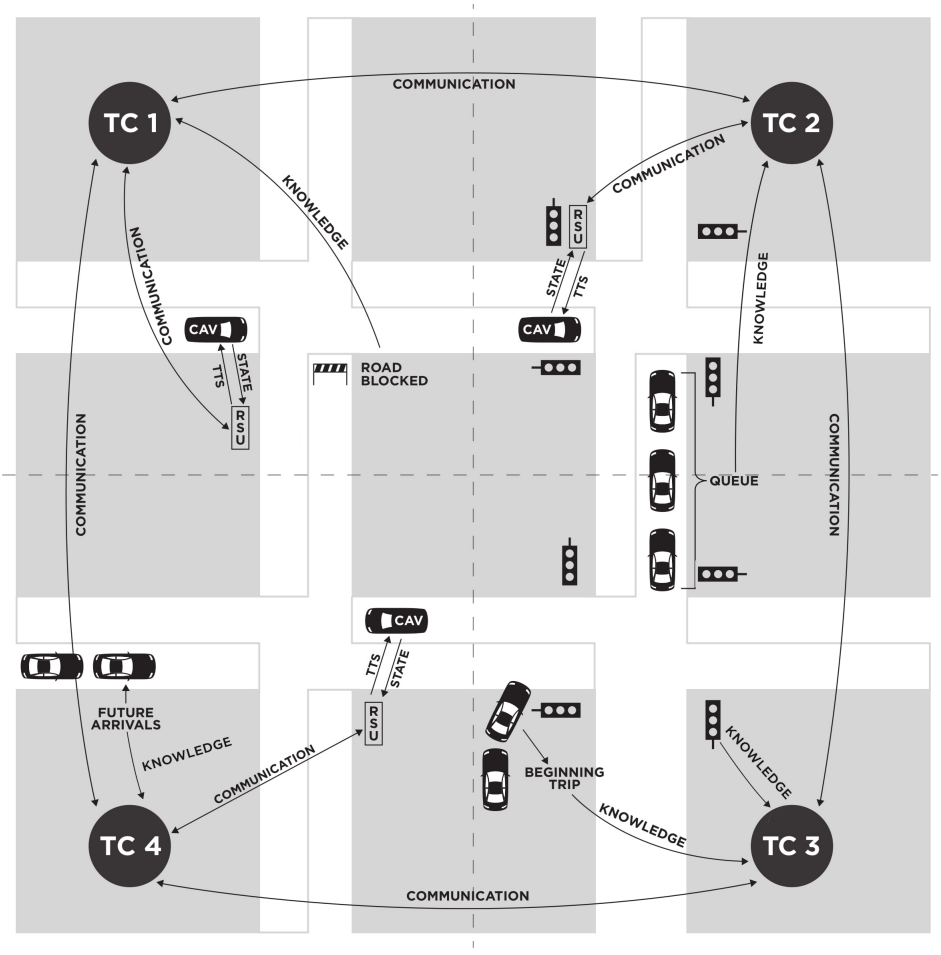


Figure 5.1: Communication among CAVs and TCs in a Traffic Network (TN).

times they may change their local destination, because from their entire route perspective it is better to change completely its route, instead of finding an alternative between current junction and local destination. This is important as an alternative faster route taking more green waves to a local destination may not improve the route to the global destination, because the lack of information from areas outside the TN. Another point is that it would be too demanding for a traffic controller to calculate best routes for each vehicle within the whole network in a very short period of time, though that could be possible for a fixed single local destination given O-D flows. Therefore, to model the route choice of automated vehicles, we need to gather their intended route that are valid for a short period of time. Meanwhile, modelled vehicles are vehicles that do not communicate with traffic controllers, or automated vehicles that didn't arrive yet at certain future junction, or they don't share their intended route.

In this way, the goal is to offer a better alternative of nowadays systems that use historical travel times and cannot capture small changes of travel times due to the dynamism of urban networks. Figure 5.2 shows a grid type

network, where there are 3 coloured areas (which form a traffic network) representing traffic controllers (TCs) that are responsible for the traffic on the edges connecting each junctions) belonging to its colour. On the left side, it represents the route a vehicle would take if it uses only information from its OBU (off-line), while on the other one real-time information (on-line) is provided. We can see that the route changes even outside of the traffic network due the traffic jams. This is because the vehicle chooses the best route considering information of the whole network, using both updated data from the TCs of the traffic network it is inside and its own knowledge (i.e. OBU). The opposite case would be the vehicle receiving an advised route to its *local destination* (inside the traffic network), what would lead to a worse alternative as it would have to pass the congested edges. Therefore, providing real-time information and leaving vehicles to use it together with what they already have seems to be a better idea as they aim to a *global* destination outside the traffic network.

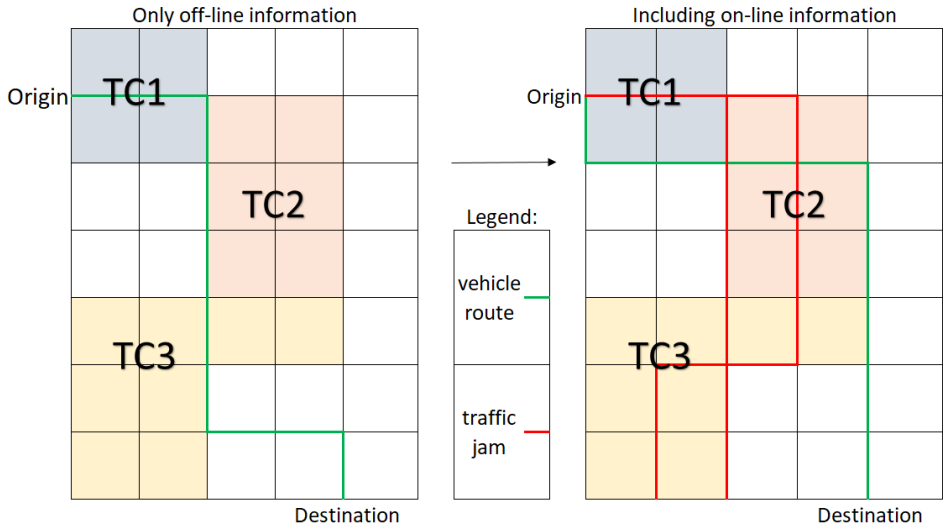


Figure 5.2: Vehicle changing its route due to on-line information from traffic controllers (TCs).

We approach the local level routing problem by assuming that vehicles have the knowledge of average travel times of the whole network, but providing more accurate edge travel times will help them to reduce travel times. The key aspect is the estimation of the travel time as precise and reliable as possible that a vehicle would experience if would enter certain edge at a specific time. Figure 5.3 briefly defines the information required, during each update of the system by a TC, to estimate travel times. First, it is needed to detect CAVs and estimate the number of non-CAVs, including their attributes (e.g initial position and speed, acceleration, desired speed, etc). Then, according to the type of junction (e.g. signalized or unsignalized) the TC either get traffic signal plans or use the junction rules of which vehicle should yield to which, besides lane status information (e.g. if blocked due to an accident). At the end, the TC should be able to predict (or get if available) the route of the

vehicle through its controlled edges, as well as estimate vehicle's arrival and departure from each lane in order to calculate the edge travel times.

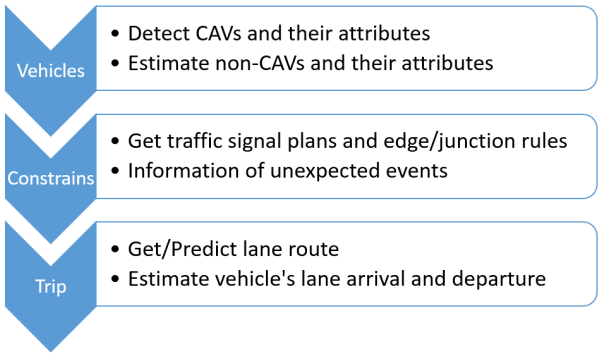


Figure 5.3: Necessary information and steps to estimate travel times.

Using these requirements of which information to collect or estimate in Figure 5.3 and applying it into a system-wide view, where we also have different TCs communicating within the traffic network, we propose a Local Level Routing (LLR) system with five sub-systems, seen in the component UML diagram in Figure 5.4.

The **Traffic Network** sub-system is responsible for defining the Traffic Controller's network attributes, from the incoming and outgoing edges as well as the possible connections between them for each junction it controls, up to the aggregation of these connections into movement groups. It not only provides such static information to all sub-systems, but it is also a single point for other sub-systems retrieve or change dynamic information like maximum lane speed and status (interface is required), queue length prediction, and inflow profile. It is composed by three classes:

- *Junctions*, containing information related to the junctions, like their type, signal plans, planning horizon length, the data received from CAVs and link to the other classes;
- *Edges*, considering the information of each edge such as travel times, lane queue prediction, inflow and outflow profiles, connection critical gap times, saturation flow and capacity; and
- *Movement Groups*, representing the groups of connections that either always get green light (when signalized) at the same time or yield to the same lanes (when unsignalized) for the TC's controlled junctions, including information related to the duration of each movement group green phases as well as its vehicles considered as probe for estimating the travel times.

Traffic Controllers Knowledge Sharing assumes that Traffic Controllers (TCs) share among themselves (through a required interface) the

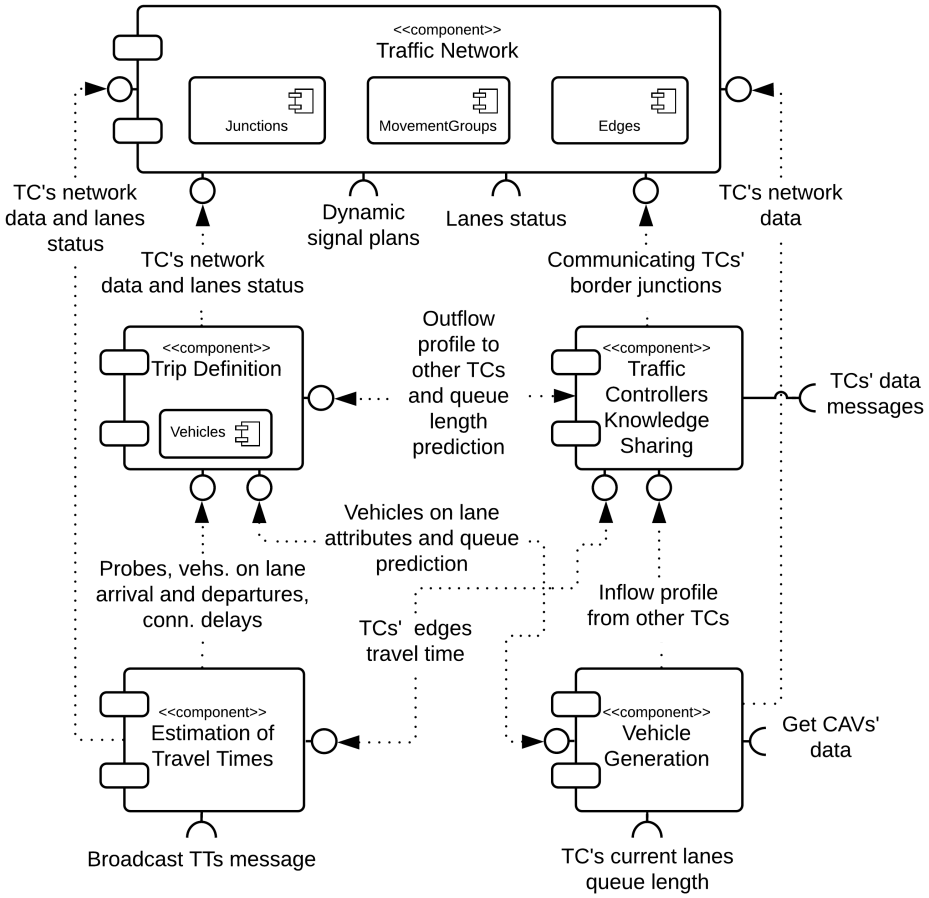


Figure 5.4: Component UML diagram containing the proposed Local Level Routing sub-systems.

travel times of their incoming edges to the junctions they model/control traffic, the arrival profile on their outgoing edges to the neighbouring junctions modelled by other TCs, and the queue length prediction of their incoming edges that are outgoing edges of junctions modelled by other TCs. The inflow profile is used to generate vehicles by the *Vehicle Generation* sub-system, which is the outflow of an upstream junction estimated by the *Trip Definition* sub-system that needs the queue length prediction of outgoing edges to delay vehicle departures when full queue. A TC needs information of which junctions controlled by other connected TCs (and their respective edges) it will send or receive data (i.e flow profile and queue prediction), provided by the *Traffic Network* sub-system.

The **Vehicle Generation** sub-system comprises the set of functions that generate vehicles on the lanes to have their trip defined by the *Trip Definition* sub-system. This sub-system defines the future arrival of vehicles based on the lane inflow profile by estimating the arrival headways; the vehicles starting on lanes by estimating vehicles beginning their trip on each lane (data from the *Traffic Network* sub-system) and current queue length (data from a required

interface), but also constrained by the current vehicles on lane and queue prediction from *Trip Definition* sub-system; as well as the vehicles already on lane, i.e. CAVs (required interface) and those expected to be on lane from last time step. The sub-system can also generate artificial probes that interact with a vehicle in front and traffic light but a vehicle behind does not interact with it.

The **Trip Definition** sub-system defines each vehicle's edge and junction through the TC's network until it reaches its local destination, while it models the movement of vehicles along the lanes (either generated by the *Vehicle Generation* sub-system or estimated, by itself at previous time steps, to be on each lane), considering: the duration of green phases that each movement group receives: the possible connections to other lanes, which vehicles have right-of-way at certain time (from the *Traffic Network* sub-system); and queue length prediction of outgoing edge which may delay the departure and change vehicle's next 1) edge's lane, 2) movement group. This sub-system also defines probe vehicles, which are those that their arrival and departure time on certain lane will be used for estimation of travel times. Moreover, the sub-system calculate connection delays, given the constrained capacity of discharge if vehicles must yield to preferential traffic.

The **Estimation of Travel Times** sub-system accounts each vehicle's lane arrival and departure times as well as their movement group (from the *Trip Definition* sub-system) to estimate an average travel time of the incoming edges. When no trip of a probe vehicle is defined within a time window of the planning horizon for certain lane and movement group, it is possible to use a queuing model considering all vehicle arrivals and departures as well as connection delays and signal timings, or use travel times of the artificial probes. In case the lane status is blocked, it also influences the travel time. Additionally, the estimated travel times (TTs) of all TCs in the same traffic network are broadcast to CAVs (requires an interface).

5.1 Traffic Network

This sub-system controls the characteristics of the TC's network, it is the database needed by other sub-systems with static and dynamic data. There are five main network objects (junctions, edges, lanes, connections, and movement groups) that need information not only related to them, but also the link/mapping between each of them (e.g. the edges of a junction or the lanes of an edge). We will start explaining the required static information and then describe the dynamic information that can be changed by other sub-systems.

5.1.1 Static Data

Figure 5.5 exemplifies some information present in this sub-system. We can see few junctions/edges, some controlled by Traffic Controller 1 (TC1) and others by TC2. Here, controlled means which TC is responsible to model the discharge process of each controlled junction, i.e traffic flowing from the junction's incoming edges to its outgoing edges. Although we usually have two main types of junctions (signalized or unsignalized, i.e. junction contains or not traffic light), we categorize them further into the following list, where a junction may have characteristics of more than one category.

- *Cooperative*, for cooperative junctions, e.g. an intersection that has Roadside Unit (RSU) able to communicate with Cooperative Automated Vehicles (CAVs).
- *Modelled Non-Connected*, for junctions not connected to the TC, i.e. unsignalized or those that don't send their queue length and signal plans to their TC.
- *Modelled Non-Connected*, for connected junctions, i.e. signalized and sharing queue length as well as signal plans.
- *Connections*, junctions that don't have traffic control, i.e. lane merge or lane split.

Each junction has incoming and outgoing edges, and each edge have its own lanes, as well as connections (the arrows in the TC Area 1) that represent which lanes are connected at the junction, i.e. the possible movements at junctions. For both lanes and connections, it should be input their driving length. In the case of connections, it is necessary their maximum capacity of discharge (saturation flow) in $vehicles/s$ and which connection they yield to. Lanes also need the expected gap time between vehicles at the speed limit, maximum speed, and the adjustments of vehicle's acceleration and deceleration. Logically, edges may be incoming to certain junction while outgoing from another, e.g. edge 1 is outgoing from J1 but incoming to J2, and incoming edges need their turning rates (probability to a possible outgoing edge) as well as the probability that vehicles will finish and begin its trip on the edge, in which for the latter it should be given the probability per distance to stop line and the average number of vehicles per length of the time window of the planning horizon.

Figure 5.5 also shows that these connections may be aggregated and receive a number, which represents the group of movements that contains a common status, there are two possibilities according to the junction's traffic control:

- *signalized*, the connections that always receive green light at the same time in all phases of the cycle are in the same movement group (same idea o signal groups); and

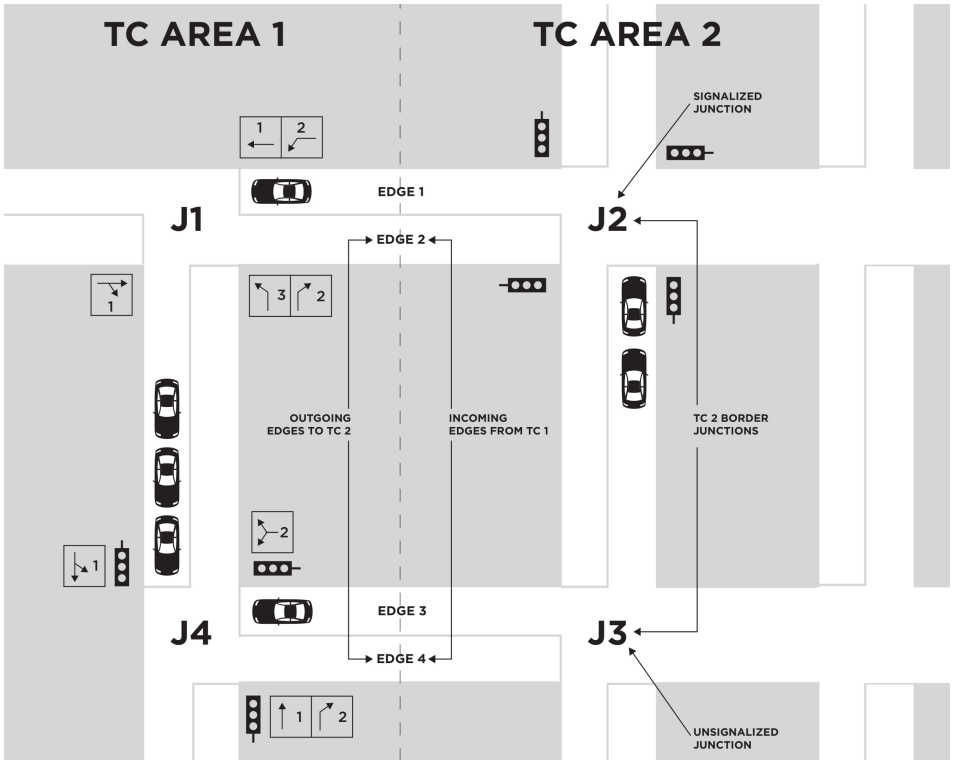


Figure 5.5: Example of some information required for the Network Representation sub-system.

- *unsignalized*, the connections that must yield to the same set of connections are in the same movement group;

For instance, in Figure 5.5, the unsignalized Junction 1 (J1), is a 3-leg junction where vehicles running West-East directions have the priority. In this way, the through and right-turning movements don't yield to any traffic, so they are in the Movement Group (MG) 1. Vehicles taking the connection from West to South must yield to vehicles from East taking any connection, while vehicles coming South to West also need to yield to East coming vehicles, defining MG 2. The last, Movement Group (MG) 3, must yield to all connections which have their incoming lanes from West-East directions. The signalized junction J4 could have the same configuration of MGs, but as its MGs depend on the traffic signal plan (which we set only two phases as example), the set of MGs is different.

Figure 5.5 also illustrates the TC border junctions, which are junctions that have either incoming or outgoing edges leading to a junction controlled by a different TC able to communicate with the TC of the junction been taken as reference. TCs may control just a single junction, or they can model/control several junctions, e.g. an arterial or controlling a signalized junction but also modelling surround unsignalized junctions. A configuration with a single TC for the whole traffic network could provide better control. However, the TC would need to have real-time information of all queue lengths, CAVs detected

by Roadside Units (RSUs) and signal timings of all junctions within the network. On the other hand, in a distributed scheme TCs communicate to each other by sending already processed dynamic data between neighbouring TCs (i.e. edge travel time, lane predicted queue lengths and lane inflow profile). This is important for defining which edges need information to be sent to or receive from other neighbouring TCs, and will be discussed in Section 5.5. This means that the system can be either totally centralized or partially (even fully) distributed, because an underlying common principle in all architectures. Vehicles flow from one junction to another in a logical order. Figure 5.6 shows vehicles passing through a simple network. For example, some vehicles may go from junction 1 (J1) to junction 6 (J6), or J2 to J5, while others do the opposite way. Moreover, there are two Traffic Controllers (TCs) on this route. This means that the system needs a logical sequence of TCs and junctions to have arrivals and departures to be estimated.

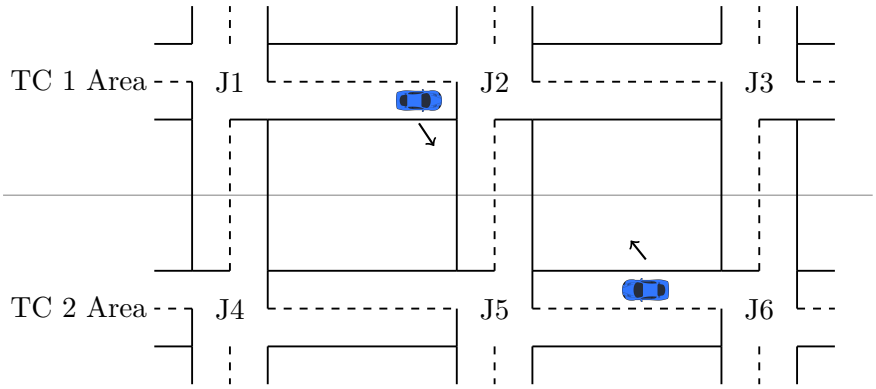


Figure 5.6: Vehicles flowing through Traffic Controller's network.

Using Figure 5.6 as example, we need to estimate the arrivals and departures of J1 first, because J2 will have its arrivals based on the departures of J1, then J3 because it is based on J2 and J4 which is based on J1. One may notice that vehicles go both ways (from J1 to J6 and from J6 to J1), so it is necessary two iterations per algorithm update time step (one iteration per order/sequence - forward or reverse) to estimate the arrivals and departures of vehicles both ways. The system also interchange the first order every update of the algorithm. For instance, at an algorithm update time step, $t = 0$, which happens at $actualt$ real time, the forward order is on the first iteration, $tcit = 1$, then it makes the reverse on the second iteration, $tcit = 2$, while on the next algorithm update time step, t_1, occurring at $actualt = actualt + lenrge$, it makes the reverse order at $tcit = 1$ and then the forward at $tcit = 2$. Notice that $lenrge$ is also algorithm update interval.

We discussed above that the information about vehicle arrivals at junction is necessary to estimate their departures, and consequently their travel time. Therefore, as we will discuss in Section 5.2, lanes also need: a fixed inflow profile (parameters necessary to generate future arriving vehicles using headway distribution functions), i.e. number of vehicles, flow, mean speed, mean

of headways and standard deviation of headways; and the low, mid, and high flow thresholds to define which headway distribution function to use.

Although we call static data, the information may be changed in case of unexpected events, e.g. reduction of maximum speed of a lane, but it is not considered dynamic data because its value is not assumed to variate throughout the planning horizon. The static information related to the TC's network must be inserted using the following order of functions provided in the system (i.e. add all junctions, then all edges and so on):

1. *addJunction*, adding new junctions and their attributes;
2. *addEdgeLanes*, adding edges, their respective lanes and their attributes;
3. *addConnection*, adding connections between edges and their attributes;
4. *addMovGroup*, adding connections' movement groups and their attributes;
5. *addConnYield2Conns*, adding which connection yield to and their attributes; and
6. *setUnsignalizedJctMovGroupPlans*, setting the right-of-way plans (permissive or protected) for unsignalized junctions and their attributes.

5.1.2 Dynamic Data

Analysing again the Figure 5.1, but looking at the flow of information in the illustrated traffic network, we see that the TC has the knowledge of traffic signal plans, lane future arrivals (dynamic inflow profile), lane queue length (predicted queue length), edge travel times (TTs) sent to vehicles, while CAVs' data (state and if possible their planned route). This information is referenced by the time interval it is valid, which is a subdivision of a prediction horizon period.

The planning horizon period, denoted t_{max}, is a sliding prediction time subdivided into a constant length *lenrange* intervals called time windows, denoted as rge_i in Figure 5.7. In addition, at each update of the algorithm, t_t that occurs between *lenrge* time intervals, it is estimated the system dynamic data. The planning horizon defines the limit where it is desired to make estimations of the system's dynamic data. This t_{max} is an input for each junction and should be, at least, the time of one cycle when signalized junction in order to all movement groups be able to discharge. The value of *lenrge* is fixed for the whole system and arbitrary chosen, but very short times (e.g. 5-15 seconds) may not lead to better results and can be computationally costly.

We should notice that this planning horizon divided into time windows of length *lenrge* is used only for edge and lane data. Instead of division with system's *lenrge*, the dynamic data of connections and movement groups use

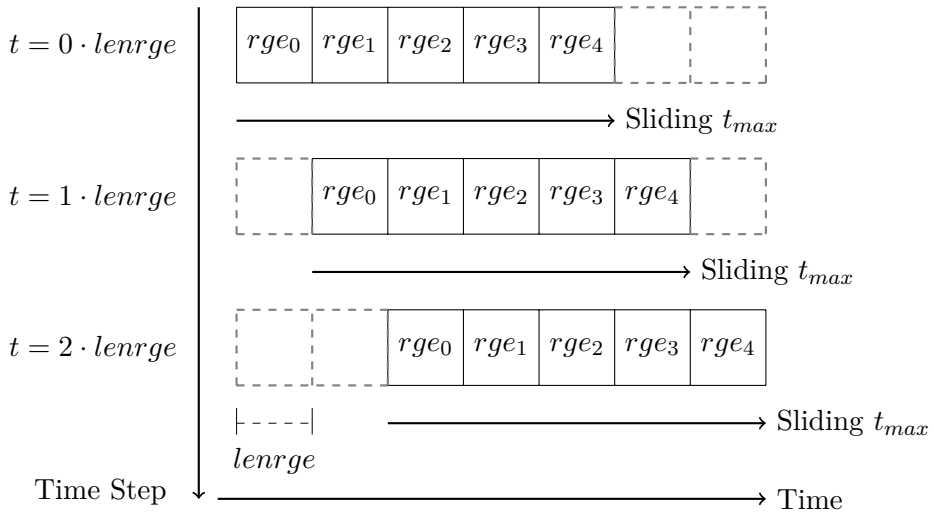


Figure 5.7: Division of a sliding prediction horizon time window into shorter intervals.

the duration of the green phase for the movement group (and consequently also the connection), which is logical due the fact that such objects don't variate within the green phase. The dynamic data related to the signal plan has to be given for each movement group (MG) separately in respect to each phase it gets green, as it follows:

1. MG phase number begin time;
2. MG phase number end time;
3. MG phase number state, *protected* is no need to yield or *permissive* if vehicle must yield to preferential traffic;
4. MG phase number red + amber time; and
5. MG phase number amber time.

An additional "green" phase must be added to each MG of signalized junctions with starting and end time (meaning zero duration) equal to $actualt+t_{max}$, and red + amber as well as amber times set to zero with any of the two possible state. The reason for this is discussed in Section 5.3. For the case of unsignalized junctions, there must be only one signal phase and the mov. groups only one green phase with begin time zero and end time infinity, while red + amber and amber times should be zero. This setting models vehicles at once by the junctions priority rules only.

Now, let's assume we have a traffic signal with 3 phases planned within t_{max}, and a movement group, which we will call $J1MG1$, has green time in 2 of this phases. Considering Figure 5.8, at the first algorithm update time step $t = 0$, the dynamic capacity of discharge of a connection that is in the movement group $J1MG1$, has values only for phase number 0, ph_0,

and number 1 ph_1, because at phase number 2 ph_2 it has red light. At the next update $t = 1$, there are values of capacity again only for the same phase numbers (0 and 1) but at different times.

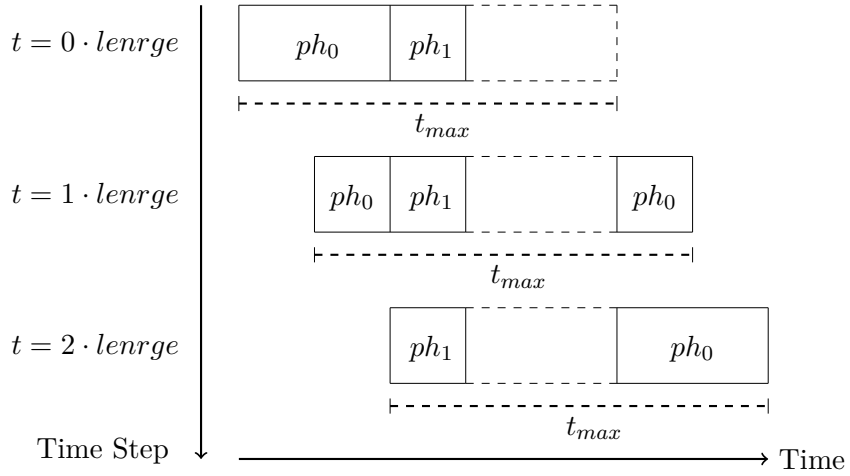


Figure 5.8: Division of movement group's, $J1MG1$, dynamic data by its green signal plan.

There are some dynamic data which auxiliary for other sub-systems and not referenced by time. Sub-systems change their values according to the way they are able to retrieve the information later. For instance, the *Vehicles on Lane* edge instance, denoted as *vehicleslane*, aggregates the list of all vehicles expected to be on each lane throughout the time (but without the information of when they will be on lane). It has three components, described in the following list.

- *Already on Lane*, *alreadylane*, represents the vehicles expected to be on lane due estimations of previous time steps or CAVs detected on the lane at the time the algorithm is going to update its calculations.
- *Starting on Lane*, *startinglane*, corresponds to vehicles that should be generated on the lane because they were not expected to be there given estimations of previous algorithm time steps.
- *Arrivals on Lane*, *arrivalslane*, accounting vehicles that will be arriving within the planning horizon, i.e. $[actualt, actualt + t_{max}]$.

The dynamic information related to the TC's network is controlled by the other sub-systems and will described in the upcoming sections.

5.1.3 List of Required Data

The following list summarizes the required information related to each TC.

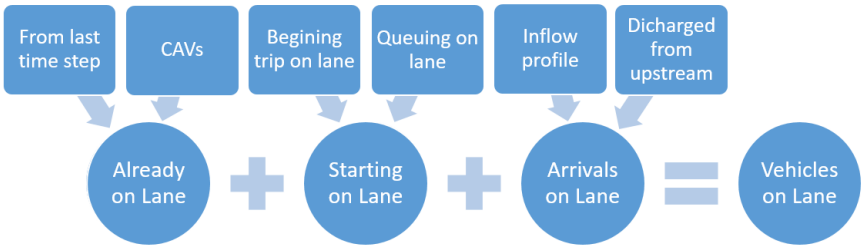


Figure 5.9: Composition of the Vehicles on Lane edge instance.

- Modelled/controlled junctions and their type in the other to be done the estimations by the system.
- Definition of which junctions are TC border junctions as well as their lanes in common with neighbouring different TCs.
- Each junction's edges, lanes, connections, and movement (signal) groups.
- Each edge's lanes, incoming edges turn rates (probability to a possible outgoing edge), and probability that a vehicle will finish its trip on the edge.
- Each lane's length, expected gap time between vehicles at the speed limit, maximum speed, fixed inflow profile, number and distance probability of vehicles beginning their trip, acc./decel. adjustments.
- Each connection's length, fixed capacity, average critical gap time and follow-up time, as well as which connection to yield to.
- Junction's roadside unit (RSU) communication range (when it equipped with one).
- Junction's planning horizon, t_{max}.
- Junction's next traffic signal plan within planning horizon (when signalized junction) for each movement group.

Additionally, the following list summarizes the information required regarding the TC border junction controlled by a different connected TC.

- edge's lanes.
- incoming lane's connections and mov. groups.
- connection's mov. group and outgoing lane.
- movement groups's egressing lanes.

At a system-level, the required information is:

- the planning horizon time window individual length (which is also algorithm update time step interval) $lenrge$; and
- low, mid, and high flow thresholds (minimum headway for the latter).

5.2 Vehicle Generation

This sub-system represents independent processes (tough one process may use data provided by another process) that generate new vehicles and define some of their information that is called attributes and classified into two types:

- *fixed*, for vehicle's parameters such as acc. capability, length and minimum gap; and
- *dynamic*, for the time a vehicle will begin its movement on the lane and the vehicle's state at such moment (speed and distance to stop line).

Cooperative Automated Vehicles (CAVs) are expected to share these attributes to the system and also which lane they are running, though only the edge is important. However, as seen in the system decomposition (Figure 5.4), the system requires an interface with the Roadside Unit to collect the data from vehicles. For non-CAVs (non Cooperative Automated Vehicles), we use standard values and probabilities of each vehicle type (e.g passenger car or truck) stored in the Trip Definition sub-system.

The *fixed* attributes are seen in Table 5.1. The minimum gap represents the distance a vehicle wish to maintain from the vehicle in front (or from the stop line if first vehicle) when it fully stops. The acceleration capability and perceived deceleration correspond to how much the vehicle is able to accelerate and stop (constant values). The speed factor is a value to change the lane maximum allowed speed in order to express vehicle i desired speed at edge/lane j, $vdes_{i,j}$. For example, if the speed limit on the lane is $14m/s$ and the speed factor is 1.1, then the vehicle's desired speed is $vdes_{i,j} = 14 \cdot 1.1 = 15.4$. The last attribute, Route Sharing, is used to flag if the CAV shares its route with the TC, the value can be True (1) or False (0). The *dynamic* attributes are seen in Table 5.2. The distance to stop line and speed correspond to the time the vehicle will start its movement on the lane.

Although the output of generating vehicles is always the same, the inputs depends upon the process defined by the component of the *Vehicles on Lane* edge instance, i.e. *Already On Lane*, *Starting On Lane* and *Arrivals on Lane* to be generated. Figure 5.10 illustrates which components of the *Vehicles on Lane* are modified by this sub-system, while a summary of all inputs, processes and outputs of this sub-system is seen in Figure 5.11.

Fixed Attribute	Unit	Notation
Type	int	$type_i$
Length	m	len_i
Minimum Gap	m	$mingap_i$
Acceleration Capability	m/s^2	acc_i
Perceived Deceleration	m/s^2	$decel_i$
Speed Factor	$dimensionless$	$vfac_i$
Route Sharing	$bool$	sr_i

Table 5.1: Fixed attributes of a generate vehicle i.

Dynamic Attribute	Unit	Notation
Starting distance to stop line	m	$startd_i$
Starting speed	m/s	$startv_i$
Starting time	s	$startt_i$

Table 5.2: Dynamic attributes for generating a vehicle i.

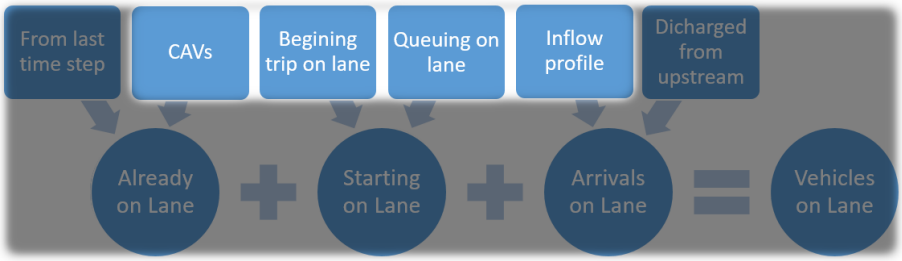


Figure 5.10: The part of Vehicles on Lane modified by the Vehicle Generation sub-system.

5.2.1 Already on Lane

This is the first process that happens and it defines the Cooperative Automated Vehicles (CAVs) that are in the list *alreadylane* by either:

1. replacing the closest vehicle expected to be on lane due estimations from previous time steps and changing attributes to the received ones by the CAV; or
2. creating a new vehicle in the system with the data from the CAV when the edge doesn't have any non-CAV already on lane.

This replacement is done in order to avoid creating excessive newly detected CAVs when it was already expected a vehicle to be on the lane (very useful in high penetration rates of CAVs). This process only needs the list of non-CAVs to be replaced on the edge, as both fixed and (Table 5.1) and dynamic (Table 5.2) attributes are provided already by the CAV. When a CAVs is to

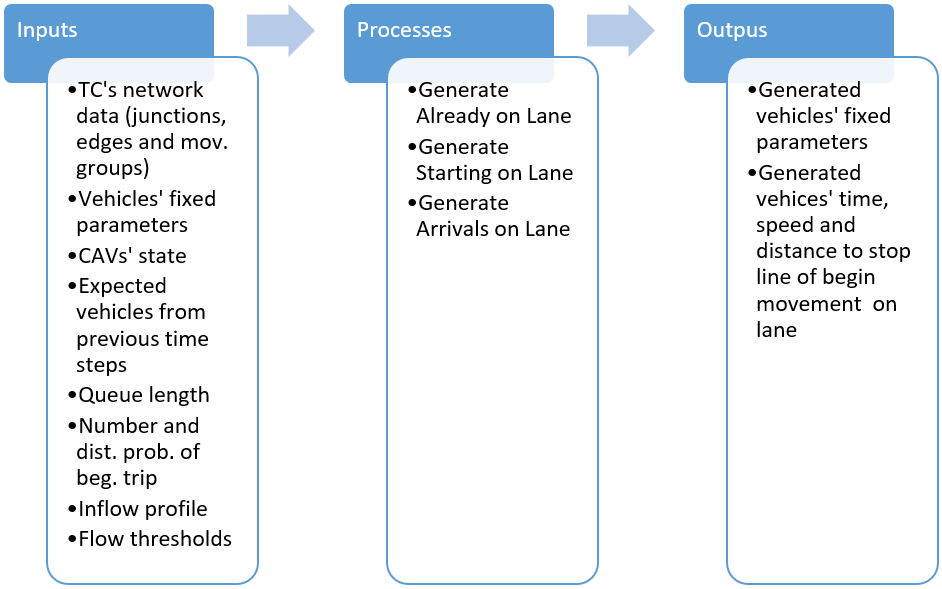


Figure 5.11: Inputs and outputs of the Vehicle Generation sub-system.

be generated but there are no vehicles expected to be on lane at the time step, the *cavlanebalance* variable (for each edge) is used to balance the number of CAVs with the vehicles expected to be on lane and beginning their trip on lane.

5.2.2 Starting on Lane

This process can be split into two parts, the first one is different for the type of vehicle to be generated and the second is the same for both. It requires the estimated queue length (in number of vehicles), *queuenum*, provided by a sub-system not included in our local level routing system (an interface is also required, see Figure 5.4), and the vehicles beginning their trip on the lane *begnum*. It generates two types of new vehicles:

- *queuing on lane*, which represents the missing queueing vehicles given *queuenum* and those stopped vehicles in the *alreadylane* list; and
- *beginning trip on lane*, when it is assumed that some vehicles may begin their trip/journey on the lane/edge at the current time step.

The inputs for queuing vehicles are only the estimated queue length number *queuenum* and the number of halted vehicles in *haltedalready* at the current time step. The needed new queuing on lane vehicles will be $queuenum = max(queuenum - haltedalready, 0)$. If $queuenum < haltedalready$, then delete non-CAVs from *alreadylane* until *queuenum* equals *haltedalready*. The main input for the vehicles beginning their beginning trip on lane is be the number of vehicles that may begin, *begnum* (in one interval of algorithm

updates *lenrge*), and the distances they may start as well as the probabilities of these distance. Table 5.3 shows an example of such input, the system applies *Binomial Distribution* (which outputs 0 or 1) for the decimal part in order to decide if an additional vehicle should be also generate besides the integer part of the number.

Number of vehicles per *lenrge*	
1.1	
Distance	**Probability**
10 meters	0.3
50 meters	0.7

Table 5.3: Input to generate vehicles beginning trip on lane i.

Another possibility is to input the string "random", and the vehicle will be generated at any distance (within the lane length). The secondary input is the edge's *cavlanebalance*, which is subtracted by the number of vehicles beginning their trip on lane, *begnum*, i.e. $begnum = max(begnum - cavlanebalance, 0)$, and updates its *cavlanebalance* by subtracting *begnum*. Once the updated values of queuing (*queuenum*) and/or beginning trip (*begnum*) are bigger than zero, it starts the second part of the process. It is randomly chosen one vehicle type for the vehicle and consequently its fixed parameters, as previously explained at the beginning of this section. Then, the sub-system need to define the vehicle's distance and speed, while the starting time, $startt_i$, is the current time step time, *actualt*.

Figure 5.12 illustrates this second part, which defines the distance where the vehicle will begin its movement by trying to find an space the vehicle would fit between two vehicles already on lane (now including the already inserted starting vehicles). It also estimates the speed if the vehicle would be influenced by the vehicle in front, $fronti$, or not (for the latter it was arbitrary chosen $4m/s$), but not higher than vehicle's desired speed, at its j^{th} junction/edge/lane of its route, $vdes_{i,j}$. The $already lane_{l,loi}$ is the ordered list with vehicles i already on lane l that have indexes called lane order index, loi, for each vehicle in the list. The $lasti$ represents the last vehicle i in the list, while $fronti$ is the vehicle to have its starting distance to stop line, *startd*, to be considered. The distance of starting, *initdist*, for the vehicles beginning their trip on lane is only the reference to initialize the algorithm, while for queueing vehicles the reference distance (*initdist*) is zero. This is because vehicles cannot share the same space and the order of vehicles (distance to stop line if already on lane) is important to ensure the FIFO (First-In-First-Out) condition. Figure 5.12 also shows that vehicles are not created when the back (rear) distance of the vehicle to be generated reaches the lane length.

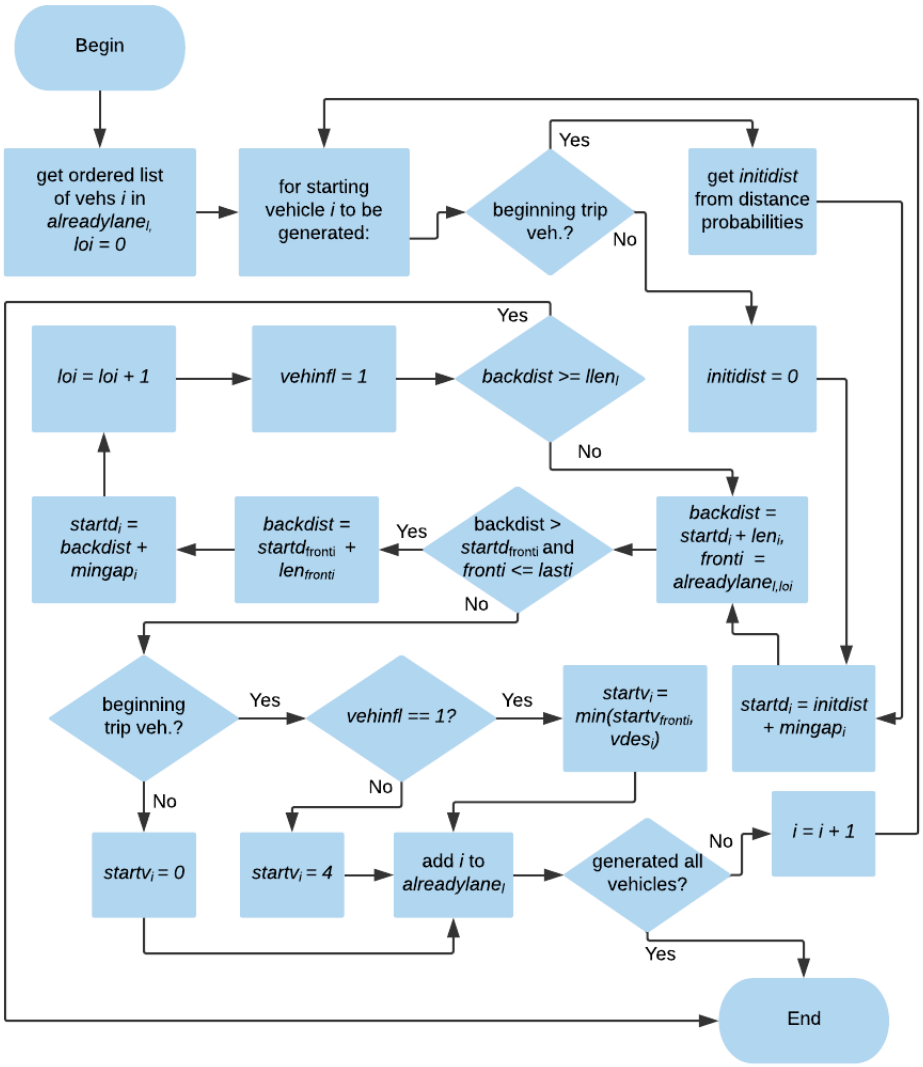


Figure 5.12: Steps to define starting vehicles' distance and speed.

5.2.3 Arrivals on Lane

The process to generate the *Arrivals on Lane*, *arrivalslane*, happens on each iteration (i.e. $it = 1$ and $it = 2$ at certain time step t) of the system, different from the *Already on Lane* and *Starting on Lane* that happens always at the first iteration. As discussed in Section 5.1, a real world network may be subdivided into sub-networks that are controlled by different Traffic Controllers that may communicate to each other, and there is an order of TCs and junctions the system estimates arrivals and departures. Another aspect of it is that a TC controls only the incoming edges/lanes of the junctions it controls. Based on this the *Arrivals on Lane* occurs for the lanes when both conditions in the following list happens.

1. the lane follows the order of junctions the system estimated arrivals

and departures (for instance from J1 to J6 in Figure 5.6), as the order changes per iteration this process happens either at the first or second iteration for each lane.

2. the lane comes from a junction not modelled by the TC (which controls only the incoming edges/lanes of the junctions it controls), the arrivals from junctions controlled by the TC is done by the *Trip Definition* sub-system (discussed in Section 5.3).

The generation of arrivals using this process is based oin the future arrival of vehicles (after the current algorithm time step) within t_{max}, and it uses three statistical headway distributions, each for a different range of traffic flow volume. Therefore, the necessary inputs are system-level flow thresholds, in $vehicles/s$, to define which distribution to use (λ_{low}, λ_{mid}) and the parameters of the distributions (the so called inflow profile). The inflow profile is given in Table 5.4.

Parameter	Unit	Notation
Number of vehicles	$vehicles$	$arrnum$
Mean speed	m/s	v_{mean}
Mean of headways	s	h_{mean}
Standard deviation of headways	s	h_{std}

Table 5.4: Inflow profile parameters.

All lanes from junctions not controlled by the TC must have a fixed inflow profile, and can receive a dynamic inflow profile when the lane comes from a junction controlled by another TC which communicates to the TC controlling the lane to have arrivals estimated (this will be discussed in Section 5.5). Figure 5.13 exemplifies the types of *Arrivals on Lane*, in which fixed and dynamic inflow profiles are used by the Vehicles Generation sub-system, while the discharge from upstream is used by the Trip Definition sub-system.

Although the system always estimates the headways between vehicles for each arrival range rge of the planning horizon, t_{max}, the fixed inflow has the same parameters all ranges, while the dynamic inflow has different parameters per arrival range as shown in Figure 5.7. An example of the parameter number of vehicles is seen in Figure 5.14. This is important because when estimating arrivals, the calculated headways of vehicles are valid only for that range which has interval length of $lenrge$.

The range interval is also used to estimate the flow, λ_l^{arr}, of vehicles during each arrival range, i.e. $\lambda^{arr} = arrnum/lenrge$, on each lane l and then define which headway distribution to use based on the flow thresholds. In all cases, the system uses the Cumulative Distributive Function (CDF), denoted $F(h)$ and computed as

$$F(h) = p(t \leq h) \tag{5.1}$$

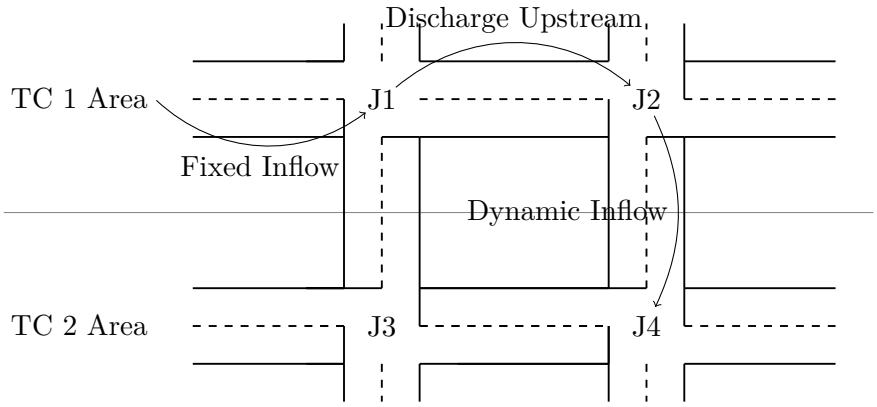


Figure 5.13: Different types of arrivals on lane. Fixed and dynamic inflow are responsibility of Vehicle Generation sub-system, while discharged from upstream is done by the Trip Generation sub-system.

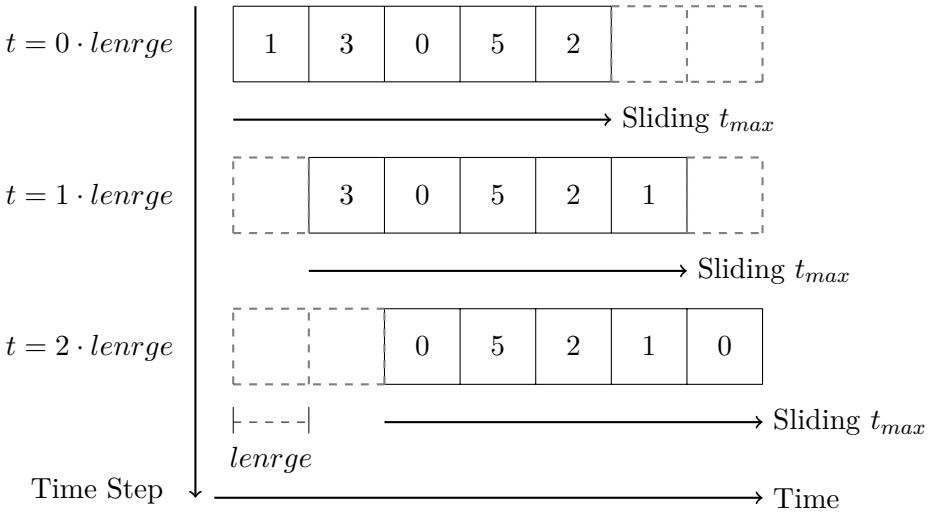


Figure 5.14: Example of the parameter number of vehicles for a dynamic inflow profile.

for each headway distribution, and calculate its value for each headway h that varies between the interval $[h_{min}, h_{min} + \Delta t, ..., lenrge - \Delta t, lenrge]$. Where h_{min} is the minimum headway (in seconds) and Δt is the minimum possible difference of headways, e.g. 1 second, and t is any value of headway. Then, for each vehicle to arrive the system randomly chooses a number between [0,1] which corresponds to a value t of the CDF that lies within a possible headway h and the next one $h + \Delta t$ (i.e. $h = [h, h + t, ..., h + \Delta t - t]$). Figure 5.15 illustrate this idea, notice that it may happen the value can be higher than the maximum headway $lenrge$, when this occurs the system uses $lenrge$ as the headway.

When $\lambda_l^{arr} < \lambda_{low}$, the flow of the arrival range is considered low and the headways assumed to follow Negative Exponential Distribution due to almost or no interaction between the arriving vehicles. According to [Mathew, 2017a],

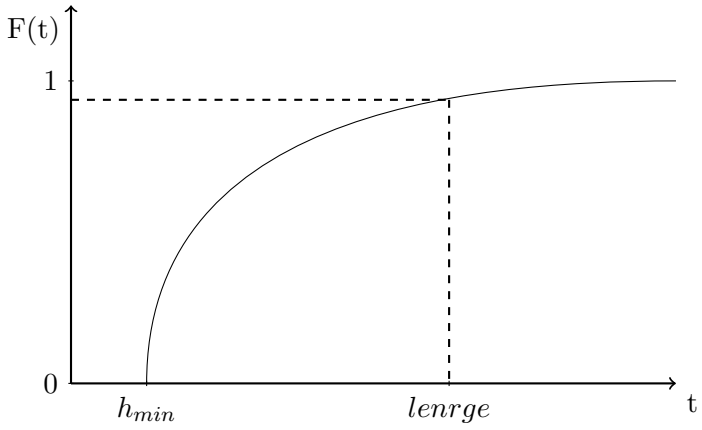


Figure 5.15: Example of a CDF and how the headways are defined.

the CDF if given:

$$F(h) = 1 - e^{-\lambda_l^{arr} h}. \tag{5.2}$$

When $\lambda_l^{arr} < \lambda_{mid}$, the flow of the arrival range is considered medium and the headways assumed to follow Pearson Type III Distribution, as certain vehicles will have interaction with the other (platoons) or not. The CDF if given as it follows from [Mathew, 2017a]:

$$F(h) = p(t \leq h) = 1 - \int_h^{\infty} f(t)dt. \tag{5.3}$$

Equation 5.3 means that the CDF of headway h is the probability of any headway time t shorter and equal than h, which is also 1 minus the probability of any headway time above h given we know its probability density function, denoted $f(t)$. Although there is no closed form solution for Equation 5.3, if we assume a linear line between $f(h)$ and $f(h + \Delta t)$, which is acceptable if Δt is small enough, as seen in Figure 5.16, we can find an estimation of the probability for $h = [h, h + t, ..., h + \Delta t - t]$ by doing

$$p(h \leq t \leq h + \Delta t) \approx \left[\frac{f(h) + f(h + \Delta t)}{2}\right] \Delta t \cdot h. \tag{5.4}$$

The probability density function of the Pearson Type III Distribution is:

$$f(t) = \frac{\lambda_p}{\Gamma(K)} \left[\lambda_p (t - h_{min})\right]^{K-1} e^{-\lambda_p (t - h_{min})}, \quad K, h_{min} \in R, \tag{5.5}$$

where K is the shape factor greater than 0 calculated as

$$K = \frac{h_{mean} - h_{min}}{h_{std}}, \tag{5.6}$$

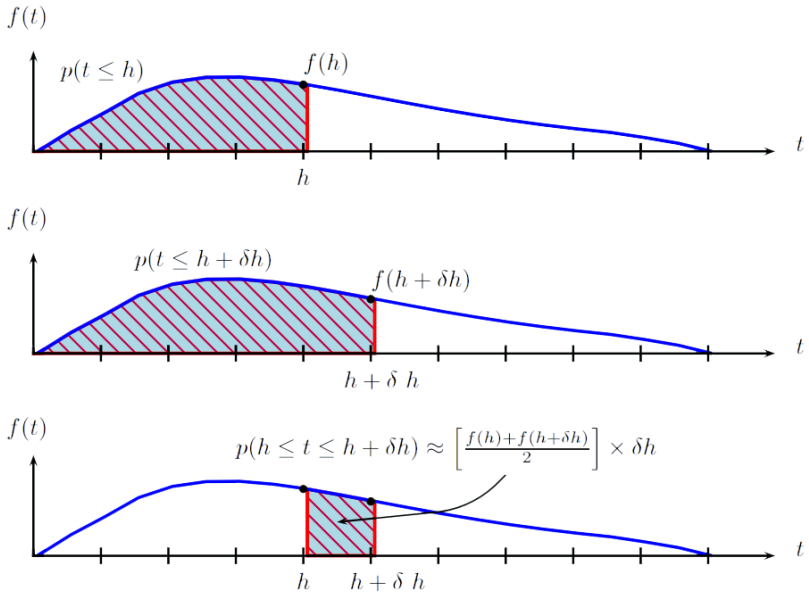


Figure 5.16: The expression for probability that the random variable lies in an interval for Person Type III distribution [Mathew, 2017a].

while λ_p is the flow rate parameter (and not the flow rate λ itself) estimated by

$$\lambda_p = \frac{K}{h_{mean} - h_{min}}, \tag{5.7}$$

and $\Gamma(K)$ is the gamma function, as it follows:

$$\Gamma(K) = (K-1)!. \tag{5.8}$$

When $\lambda_l^{arr} >= \lambda_{mid}$, the flow of the arrival range is considered high and the headways assumed to follow Normal Distribution, where vehicles are expected to arrive close to each other in platoons. The CDF is given as it follows from [Mathew, 2017a]:

$$F(h) = p(t \leq h) = \int_{-\infty}^{h} f(t)dt. \tag{5.9}$$

Equation 5.9 also doesn't have a closed form solution, though the probability of a headway for an interval $h = [h, h+t, ..., h+\Delta t-t]$ can be computed easily by making $p(h \leq t \leq h+\Delta t)$, similarly to equation 5.4, and solving it using numerical integration, as Figure 5.17 shows.

However, considering that the probability for any random variable can be normalized by its mean and standard deviation, we can use the standard normal distribution table to find the values for

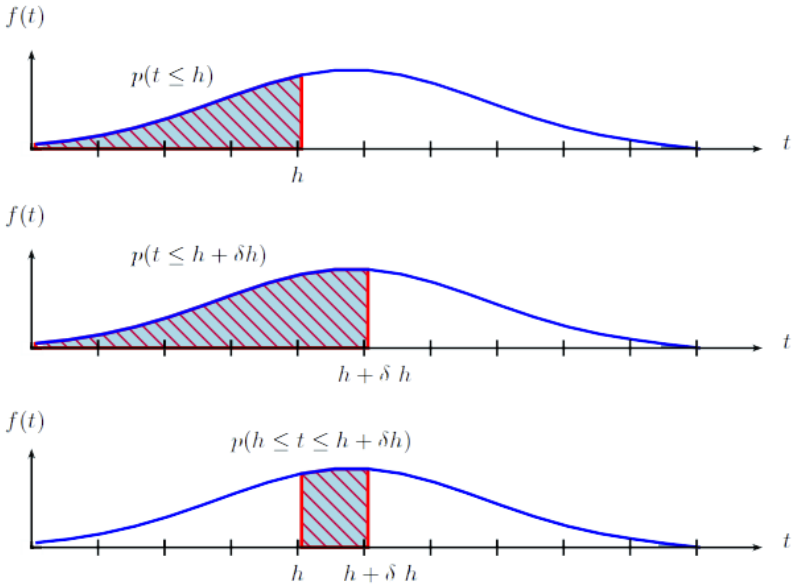


Figure 5.17: The expression for probability that the random variable lies in an interval for Normal distribution [Mathew, 2017a].

$$F(h) = p(t \leq h) \approx p_{table}\left(t \leq \frac{h - h_{mean}}{h^*_{std}}\right), \tag{5.10}$$

in which h^*_{std} is a correction of the standard deviation of headways when the random variable t cannot be negative. This correction is given by

$$h^*_{std} = \frac{h_{mean} - h_{min}}{3}, \tag{5.11}$$

where the 3 comes from the fact that if $h_{min} = h_{mean} - 3h_{std}$, which means that 99% of the headways will be greater than h_{min}, as seen in Figure 5.18.

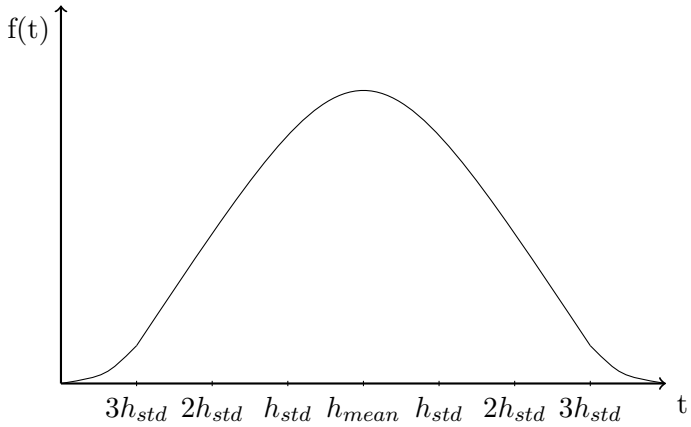


Figure 5.18: Intervals of standard deviation for Normal Distribution.

5.3 Trip Definition

This sub-system is responsible for modelling the movement of vehicles on the lanes and through the traffic controller's network. This is done by defining vehicle's dynamic attributes on each edge of its route, as well as choosing the probe vehicles that will be used to estimate travel times. In addition, given this modelling process, it also updates the lanes' queue length prediction and connections' capacity and delays. Figure 5.19 illustrates the inputs, processes and outputs of this sub-system.

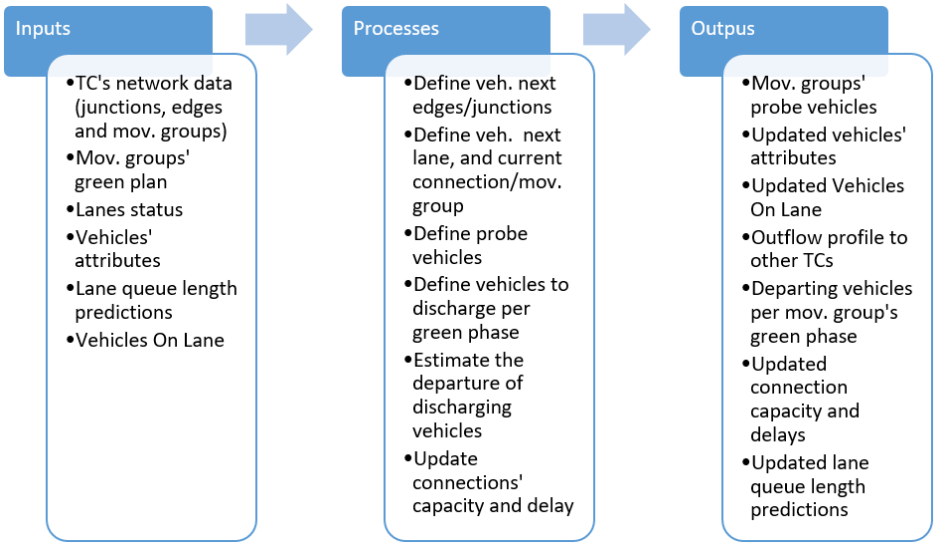


Figure 5.19: Inputs and outputs of the Trip Definition sub-system.

As discussed in Section 5.2, vehicles have fixed and dynamic attributes, in which the fixed ones are defined by the *Vehicle Generation* sub-system together with 3 dynamic ones, i.e. *startd*, *startt* and *startv*. The dynamic attributes are specific for each edge (and consequently lane and junction) the vehicle will use throughout its route. For example, $route_{i,j}$ is list to stores the edge ID of vehicle i running on its j^{th} junction/edge/lane of its TC's network route. Notice that it is necessary only the route on the edges controlled by the TC that is modelling the vehicle. The list of all attributes for a vehicle i on its j^{th} junction/edge/lane of its route are seen in Table 5.5.

The junction ID, $jct_{i,j}$, is the one that the vehicle i edge j goes to (the junction's incoming edge), while the connection ID, $conn_{i,j}$, and mov. group, $movg_{i,j}$, are defined by the next edge $(j+1)$ of the vehicle i route. The lane ID, $lane_{i,j}$, correspond to the expected lane the vehicle will depart from stop line, and such lane may be different from the lane that two type of vehicles were initially generated:

- Cooperative Automated Vehicles (CAVs); and
- vehicles generated by inflow profile.

Dynamic Attribute	Unit	Notation
Junction ID	*string*	$jct_{i,j}$
Lane ID	*string*	$lane_{i,j}$
Mov. Group ID	*string*	$movg_{i,j}$
Connection ID	*string*	$conn_{i,j}$
Arrival Time	s	$arrt_{i,j}$
Arrival Speed	m/s	$arrv_{i,j}$
Phase Start Time	s	$startt_{i,j}$
Phase Start Speed	m/s	$startv_{i,j}$
Phase Start Dist.	m	$startd_{i,j}$
Phase End Time	s	$endt_{i,j}$
Phase End Speed	m/s	$endv_{i,j}$
Phase End Dist.	m	$endd_{i,j}$
Lane Acc.	m/s^2	$lacc_{i,j}$
Lane Decel.	m/s^2	$ldecel_{i,j}$
Desired Speed	m/s	$vdes_{i,j}$
Veh. Len. Crossed Time	s	$lent_{i,j}$
Travel Time	s	$vehtt_{i,j}$
Veh. Crossing	*bool*	$vehc_{i,j}$
Veh. Next Updt. Speed	m/s	$nuptv_{i,j}$
Veh. Next Updt. Dist.	m	$nuptd_{i,j}$

Table 5.5: Dynamic attributes for defining the trip of vehicle i at each edge/lane j.

This is because the *Vehicle Generation* sub-system only "says" that certain vehicle is being generated at an specific lane and it gives vehicle's fixed attributes as well as the time, speed and distance to stop line. The *Trip Definition* sub-system must define which lane it thinks the vehicle will finish its movement on the lane, and there are two reasons for this. The first one is that according to the next edges the vehicle will take, it has to be on other lane than it was generated (or detected if CAV). The second one tries to consider that vehicles will not stay in a long queue if they can take another which is shorter. It is important to say that the system doesn't model lane changes, it just put the vehicle on the lane that would be more suitable, i.e. overtaking is not possible. Meanwhile, other types of vehicles continue on their lane that they were generated, either because the lane is already the expected final one (vehicles from last time step and discharged from upstream) or because they are likely on their intended lane at departure (beginning trip on lane and queueing on lane).

For other dynamic attributes we have the arrival time and speed, $arrt_{i,j}$ and $arrv_{i,j}$ respectively, of the vehicle on the lane (no needed distance because it is implicit the lane length $llen_j$). Then the set of attributes phase start and phase end, representing the distance ($startd_{i,j}$ and $endd_{i,j}$), time ($startt_{i,j}$ and $endt_{i,j}$), and speed ($startv_{i,j}$ and $endv_{i,j}$) of a vehicle at the start and

end of its movement during each analysing phase of its movement group on the lane. As we are going to see later at this section, the modelling of vehicle movement along the lane is divided by its movement per green phase of its movement group until it crosses the stop line, what doesn't store values of phase start and end for each phase but updates them per analysed phase, instead. Other attributes are the lane acceleration, $lacc_{i,j}$, and deceleration, $ldecel_{i,j}$, that represents the acceleration and deceleration of the vehicle adjusted to a specific edge. The lane desired speed is the expected maximum speed vehicle i will try to achieve, i.e. $vdes_{i,j} = vfac_i \cdot lvmax_j$, in which $lvmax_j$ is the maximum allowed speed on edge/lane j. The vehicle length crossed time, $lent_{i,j}$, is the time in which the whole vehicle (its length) crosses the stop line, and it is calculated only when the vehicle will cross the stop line (given by the vehicle crossing flag variable $vehc_{i,j}$). The vehicle next update speed and distance, $nuptv_{i,j}$ and $nuptd_{i,j}$ respectively, are used to store the expected distance and speed the vehicle will have at the next time the algorithm will update, and use it as the vehicle starting point at next update. Finally, we have the vehicle travel time on the edge/lane, denoted $vehtt_{i,j}$.

The main dynamic information for this sub-system is the *Vehicles On Lane*, *vehicleslane*, because it checks which vehicle is or will be on the lane and estimate their discharge process at the junction in order to add it to a downstream *vehicleslane*. Figure 5.20 illustrates which component of the Vehicles on Lane is modified by this sub-system (but reads all of them). If a vehicle and its attributes were already in the TC's *Vehicle* object before the generation of new vehicles, then the vehicle was discharged from a upstream junction controlled by the same TC, or the vehicle is expected to be at certain place on the lane at the time the algorithm is updating.

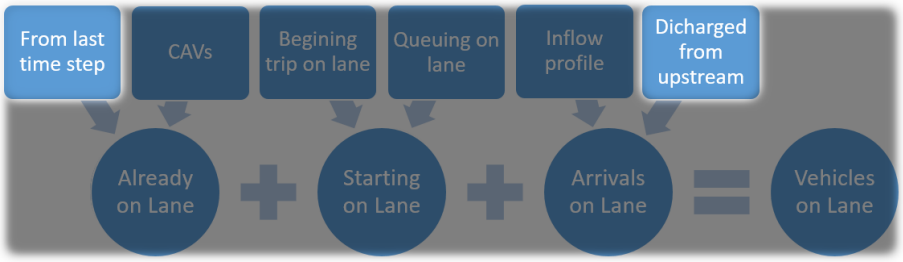


Figure 5.20: The part of Vehicles on Lane modified by the Trip Definition sub-system.

Figure 5.21 illustrates that, differently from the other sub-systems that have independent processes, the Trip Definition sub-system have processes that are dependent of each other. Although the description of the flowchart and processes are in the upcoming subsections, we can point out the cases when the sub-system is called for different needs, either by:

1. defining the initial trip attributes of newly generated vehicles;

2. estimate the departure of vehicles on the incoming lanes of the junction; or

3. calculating the delay that each connection will have for its mov. group's green phase.

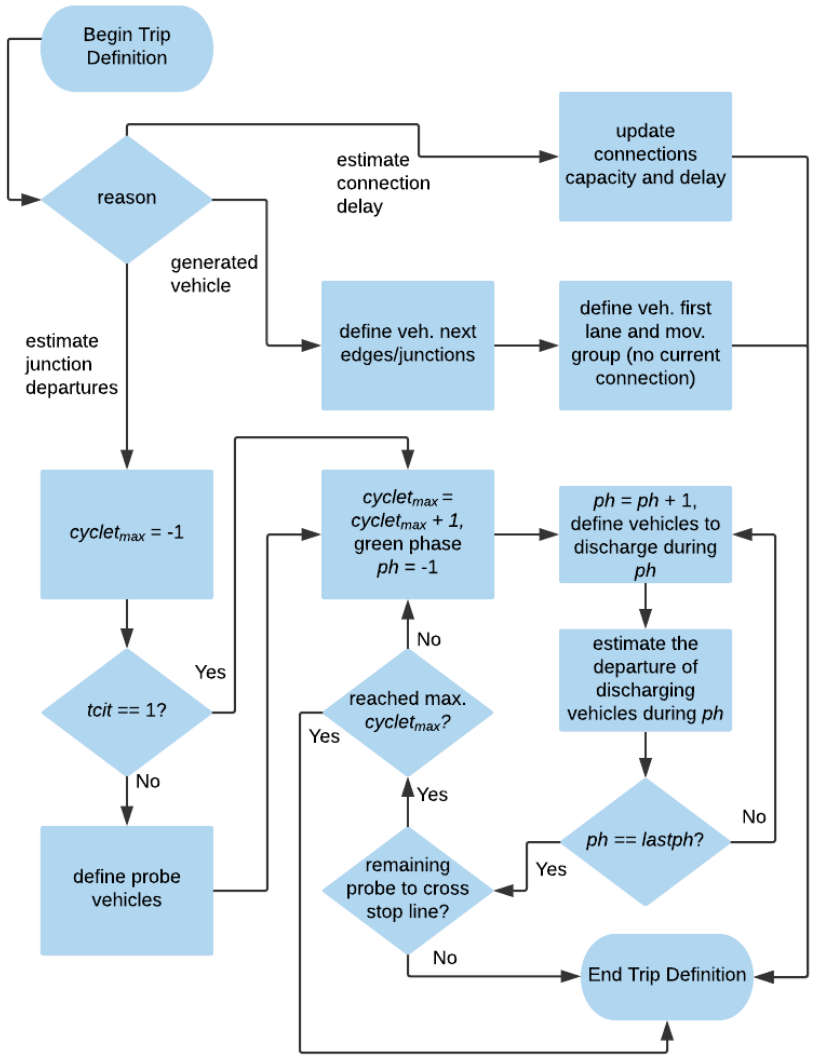


Figure 5.21: Sequence of the Trip Definition sub-system processes according to the reason and algorithm update iteration.

5.3.1 Define Vehicle Edges and Junctions

This process happens right after the vehicle is generated by the *Vehicle Generation* sub-system (explained in Section 5.2) and during the *Estimate the Departure of Discharging Vehicles* process (discussed in Section 5.3.5). Considering the current junction/edge, j, being modelled, the function of this

process is to define the next 2 edges (and consequently junctions), i.e. $(j+1)$ and $(j+2)$, to be modelled towards its local destination. This is because the choice of the lane on edge $(j+1)$ is based on edge $(j+2)$, as we will see in Section 5.3.2 and illustrated in Figure 5.22 for vehicle i.

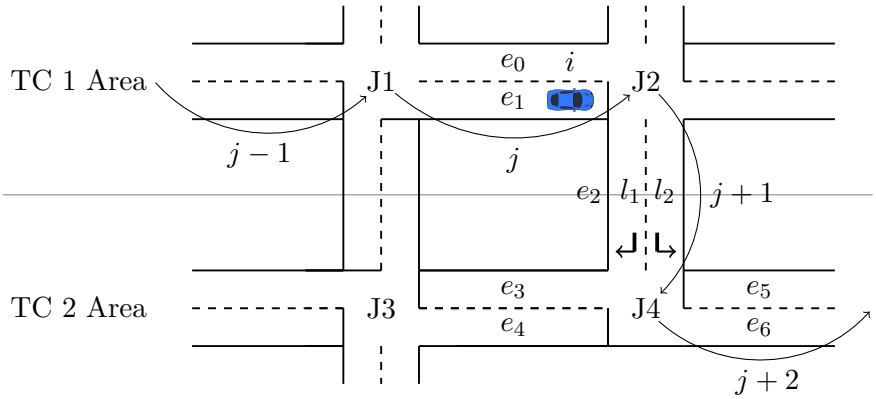


Figure 5.22: The chosen lane on edge $(j+1)$ depends o the edge $(j+2)$, what makes necessary to have always knowledge of the next 2 edges ahead.

One may notice that Cooperative Automated Vehicles (CAVs) may share their routes, so this process is narrowed to only defining the remaining junctions to be modelled. The chosen method for defining which outgoing edge a vehicle will go to once it is running on a certain incoming edge is through turning rates. Although this method might be the best one, it is the simplest one to apply and the easiest to get real data compared to others using route choice models (see Appendix B.3) which may lead to better results. In addition, the following list presents the rules that either stops or restricts the possible next edges.

1. If vehicle will finish its route at the last chosen edge (due to certain probability of it).
2. If vehicle is a *Starting Vehicle*, then the possible first next edges is constrained by the vehicle's lane which it was generated at. For instance, in Figure 5.22, if a vehicle starts on edge e_2 at lane l_1, then the possible next edge can only be e_3.
3. If vehicle would go to an edge already in the route, what would create an unrealistic loop.

5.3.2 Define Vehicle Lane, Connection and Movement Group

This process is responsible for choosing a vehicle's next lane and mov. group (of this next lane), as well as the current connection from the current lane to the next one. These choices are based on the already chosen next edge, the

lanes' queue length prediction of the next edge and the lanes status (if any blockage). The process is called at two situations:

1. vehicle is starting on lane (when newly generated CAV or arriving vehicle), in which the "next" is in fact the first lane and mov. group while no current connection is given; or
2. vehicle have reached at stop line (when TC is modelling vehicle's movement along the edge).

The process occurs right after the first definition of the next edges and junctions (described in Section 5.3.1), i.e. after generating a vehicle, and during the *Estimate the Departure of Discharging Vehicles* process (discussed in Section 5.3.5). Figure 5.23 exemplifies the situation where the movement of vehicle i is being modelled by TC1 for junction J2 along the edge e_1, which is its j^{th} junction/edge/lane of its route. The next edge of vehicle i, $(j+1)$, is e_2 and the after next edge, $(j+2)$, is e_7. There are two possible lanes at e_2, l_2 and l_2, as they both have connections that go through. However, at the moment vehicle i reaches the stop line ($t = 10$) there is a predicted queue on lane l_2 of edge e_2 because the vehicles are left-turning and waiting due another predicted queue on edge e_6. Therefore, vehicle i will choose lane l_1 because it doesn't have any predicted queue for that moment. The next mov. group is the one which enables vehicle i to go from the next lane l_1 of its next edge e_2 to the after next edge e_7. Meanwhile, now it is known that the next lane is l_1 of e_2 and the current lane is l_1 or e_1, so the current connection (and next mov. group) is the one that connect these two lanes. In fact, the lane choice also consider the lanes status (which has the highest priority), i.e. if a lane has shorter predicted queue at certain moment but it is blocked/closed, the vehicle will not go there, but to an open one with longer queue instead.

5.3.3 Define Probe Vehicles

As the main goal of the system is to estimate travel times per time windows of the planning horizon corresponding to the arrival range of a vehicle at certain edge, this process defines which vehicles generated by the *Vehicle Generation* sub-system will be used to estimate the travel time for the edges they will pass according to their arrival time. It is called on the second iteration of the algorithm, i.e. $tcit = 2$.

If a vehicle is defined as probe vehicle, then the system may extrapolate the planning horizon time, t_{max}, until certain arbitrary limit, to ensure that such vehicle may cross the stop line even after the end of the planning horizon $actualt + t_{max}$, and manage the travel time to be estimated by $vehtt_{i,j} = endt_{i,j} - arrt_{i,j}$. Meanwhile, non-probe vehicles that don't cross the stop line within t_{max} will not have their departure time estimated at a later point. This is because arrivals way later than t_{max} will not influence the estimations of travel time within t_{max}, and may slow down the system with

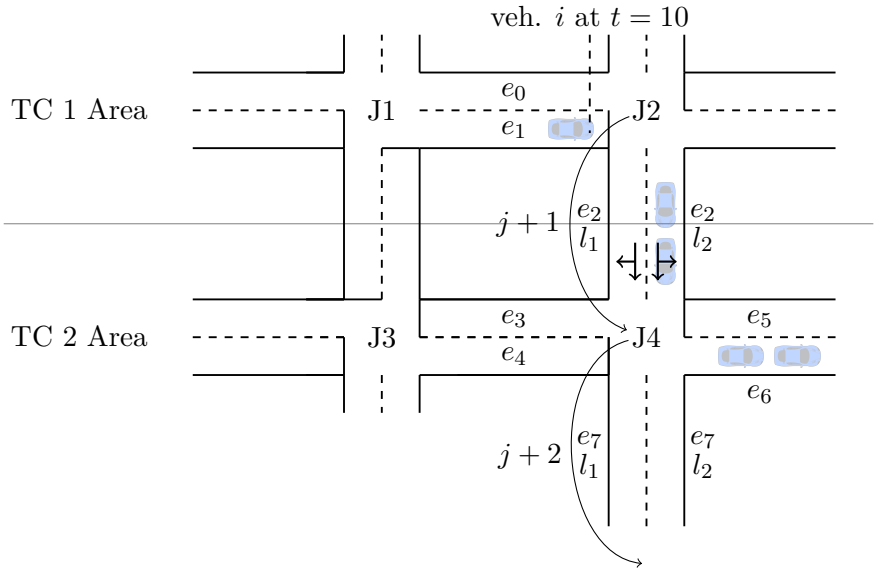


Figure 5.23: The lane vehicle i will use on its next edge $(j+1)$ is the one that enables it to go to its after next edge $(j+2)$ and has no blockage and preferably shorter queue, which is the case of lane l_1.

unnecessary calculations. The conditions for a vehicle to be a probe vehicle for its movement group mg of lane l are in the following ordered list.

1. A vehicle i on its j^{th} junction/edge/lane of its route must be generated using inflow profile, discharged from an upstream junction or one of these conditions but from last time step, in other words, it needs to have a value for the attribute $arrt_{i,j}$.

2. Vehicle's arrival time on lane $arrt_{i,j}$ must be after the time of algorithm update, $actualt$, and before $actualt + t_{max}$.

If a vehicle fulfil the conditions above mentioned for certain lane, l, and mov. group, mg, then the next step is to classify it according to its arrival time and find which range, rge, which is the time window of the division of the planning horizon, vehicle i will be defined as probe vehicle and added into the list $probes_{l,mg,rge}$.

For example, if actual time is $actualt = 10$, the length of ranges $lenrge = 10$, and the planning horizon is $t_{max} = 50$, then the end of planning horizon is $actualt + t_{max} = 60$ and there are 5 ranges (rge_0, rge_1, rge_2, rge_3 and rge_4). If we also consider that there will be 6 vehicles on certain lane l_1 within t_max and they are ordered based on their phase start time ($startt_{i,j}$), and three of them (e.g. i_0, i_1, and i_2) use mov. group mg_1 while the others (e.g. i_3, i_4 and i_5) use mg_2. Their arrival times on lane l_1 are: $i_0 = 5$, $i_1 = None$, $i_2 = 20$, $i_3 = 50$, $i_4 = 59$ and $i_5 = 65$. One may notice that the i_1 doesn't

have arrival time, this is because it is either a starting vehicle on lane l or a newly generate Cooperative Automated Vehicle (CAV), what means they don't start at the begin of the edge/lane. Therefore, it is not possible to know the time they were at the distance from the stop line equal to the lane length (but if the CAV was already detected before - generated on previous time steps on another lane - then they may have values for their arrival on lane). Additionally, vehicle i_5 has arrival after end of the planning horizon, while i_0 had arrived before the *actualt* and didn't cross the stop line yet, so they will not be defined as probe vehicles. Table 5.6 shows the begin and end times of each range (time window) as well as the assigned probe vehicles for each mov. group of lane l.

Range Definition			**Probe Vehicles Lane l_1**	
Range	**Begin Time**	**End Time**	mg_1	mg_2
rge_0	10	20	None	None
rge_1	20	30	i_2	None
rge_2	30	40	None	None
rge_3	40	50	None	None
rge_4	50	60	None	i_3, i_4

Table 5.6: Example of ranges' begin and end times when dividing the planning horizon time and probe vehicles assigned based on their arrival time on the lane.

As we can see in Table 5.6 some ranges have no probes. There is an option in which the system can generate artificial probe vehicles for each mov. group that didn't have itself assigned to a modelled vehicle. However, this considerably increases the number of calculations and slow down the system, what is still acceptable when the traffic controller doesn't control too many junctions and/or model too many vehicles. We will see in Section 5.4 that a deterministic queuing model is used to estimate the travel time at certain range (time window) when there is no probe vehicle.

5.3.4 Define Vehicles to Discharge

This is the first process to happen in the Trip Definition sub-system (except when $tcit = 2$ and in which it occurs after the definition of probes). First, it reads the signal plan of the junction that is being estimated the departures (remember that unsignalized junctions also have a "signal plan" with only one phase with infinity green time, as seen in Section 5.1). Then, for each cycle of t_{max}, denoted as $cyclet_{max}$ (used to extrapolate values when there are remaining probes to cross the stop line after t_{max}), it select the vehicles that will move on the lanes that will get green light during each green phase, denoted ph, from 0 up to $lastph$ of the signal plan. The meaning of "discharge" represents vehicles that will move on a lane that will get green, and it doesn't necessarily mean that a discharged vehicle is a vehicle that will cross the stop

line. A *discharged vehicle* is a vehicle that had its movement during a phase already modelled, crossing or not the stop line. If it doesn't cross the stop line during certain analysing phase, the vehicle will be tried to discharge on the next phase.

As Figure 5.24 shows, it may happen that a lane may get green because of one mov. group green phase but another mov. group using the same lane may not have green. The lane from East have right-of-way for the right-turn connection while the through connection do not. However, vehicles will move on the lane because of the right-turn connection until the vehicle going through will stop and block vehicles behind it to take the right-turn, what causes them to be discharged only on the next green phase. That's why it is necessary to model all vehicles on the lane with green (even if they will not have right-of-way). It is also the reason for the additional "green" phase of the mov. group starting at t_{max} (with zero duration), pointed out in Section 5.1, because in case the analysing phase is the last one and the vehicle will not get right-of-way on the lane which will get green. A vehicle i, on a lane that will get green light, is considered to be discharged during the analysing phase if its:

1. phase start time, $startt_{i,j}$, is before the end of analysing the phase when vehicle i didn't have crossed (i.e. $vehc_{i,j} = 0$); and
2. phase end time, $endt_{i,j}$, is before the end of analysing the phase and after the previous phase when vehicle i have crossed already (i.e. $vehc_{i,j} = 1$).

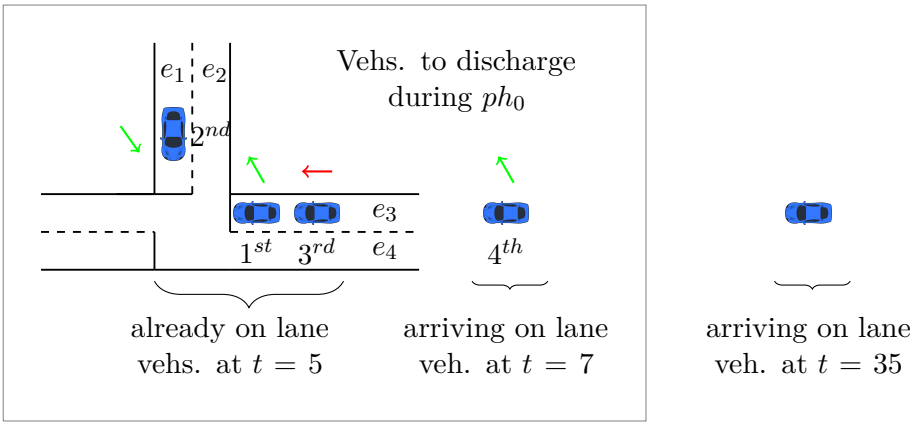


Figure 5.24: Example of selecting vehicles to discharge for green phase ph_0. Vehicles already on lane will be discharged, as well as the earliest one arriving. The last arriving will not manage to arrive before the end of the green phase at $phendt_{ph} = 20$.

We should notice that even if a vehicle was discharged (and crossed the stop line, i.e. $vehc_{i,j} = 1$) in the first algorithm iteration, it will have to be in the list of vehicles to be discharged again, in order to be considered by its vehicles behind and others that may yield to it in the second iteration. However, the modelling is not done twice (this will be discussed in Section 5.3.5).

Additionally, another important task of this process it to order the vehicles that will be discharged during the analysing phase. First, already on lane vehicles are put in the beginning of the order, then arriving vehicles, as it follows:

1. already on lane (which also includes starting on lane) vehicles, from the closest distance to stop line; then
2. arriving on lane vehicles, from the earliest arrival time.

We will see on the next process, in Section 5.3.5, the order of vehicles to be discharged on lane l, denoted $dischlane_l$, is important because a vehicle is not only influenced by the green phase duration of its mov. group, but also by the vehicle in front and other preferential traffic the vehicle needs to yield to.

5.3.5 Estimate the Departure of Discharging Vehicles

This process happens after the *Define Vehicles to Discharge* process, described above in Section 5.3.4 and which gives the phase, its duration and the vehicles to discharge. It models the movement of a vehicle to be discharged that didn't have crossed on previous phases, modelling vehicles along the lane during certain analysing phase, while it also update the queue length prediction of the lane. There two main principles of this movement modelling, they are listed in the following list.

1. The *first principle* defines that the modelling of a vehicle depends on the movement of the vehicle in front (if exists), the signal timings, and vehicles to yield coming from preferential traffic.
2. The *second principle* states that the three values corresponding to the starting movement during the phase ($startt_{i,j}$, $startd_{i,j}$ and $startv_{i,j}$) should be updated through the process for each vehicle and the end values ($endt_{i,j}$, $endd_{i,j}$ and $endv_{i,j}$) reused as start values for the next green phase in case the vehicle will not cross the stop line during the analysing green phase.

The first principle gives us the reason why we need to order the discharging vehicles and estimate the departure one by one, as example of Figure 5.24. It is also important to notice that lane change and overtaking are not modelled, though the final lane a vehicle will use at its departure from the edge is previously estimated given the route, queue prediction and lane status (see Section 5.3.2). This is due the fact that the system assumes the movement of vehicles as deterministic, and once the initial conditions of the vehicle are known as well as its possible actions (decelerate or accelerate with no lane change), there is no need for continuous and fixed interval update of position and speed for all vehicles at the same time. Therefore, the modelling

is *event-based* in which key events represent what vehicles will do during its movement within the analysing green phase until it finally crosses the stop line and go to its next edge.

Figure 5.25 illustrates how the system divide a real-world scenario, containing the events in which it is needed to estimate the state of each vehicle (time instant, distance to stop line and speed). It also shows 4 main cases of such modelling described in the following list.

1. The 1^{st} case is the vehicle departing earliest at stop line, and it represents a vehicle already stopped at its event (0) phase start movement, i.e $startv_{i,j} = 0$, that waits until it gets right-of-way.
2. The 2^{nd} case is the second vehicle to depart at stop line, it is the movement when a vehicle must brake and completely stops until it gets green.
3. The 3^{rd} case is similar to the second case (last vehicle to arrive at stop line), but on this one the vehicle slows down but don't stop completely. Notice that after short braking time, there is a displacement at constant speed due the reaction time of vehicles.
4. The 4^{th} case is the only vehicle that didn't crossed the stop line, it represents vehicles that their phase end values correspond to their final state at the end of green time, opposite to other cases that have phase end values assigned at the departure of the stop line.

The vehicle's key event attributes (1 up to 5) will be denoted as $evtd^{e}_{i,j}$ (distance to stop line in meters), $evtt^{e}_{i,j}$ (time instant in seconds) and $evtv^{e}_{i,j}$ (speed in m/s), where e stands for the event index, as event (0) correspond to the phase start attributes ($startt_{i,j}$, $startv_{i,j}$ and $startd_{i,j}$). One may notice that phase end attributes ($endt_{i,j}$, $endv_{i,j}$ and $endd_{i,j}$) represent values for event (4). In fact, the system only needs the values of phase start and phase end, while the key events are used only for explaining the functionality of the system. In addition, at each event time estimation the system checks if the time at the previous event ($evtt^{e-1}_{i,j}$) was before the next algorithm update time ($actualt + lenrge$) and the analysing event ($evtt^{e}_{i,j}$) happens after $actualt + lenrge$. If so, the vehicle is flagged as a *Vehicle Already on Lane* (from last time step as seen in Figure 5.20) and it is estimated the speed and distance to stop line the vehicle will be at $actualt + lenrge$, in order to have a "memory" of the vehicle state on the next algorithm update. This "memory" correspond to the attributes $nuptd_{i,j}$ and $nuptv_{i,j}$, they will have their values assigned to $startd_{i,j}$ and $startv_{i,j}$, respectively, on the next algorithm update if a vehicle will be in the *Vehicles Already on Lane* list and were generated before or at the last time step.

Before describing the steps for estimating the departure of discharging vehicles, we will introduce three sets of equations that are recursively used throughout the process (acceleration, deceleration and threshold distance).

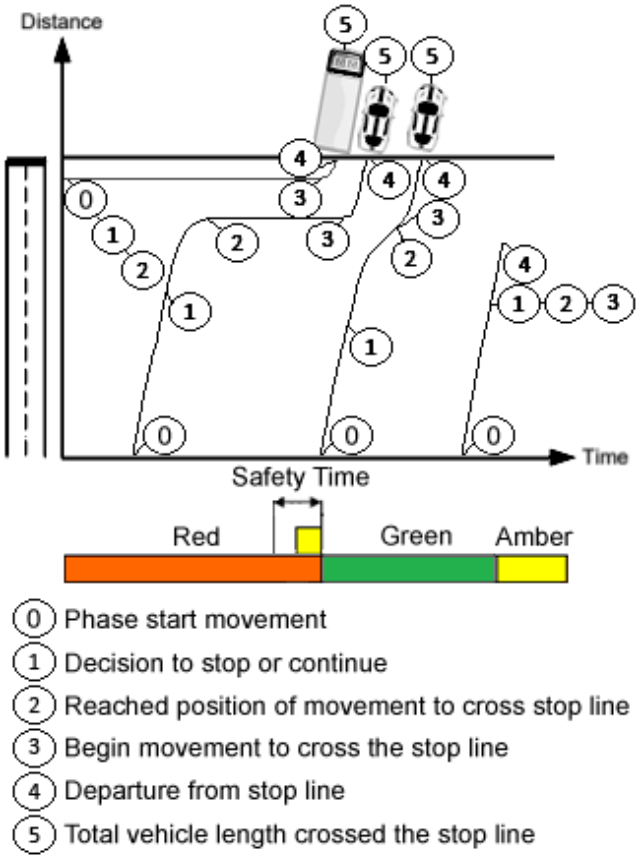


Figure 5.25: Key events to estimate vehicle state in 4 cases.

They are based on the idea that for modelling acceleration or deceleration between events we need to find out when the vehicle will decide to stop. If we assume that a vehicle i being modelled on its j^{th} junction/edge/lane of its route want to stop at its minimum distance gap ($mingap_i$ in meters) from the vehicle in front, it will start slowing down at certain threshold distance from the stop line, which is event (1) $evtd^1_{i,j}$ in meters. This means, vehicle i will (if possible) accelerate from $evtd^0_{i,j}$ (which is equal to $startd_{i,j}$) up to its desired speed on the lane, $vdes_{i,j}$, and have constant speed movement until it starts braking. Based on the simplified Gipps'model presented in [Treiber and Kesting, 2013] we have the threshold distance from the final speed, $finalv$, of the accelerated and/or constant movement before start braking given the distance of acceleration Δd_{acc}, as it follows:

$$decelfinald_{i,j} = \begin{cases} evtd^e_{fronti,j} + len_{fronti} + mingap_i & \text{if } \exists\ vehc_{fronti,j} = 0 \\ mingap_i & otherwise, \end{cases} \tag{5.12}$$

$$\Delta d_{acc} = \frac{vdes_{i,j} - startv^2_{i,j}}{2 \cdot lacc_{i,j}}, \tag{5.13}$$

$$finalv = \sqrt{startv^2_{i,j} + 2 \cdot lacc_{i,j} \cdot \Delta d_{acc}}, \tag{5.14}$$

$$thresdfinalv = finalv \cdot react - \frac{finalv^2}{2 \cdot ldecel_{i,j}} + decelfinald_{i,j}, \tag{5.15}$$

where $decelfinald_{i,j}$ is the deceleration minimum final distance (given by the existence of a vehicle in front, $fronti$, and that didn't cross the stop line), and $react$ is the reaction time. However, if this calculation leads to $startd_{i,j} < \Delta d_{acc} + thresdfinalv$, then vehicle is either braking already, or it will accelerate only to certain speed below its desired speed before start braking. For the former case, we have:

$$thresdinitv = -\frac{startv_{i,j}^2}{2 \cdot ldecel_{i,j}} + decelfinald_{i,j}. \tag{5.16}$$

In the latter case, we need to calculate the distance of the acceleration constant movement (due reaction time) between acceleration and deceleration events given by:

$$\begin{aligned}
a &= 4 \cdot ldecel_{i,j}^2 \\
b &= -(2 \cdot lacc_{i,j} \cdot ldecel_{i,j}^2 \cdot react^2 \\
&\quad + 4 \cdot ldecel_{i,j}^2 \cdot startd_{i,j} \cdot 2 \cdot ldecel_{i,j} \\
&\quad + 4 \cdot ldecel_{i,j}) \\
c &= 4 \cdot startv_{i,j} - 4 \cdot ldecel_{i,j}^2 \cdot startd_{i,j}^2 \\
&\quad + 4 \cdot ldecel_{i,j} \cdot startd_{i,j} - 1 \\
\Delta d_{acconst} &= \max\left(\frac{-b + \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}, 0\right), \qquad (5.17) \\
thresdpartv &= \max\left(startd_{i,j} - \Delta d_{acconst}, decelfinald_{i,j}\right). \qquad (5.18)
\end{aligned}$$

The threshold distance, which is the event (1) in Figure 5.25, is calculated as:

$$evtd_{i,j}^1 = \begin{cases} thresdfinalv & \text{if } \Delta d_{acc} + thresdfinalv <= startd_{i,j} \\ thresdinitv & \text{if } thresdinitv >= startd_{i,j} \\ thresdpartv & otherwise. \end{cases} \tag{5.19}$$

For the acceleration and deceleration movements we derive two different sets of equations from the fundamental equations of constant translational acceleration in a straight line from Physics [Johnson, 2001]. One set when vehicle i will be accelerating between key events e and $(e-1)$ given by Equations 5.23, 5.25 and 5.26), constrained by the maximum displacement, $maxd$, and the available time for the accelerate $maxt_{acc}$:

$$maxd = evtd_{i,j}^{e-1} - accfinald_{i,j}, \tag{5.20}$$

$$maxt_{acc} = \min\left\{\frac{vdes_{i,j} - evtv_{i,j}^{e-1}}{lacc_{i,j}}, acct_{exp}\right\}, \tag{5.21}$$

in which $accfinald_{i,j}$ is the acceleration minimum final distance and $acct_{exp}$ is the expected time for the acceleration. Both variates according to the event number and the context. The distance of accelerated movement, Δd_{acc}, also depends on the difference of the speed at the beginning of event e, $evtv_{i,j}^{e-1}$, and the desired speed $vdes_{i,j}$:

$$\Delta d_{acc} = \min \left\{ \frac{vdes_{i,j}^2 - (evtv_{i,j}^{e-1})^2}{2 \cdot lacc_{i,j}}, maxd, \right.$$
$$\left. evtv_{i,j}^{e-1} \cdot maxt_{acc} + \frac{1}{2} \cdot lacc_{i,j} \cdot (maxt_{acc})^2 \right\}, \tag{5.22}$$

what give us the speed at the end of event e, $evtv_{i,j}^{e}$ and the time spent accelerating Δt_{acc},

$$evtv_{i,j}^{e} = \sqrt{(evtv_{i,j}^{e-1})^2 + 2 \cdot lacc_{i,j} \cdot \Delta d_{acc}}, \tag{5.23}$$

$$\Delta t_{acc} = \frac{evtv_{i,j}^{e}}{lacc_{i,j}}. \tag{5.24}$$

Consequently, the maximum time for the constant movement, $maxt_{const}$, as well as the distance and time of the constant movement, Δd_{const} and Δt_{const} respectively:

$$maxt_{const} = acct_{exp} - \Delta t_{acc},$$
$$\Delta d_{const} = \min\{maxd - \Delta d_{acc}, maxt_{const} \cdot vdes_{i,j}\},$$
$$\Delta t_{const} = \begin{cases} \frac{\Delta d_{const}}{evtv_{i,j}^{e}} & \text{if } evtv_{i,j}^{e} > 0 \\ 0 & otherwise. \end{cases}$$

Finally, the distance, $evtd_{i,j}^{e}$, and time, $evtt_{i,j}^{e}$, of event e is given by:

$$evtt_{i,j}^{e} = evtt_{i,j}^{e-1} + \Delta t_{acc} + \Delta t_{const}, \tag{5.25}$$

$$evtd_{i,j}^{e} = maxd - \Delta d_{acc} - \Delta d_{const}. \tag{5.26}$$

For the set with deceleration, we have Equations 5.28, 5.29 and 5.30. The deceleration minimum final distance, $decelfinald_{i,j}$, is given in 5.12, while the decelerating time Δt_{decel} is constrained by the expected time for the deceleration, $decelt_{exp}$, reaction time $react$ and the time needed to stop completely:

$$\Delta t_{decel} = \begin{cases} \min\left(decelt_{exp} - react, -\frac{evtt_{i,j}^{e-1}}{ldecel_{i,j}} \right) & \text{if } decelt_{exp} >= react \\ 0 & otherwise. \end{cases} \tag{5.27}$$

The time of the event after decelerating, $evtt_{i,j}^{e}$, is the minimum between the beginning until the expected time for the deceleration and the end of decelerating time, while the speed $evtv_{i,j}^{e}$ based on this elapsed time:

$$evtt_{i,j}^{e} = \min\left\{ evtt_{i,j}^{e-1} + react + \Delta t_{decel}, evtt_{i,j}^{e-1} + decelt_{exp} \right\}, \tag{5.28}$$

$$evtv^{e}_{i,j} = evtv^{e-1}_{i,j} + ldecel_{i,j} \cdot \Delta t_{decel}. \tag{5.29}$$

For the distance of the event $evtd^{e}_{i,j}$, it depends on the expected time for the deceleration ($decelt_{exp}$) and it is calculated in three different ways as the calculations always add the movement during reaction time:

$$evtd^{e}_{i,j} = \begin{cases} deceldfinalt & \text{if } decelt_{exp} >= react \\ deceldpartt & \text{if } 0 < decelt_{exp} < react \\ evtd^{e-1}_{i,j} & otherwise, \end{cases} \tag{5.30}$$

$$deceldfinalt = \max \Bigg\{ evtd^{e-1}_{i,j} - \Bigg[evtv^{e-1}_{i,j} \cdot react + \frac{(evtv^{e}_{i,j})^2 - (evtv^{e-1}_{i,j})^2}{2 \cdot ldecel_{i,j}} \Bigg], decelfinald_{i,j} \Bigg\}, \tag{5.31}$$

$$deceldpartt = \max \Bigg\{ evtd^{e-1}_{i,j} - \Bigg[evtv^{e-1}_{i,j} \cdot \left(evtt^{e}_{i,j} - evtt^{e-1}_{i,j} \right) \Bigg], decelfinald_{i,j} \Bigg\}. \tag{5.32}$$

If we compare the deceleration movement of the 2^{nd} and 3^{rd} cases in Figure 5.25, we see that the 2^{nd} had longer $decelt_{exp}$ than the necessary to stopped completely, while the 3^{rd} just slowed down due to braking time shorter than the needed to fully stop. On the other hand, the 4^{th} vehicle had an $acct_{exp}$ constrained by the end of green time which made not possible to continue the movement. The 2^{nd} case vehicle also have its $decelfinald_{i,j} = evtd^{e}_{fronti,j} + len_{fronti} + mingap_i$, because it exists a vehicle in front of i, $fronti$, and it needs to stop behind it with $mingap_i$. Figures 5.26 and 5.27 show the steps throughout this process, being Figure 5.26 for key events 0 to 2 (braking movement) while Figure 5.27 for 3 to 5 (accelerating movement). They are connected through the connectors on the bottom right side of Figure 5.26 and on the top middle of Figure 5.27. Both figures illustrates the procedure for estimating the departure of vehicle i on its j^{th} junction/edge/lane that belongs to the ordered list of vehicles to be discharged on lane l, $dischlane_{l,loi}$, which have indexes called lane order index, loi, for each vehicle in the list. The $fronti$ is the vehicle in front o vehicle i, i.e. $i = dischlane_{l,loi}$ and $fronti = dischlane_{l,loi-1}$.

In Figure 5.26, step 1 initialize the lane order index while 2-3 take each vehicle in the $dischlane_{l,loi}$ and check if it is already known (from previous discharges) that the vehicle will cross the stop line. If vehicle will cross, then it is not necessary to estimate the departure again and the next vehicle is

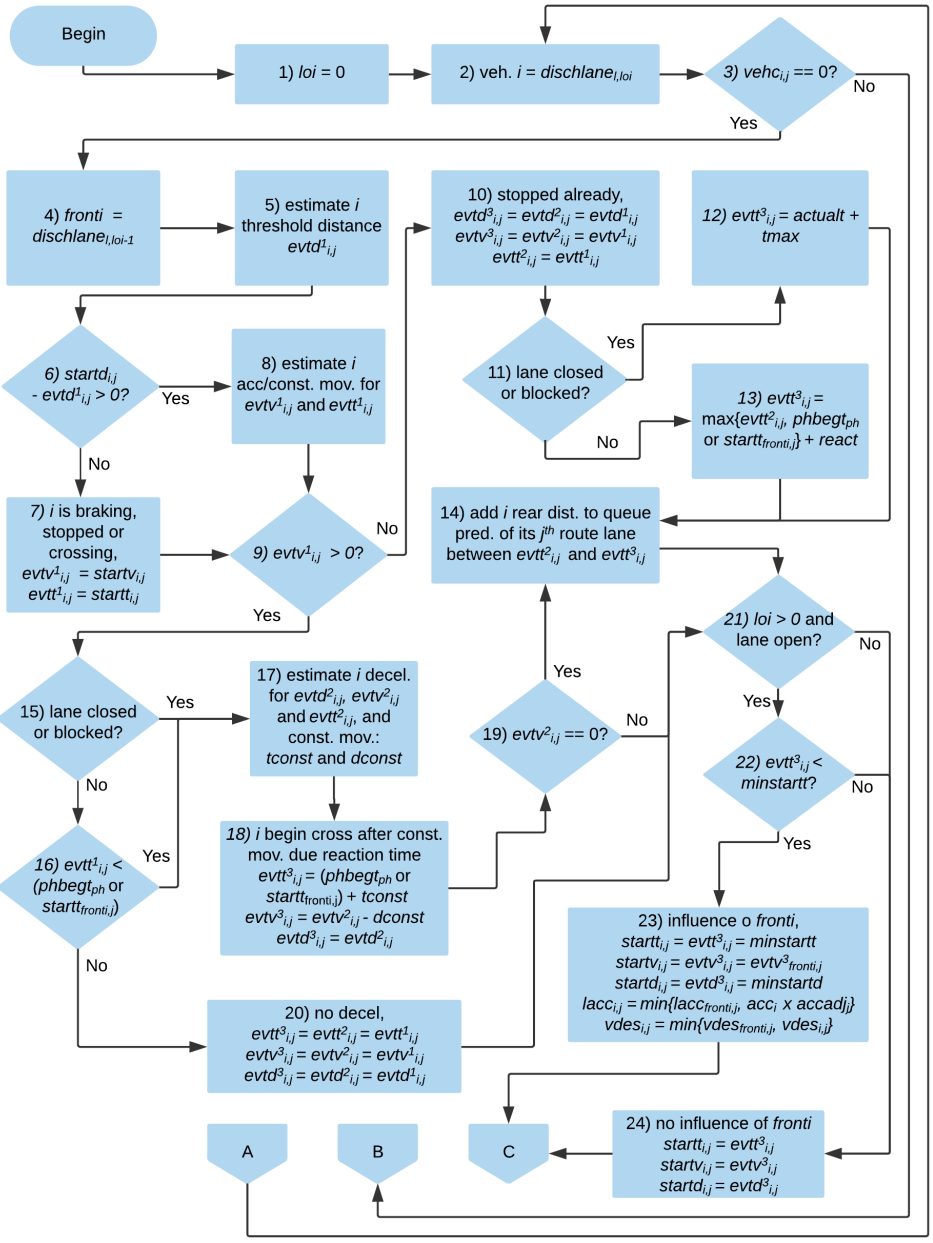


Figure 5.26: Actions taken during the process of estimate the departure of discharging vehicles for key events 0 to 2.

selected, otherwise goes to step 4 onwards. Steps 4-8 defines the vehicle in front of vehicle i (if any), then using Equation 5.19 finds the distance of event (1) that if it is shorter than event (0) then vehicle will have accelerated/constant movement towards event (1), otherwise vehicle is already braking, stopped or crossing. Steps 9-20 check if vehicle is either already stopped (speed at event 1 is zero), or if it needs to decelerate from event (1) by checking if time at event 1 is before the vehicle in front starts its movement to cross $startt_{fronti,j}$, or the begin time of the green (of vehicle i mov. group, or the

lane is blocked/closed). This means that if a vehicle reaches the position where it needs to start braking before vehicle in front start moving or the green start, it will have to brake. After that, values for event (2) are defined, but if there is deceleration it is also estimated a constant movement after it cause by the reaction time, given by *tconst* and *dconst*. The estimations for event (3) time are the maximum between the starting time of $fronti$ or begin green and the event (2), as vehicle may arrive at event (2) after that, or the end of the planning horizon time if lane is closed. If the speed at event (2) is zero, then it accounts the vehicle rear distance to the queue prediction of lane l between the time of event (2) and event (3). Steps 21-24 occurs only if the vehicle is not the first one to discharge (i.e. exists a $fronti$), in which if the time at event (3) is before a minimum starting time $minstartt$, then vehicle i will have to correct its event (3) values and i will be following $fronti$ (i.e. at least same speed, desired speed and lane acceleration). The minimum values are based on the fact that stopped vehicles will start accelerating after the reaction time, *react*, and moving vehicles will try to maintain a gap time, $lgapt_l$, between each other at the speed limit, denoted $lmaxv_l$, on lane l with length $llen_l$:

$$minstartt = \begin{cases} evtt^3_{fronti,j} + lgapt_l \cdot \left(\frac{evtv^3_{fronti,j}}{lmaxv_l}\right) & \text{if } evtv^3_{fronti,j} > 0 \\ evtt^3_{fronti,j} + react & otherwise, \end{cases} \tag{5.33}$$

$$addist = \begin{cases} lgapt_l \cdot lmaxv_l \cdot \left(\frac{evtv^3_{fronti,j}}{lmaxv_l}\right) & \text{if } evtv^3_{fronti,j} > 0 \\ mingap_i & otherwise, \end{cases} \tag{5.34}$$

$$minstartd = \min\left\{evtd^3_{fronti,j} + len_{fronti} + addist,\ llen_l\right\}. \tag{5.35}$$

Continuing the flowchart in Figure 5.27, there is one entry point related to the estimating the departure (C), because (B) is used when vehicle has crossed already and goes direct to step 62 that increase the lane order index, and then use connector (A) back to step 2. Steps 25-27 happen when the mov. group of the vehicle i or $front$ will not get green in the analysing phase and it will not cross, finishing the modelling. Steps 28-31 will estimate the acc./const. movement for event (4) and if vehicle cannot reach the stop line either because the green time will finish before it or $fronti$ will not cross, then i will not cross and finish modelling; on the contrary it defines i next lane, mov. group, connection and initializes the next lane queue delay $qdt = evt^4_{i,j}$ and the preferential delay $pdt = None$. Here lies the importance of how to input the green phases for a movement group with permissive green followed by a protected green (e.g. phase extension of certain signal group). We model both greens as a different phase, in which there is no amber time for the permissive while no safety time for the protected one. If we consider the case of the last vehicle in Figure 5.25, the system models the vehicle until the end of the permissive green and then when modelling the protected green vehicle would cross without stop. On the other hand, if the next phase is

not protected green, then the vehicle will have only modelled its braking and stop during the analysing phase.

The loop 31-33 adds queue delay for vehicle i for each queue range time qrg the next lane is full due queue (the precision of the queue prediction is every $lenrge$ seconds for junctions not modelled by the TC and one step length Δt for modelled ones). Steps 34-35 check if it is needed to reduce the speed at departure from stop line when vehicle need to wait due the next lane full queue. The loop 36-40 represents the decision if vehicle i will cross during amber time, notice that it is allowed only one vehicle per amber time. Steps 41-44 occurs when vehicle may cross the stop line, and it either doesn't need to yield to preferential traffic or it had but even with preferential delay, pdt, the vehicle could cross. Then vehicle is added to the list of departures on lane l during phase ph of vehicle mov. group, mg, denoted $deps_{l,mg,ph}$, and estimate event (5) as well as arrival time and phase start on the next lane/edge. Step 46 happens when the modelling is for a probe vehicle and the modelling is after then planning horizon t_{max}. Therefore, as it is known the departures of each phase within t_{max} (from $deps_{l,mg,ph}$) and non probes are not modelled after t_{max}, the delay for preferential traffic, pdt, is given by the capacity of dicharge of the connection vehicle i is taking, discussed in Section 5.3.6. Steps 47-61 calculate the preferential delay within t_{max}. The system estimates when vehicle i will be able to cross, bc, by analysing if the gap between event (5) of vehicles on the preferential traffic is longer than a reference gap, rg (connection average critical gap $avecr_{conn}$ if i is the first to discharge or the follow-up time fup_{conn} otherwise), and does it for each lane to yield, yl, in a list of lanes to yield ($yield2lanes$ and where the last one is $lastyl$). In addition, it always stores the highest delay time, $spdt$, as well as the last vehicle index to yield to $yloi_{yl}$ on each lane that correspond to a preferential vehicle $prefi$ and where the last one is $lastpi_{yl}$, in order to find a suitable gap again for all lanes to yield if the $pdt > spdt$.

5.3.6 Update Connections' Capacity and Delay

This process estimates the delay, $pdt_{conn,ph}$, that a vehicle will experience when taking a connection based on the connection's capacity of discharge during certain green phase, ph, of its mov. group, mg. The capacity, $cap_{conn,ph}$, is calculated in three different ways according to the system-level flow thresholds (λ_{low} and λ_{mid}), and the flow of vehicles departing on a lane to yield yl for its mov. group, $ylmg$ at same phase, denoted $\lambda^{deps}_{yl,ylmg,ph}$ (in $vehicles/s$), that is given by the number of vehicles departing, $numdeps_{yl,ylmg,ph}$, in the list $deps_{yl,ylmg,ph}$, as it follows:

$$\lambda^{deps}_{yl,ylmg,ph} = \frac{numdeps_{yl,ylmg,ph}}{phendt_{ph} - phbegt_{ph}}. \tag{5.36}$$

If $\lambda^{deps}_{yl,ylmg,ph} < \lambda_{low}$, then the dynamic capacity on connection $conn$ assumes *Negative Exponential Distribution* of the headways on the preferential traffic.

Equation 5.37 is the same of the capacity for non-signalized junctions with random arrivals [FHWA, 2001], due the fact that vehicles also yield to others in order to cross.

$$cap_{conn,ph} = \frac{\lambda^{deps}_{yl,ylmg,ph} \cdot e^{-\lambda^{deps}_{yl,ylmg,ph} \cdot avecr_{conn}}}{1 - e^{-\lambda^{deps}_{yl,ylmg,ph} \cdot fup_{conn}}}, \tag{5.37}$$

where $avecr_{conn}$ is the critical gap time and fup_{conn} is the follow-up time.

If $\lambda^{deps}_{conn,ph} < \lambda_{mid}$, Equation 5.38 assumes a *Dichotomized Distribution* (some vehicles in platoon and some not) of the arrivals on the preferential traffic, which is also used to estimate the capacity for non-signalized junctions [FHWA, 2001], what give us:

$$cap_{conn,ph} = (1 - \lambda^{deps}_{yl,ylmg,ph} \cdot h_{min}) \cdot \frac{\lambda^{deps}_{yl,ylmg,ph} \cdot e^{-\lambda^{deps}_{yl,ylmg,ph} \cdot (avecr_{conn} - h_{min})}}{1 - e^{-\lambda^{deps}_{yl,ylmg,ph} \cdot fup_{conn}}}. \tag{5.38}$$

For the both cases when $\lambda^{deps}_{conn,ph} < \lambda_{mid}$, it is important to notice that this capacity cannot be higher than the fixed capacity of the connection, $cap_{conn,sf}$, which represents its saturation flow. In case the connection yields to more than one lane and/or mov. groups, then it is used the lowest capacity for the calculation of the connection delay for a vehicle i, $pdt_{i,conn,ph}$. This connection delay is constrained by its capacity $cap_{conn,ph}$ and given by:

$$pdt^{lowmid}_{i,conn,ph} = \max\{conndt - \max\{slarr_i - refhdwy,\ 0\},\ 0\}, \tag{5.39}$$

$$pdt^{lowmid}_{i,conn,ph} = \min\{pdt^{lowmid}_{i,conn,ph}, phendt_{ph} - slarr_i\},\ phendt_{ph} > slarr_i, \tag{5.40}$$

where *conndt* is delay without considering the time interval between arrivals (or begin green time and actual arrival of vehicle), the stop line arrival time *slarr*, and the reference time for headway *refhdwy*. They are calculated as it follows:

$$conndt = \begin{cases} 1/cap_{i,conn,ph} & \text{if } cap_{conn,ph} > 0 \\ \max\{phendt_{ph} - slarr_i,\ 0\} & \text{if } cap_{conn,ph} = 0, \end{cases} \tag{5.41}$$

$$refhdwy = \begin{cases} slarr_{fronti} & \text{if } \exists\ fronti \\ phbegt_{ph} & \text{if } signalized\ junction \\ slarr_i & otherwise, \end{cases} \tag{5.42}$$

in which Equation 5.40 is valid only if this process is called by the *Estimate the Departure of Discharging Vehicles* process, from where $fronti$ is the vehicle in front. This is because when called by the *Estimation of Travel Times* process, the connection delay must consider the connection delay of discharging even on the next phases as it is just an estimation of a connection delay and not modelling, while in the *Estimate the Departure of Discharging Vehicles* the

vehicle is modelled again on the next phase. Equation 5.41 shows that if the capacity is zero, the maximum delay is the end of the green time, $phendt_{ph}$.

If $\lambda^{deps}_{conn,ph} >= \lambda_{mid}$, the case is different as there are too much traffic on the preferential lanes. It is expected then the vehicle may cross only after the last one on the preferential traffic crosses. However, as it is an extrapolation of the planning horizon t_{max}, we need to assume the departure but after the suitable cycle, $cyclet_{max}$, that enables the vehicle to depart considering same future conditions of the planning horizon:

$$pdt^{high}_{i,conn,ph} = \max\left\{(evt^4_{lastpi,j} + cyclet_{max} \cdot t_{max}) - slarr_i,\ 0\right\}, \qquad (5.43)$$

where $lastpi$ is the last vehicle on the preferential traffic of all lanes to yield on its j^{th} junction/edge/lane of its route, and evt^4. In case the connection yields to more than one lane and/or mov. groups, then it is used the last vehicle to depart from all lanes to yield.

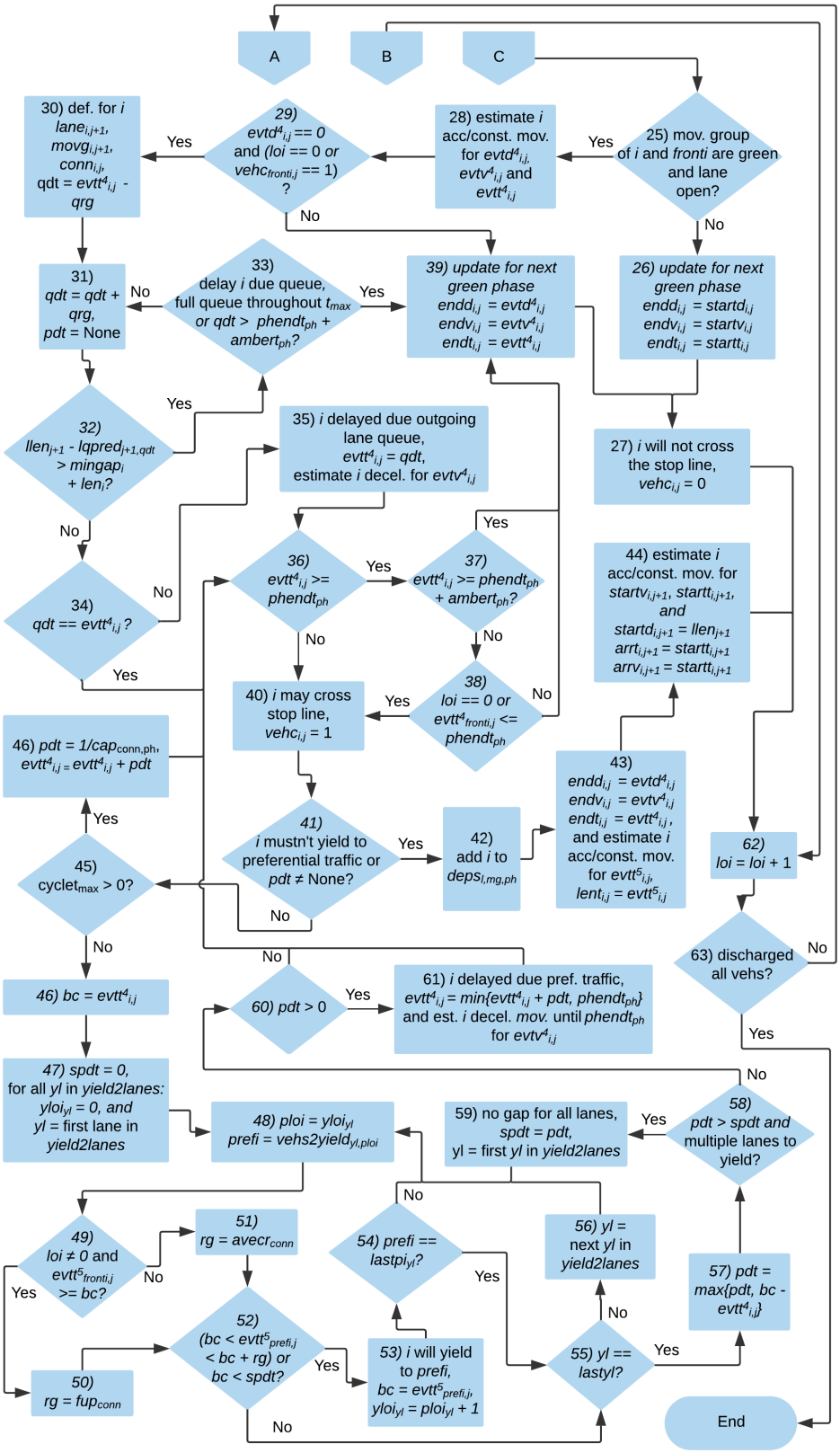


Figure 5.27: Actions taken during the process of estimate the departure of discharging vehicles for key events 3 to 5.

5.4 Estimation of Travel Times

This sub-system estimates the travel time on the controlled edges by the Traffic Controller (TC) per range (time window), rge, of the division of the planning horizon t_{max}. The value for an edge is the average of the travel times of its lanes and movement groups, as the assumption is that the travel time is different for each lane of the edge and especially per movement group. However, sharing between traffic controllers the travel time per mov. group of each lane of the network and for each range (time window) may require too much data to be transmitted. In other words, the system calculates travel times at the precision of each range for each movement group per lane but only the average for the edge will be shared. The inputs, processes and outputs of this sub-system are seen in Figure 5.28.

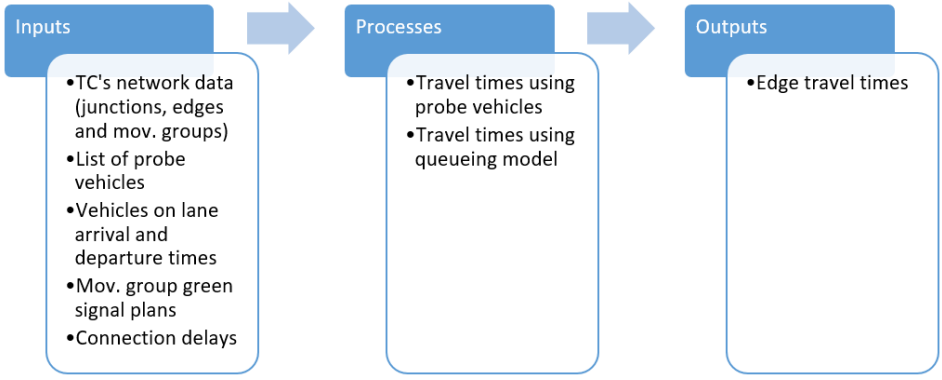


Figure 5.28: Inputs and outputs of the Estimation of Travel Times sub-system.

5.4.1 Travel Times Using Probe Vehicles

The estimation using probe vehicles takes advantage of the fact that vehicles are modelled from their arrival on the edge until their departure time. Therefore, the edge, k, travel time for range, rge, denoted as $edgett_{k,rge}$ is given by:

$$edgett_{k,rge} = \frac{\sum vehtt_{i,j}}{numprobes_{l,mg,rge}} \quad \forall\, i \in probes_{l,mg,rge}, \quad \forall\, l \in lanes_{e_k} \text{ and } \forall\, mg \in movgs_{e_k}, \tag{5.44}$$

where l is the index of the lanes of edge k and mg the mov. group, $lanes_{e_k}$ is the list of edge e_k lanes and $movgs_{e_k}$ its mov. groups, while $numprobes_{l,mg,rge}$ is the number of probes. The probe vehicle, i, is in $probes_{l,mg,rge}$, which correspond to its j^{th} junction/edge/lane of its route and edge k, where:

$$vehtt_{i,j} = endt_{i,j} - arrt_{i,j}. \tag{5.45}$$

5.4.2 Travel Times Using Queueing Model

When probe vehicles are not assigned to a certain range rge of an edge, then a modified version of the deterministic queueing model presented in Equation 4.8 is used. The choice of this model is due the fact that as we model vehicle arrivals and departures, as well as the initial queue is also modelled by the vehicles already on lane, we use our traffic theoretic model as input for this queueing model, in which:

$$q(t) = q(actualt) + A(t) - D(t), \tag{5.46}$$

$$adpv(rge) = \frac{1}{A(rgendt_{rge})} \int_{actualt}^{rgendt_{rge}} q(t)dt, \tag{5.47}$$

where

$q(t)$ is the queue length at time instant t,

$rgendt_{rge}$ is the end time of the range rge being estimated the travel time,

$A(t)$ is the cumulative number of arrivals from the current time of updating the algorithm $actualt$ until t,

$D(t)$ is the departures under continuous presence of vehicle queue from the current time of updating the algorithm $actualt$ until t, and

$adpv(rge)$ is the average delay of vehicles queuing during the time period $[actualt, rgendt_{rge}]$.

The model needs the "arrivals" and "departures" from the stop line what we don't really have in some cases if the vehicle is not a probe vehicle and it couldn't depart from stop line before the end of the planning horizon. To solve this, we look into how it models the queue. Vehicles are considered to stack at the stop line, the so called vertical queue. Therefore, the arrival time means the time vehicle would arrive at stop line without any influence of vehicle in front or signal plan, $qmodelarr_{i,j}$, given by:

$$\begin{cases} qmodelarr_{i,j} = arrt_{i,j} + \frac{llen_l}{(lmaxv_l \cdot vfac_i)} & if\ \exists\ arrt_{i,j} \\ qmodelarr_{i,j} = nupt_{i,j} + \frac{nupd_{i,j}}{(lmaxv_l \cdot vfac_i)} & otherwise, \end{cases} \tag{5.48}$$

where $arrt_{i,j}$ is vehicle's arrival time on the edge (entering it), $nupd_{i,j}$ is vehicle's distance when being generated (as $startd_{i,j}$ is updated with new values), $llen_l$ is the lane length (in metres), $lmaxv_l$ the lane maximum speed, and $vfac_i$ vehicle i speed factor. The vehicle i is classified as part of the initial queue, $q(actualt)$, if $qmodelarr_{i,j} <= actualt$, otherwise it is assigned to the respective arrival on range, rge, when $qmodelarr_{i,j} >= rgstartt_{rge}$ and $qmodelarr_{i,j} < rgstartt_{rge+1}$, in which $rgstartt_{rge}$ is the begin time of the range rge. For the departures, the system accounts vehicle departure

only if the vehicle crosses the stop line, the value used is the phase end time $endt_{i,j} = evt^4_{i,j}$.

Equation 5.47 only estimates the delay due the queue, we still need to calculate the delay due red light for a vehicle arriving on the edge at the begin time of the range rge, denoted $rgstartt_{rge}$, and also the connection delay when yielding to preferential traffic. As the idea is to estimate the travel time for each range but there is no probe vehicle, what is possible is to estimate these delays if a vehicle would enter the edge at begin time of the range, $rgstartt_{rge}$. In this way, we need to calculate the travel time without any delays, $llentt_l$, in order to find the arrival time at the vertical queue, $vtarr_{l,rge}$:

$$llentt_l = \frac{llen_l}{lvmax_l}, \tag{5.49}$$

$$vtarr_{l,rge} = rgstartt_{rge} + llentt_l. \tag{5.50}$$

For the traffic light delay, $tsdt_{l,mg,sph}$, we need to find a suitable phase, sph, which is the first phase the vehicle may cross and it is based on the arrival time at stop line:

$$tsdt_{l,mg,sph} = \max\{begint_{mg,sph} - vtarr_{l,rge}, 0\}. \tag{5.51}$$

For the calculation of the queue delay, it is necessary to know the queue at the time a vehicle arriving at $rgstartt_{rge}$ would experience. This is given by a suitable range, $rgevtar$, that is the first possible range of the arrival time at stop line. Finally, the travel time for the movement group mg of lane l at range rge is estimated by:

$$lanemovtt_{l,mg,rge} = llentt_l + tsdt_{l,mg,sph} + adq(rge) + pdt_{conn,sph}, \tag{5.52}$$

where $adq(rge) = adpv(rge) \cdot q(rgevtar)$ represents the average queue delay at rge, while $pdt_{conn,sph}$ is the connection delay (discussed in Section 5.3.6) using $slarr = vtarr_{l,rge} + adq(rge)$, because it considers the arrival time at the stop line as the time the vehicle may depart after waiting for the queue delay. The travel time on the edge, $edgett_{k,rge}$, is the average travel time for all mov. groups of the lanes belonging to the edge k.

5.5 Traffic Controllers Knowledge Sharing

This sub-system is responsible for the definition of the content and format of the messages exchanged between traffic controllers as well as getting information from these messages once it is known their formats. As it is required an interface for the real exchange of messages, this sub-system only focus on what should be shared and when, while how to do it is out of scope. Figure 5.29 shows the inputs, processes and outputs of this sub-system.

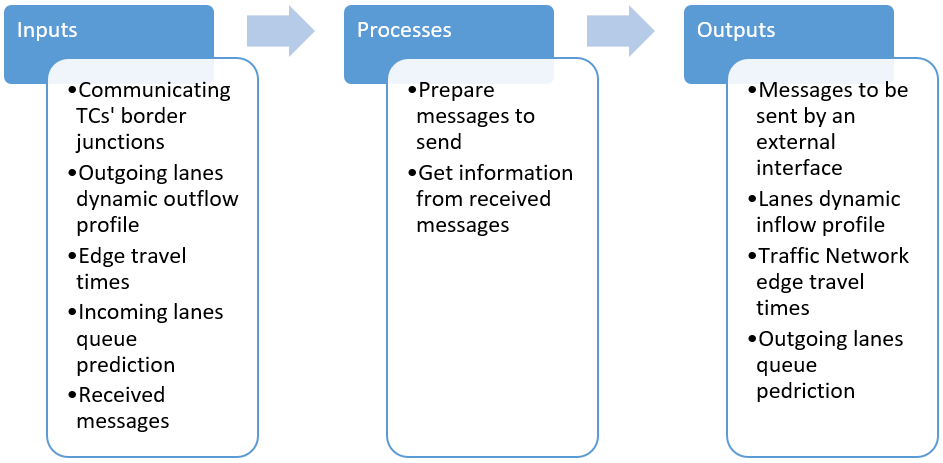


Figure 5.29: Inputs and outputs of the Traffic Controllers Knowledge Sharing sub-system.

Figure 5.30 illustrates the format of the exchanged messages. The information is valid from certain initial time (*actualt* corresponding to the time the algorithm updated) and specific for each range (time window), rge, of the planning horizon t_{max}. As the length of the ranges, $lenrge$, is fixed for the whole system we only need to know from which time the received information is valid (t_{max} is not necessary). Notice that the messages are junction-based in order to reference which TC border junction would need to send information to which TC border junction:

- source junction ID ($jctID$) which is the identification code of the source junction within the traffic network that send a message
- destination junction ID ($jctID$) which is the identification code of the destination junction within the traffic network

The **lane inflow profile** contains the inputs for the *Vehicle Generation* sub-system, and only for the outgoing lanes that follow the order of junctions to have their arrivals and departures to be estimated (i.e. a junction sends its outflow on the lanes that go to a junction which still needs to have its arrivals and departures to be estimated). This means that such messages are exchanged at each algorithm iteration (twice per update). The information to be shared is the parameters of headways distribution functions:

- lane network identification ($laneID$);
- number of vehicles;
- flow (veh/s);
- mean speed (m/s);
- mean of headways (s); and

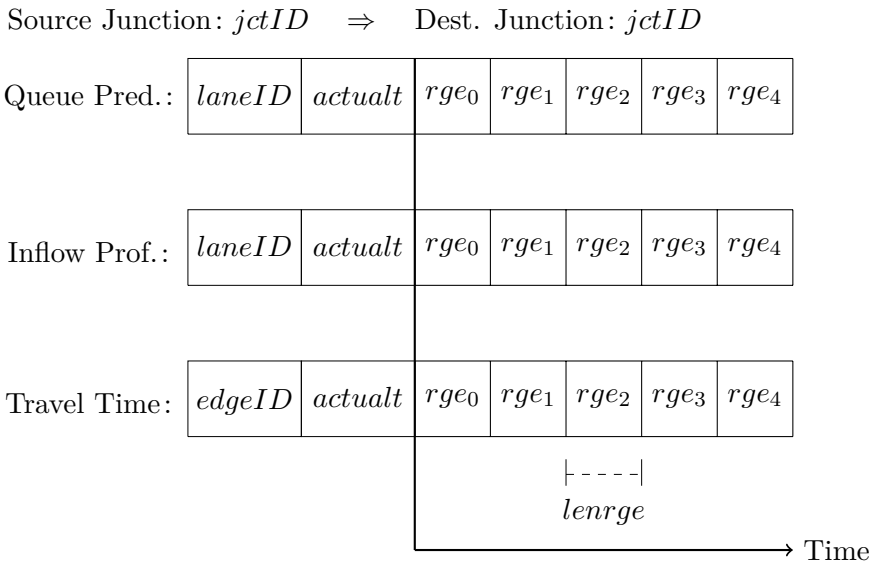


Figure 5.30: Format of exchanged messages.

- standard deviation of headways.

The **lane queue length prediction** is sent only once at the second iteration (i.e $tcit = 2$) as it will be useful only on the next algorithm update. For the predicted queue length for each lane that is common between two connected traffic controllers, the needed data is the rear distance of the last stopped vehicle on each range throughout the planning horizon stored by the *Trip Definition* sub-system:

- lane network identification ($laneID$); and
- queue length (m).

The **edge travel times** (TTs) message must be multicast to all traffic controllers connected on the same Traffic Network (TN) and it is sent only at the second iteration (i.e $tcit = 2$). This is due the fact that the broadcast of travel times to Cooperative Automated Vehicles (CAVs) happen only after all TCs estimate their edges' travel times. It can be sent with any origin junction of the TC as well as destination of each TC to receive, though it is important to avoid sending to different junctions controlled by the same TC, what would be redundant. These travel times are the expected ones per range (time window) if a vehicle arrives on the edge between the interval times of the range:

- edge network identification ($edgeID$); and
- travel time (s).

In general, the main sequence of tasks a Traffic Controller does is receive messages; generate vehicles; estimate departures; estimate travel time; and

send messages. Additionally, receiving messages happen at all time steps, while sending only at algorithm update times. Figures 5.31 and 5.32 show an example of messages received and sent, respectively, between three traffic controllers. We can see that TC1 receives queue prediction of edges 1 and 3 while send the inflow of such edges, meanwhile receives the dynamic inflow from edge 2 but send queue prediction. If we consider J1, this is because for estimating the queue length on edge 1 it needs to know which vehicles will be on the lane (which is also generated by the inflow profile) and such queue prediction is used by J4 to delay or not vehicles if the queue will be full and choose which lane vehicles will go to J1. For the case of J2, edge 3 has only one way from J2 to J5, so J2 only needs the queue prediction from J5 and it only needs to send the dynamic inflow profile.

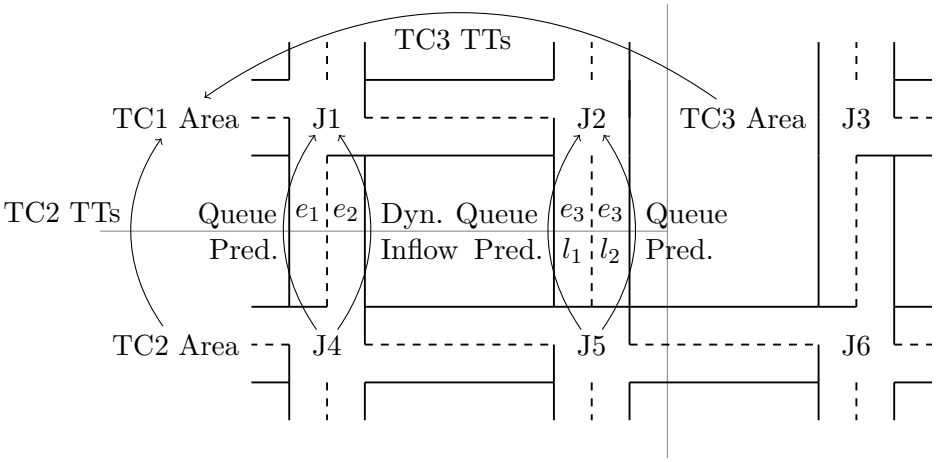


Figure 5.31: Example of received messages by TC1 in a traffic network with 3 connected traffic controllers.

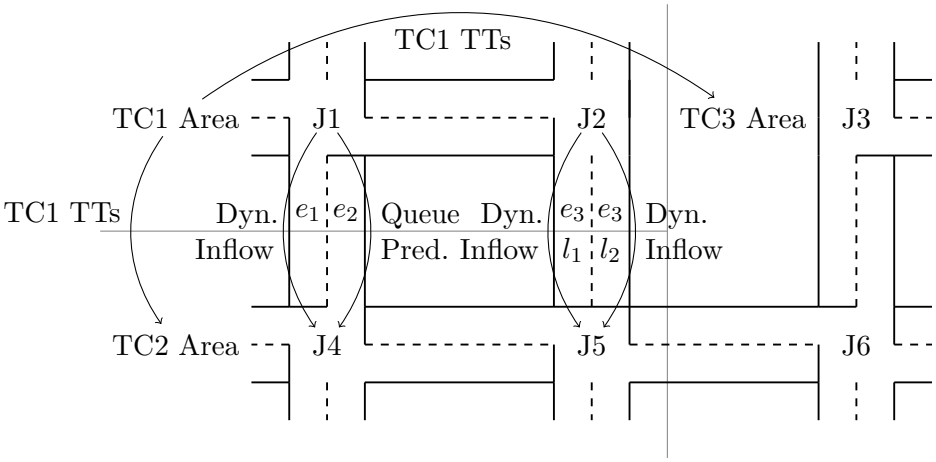


Figure 5.32: Example of sent messages by TC1 in a traffic network with 3 connected traffic controllers.

Chapter 6

Simulation and Results

Although the main focus of this thesis is the development of a local level routing system, in this chapter we evaluate our proposed system in a realistic environment, demonstrating that it works for different penetration rates of Cooperative Automated Vehicles (CAVs). Traffic simulations replicate the traffic environment and can analyse the characteristics of traffic depending on the problem that we want to study. Macroscopic traffic simulations are suitable for traffic flows metrics, as they don't take into account individual vehicle interactions. On the other hand, microscopic simulations are able to accurately model the behaviour of an individual vehicle in its environment [Martinez et al., 2011], this is the type we are looking for as we need to simulate CAVs getting the information from our system and feeding it back with their fixed parameters and state (position and speed) information.

6.1 Simulation Software

We chose the simulation software SUMO [Behrisch et al., 2011] due to its open source distribution and its TraCI (Traffic Control Interface) which is used to interact the simulation in SUMO with Python scripts. This enabled us to write the algorithm of our system in Python language that can exchange information with the simulation replicating the real environment of CAVs interacting with non-CAVs and communicating to traffic controllers through Roadside Units (RSUs). Additionally, SUMO has a embedded rerouteing mechanism where we can define vehicles to be equipped with a routing device that changes the route to its destination at run-time at regular intervals. First, the mechanism checks an optimal-route and then uses the current position of the vehicle, its destination and either the current status of the network or defined costs/weights (that can be time-dependent travel times) per edges to compute a new optimal path. After that, the vehicle changes its route to the new path if it is has lower cost than the current one [Codeca et al., 2017].

6.2 Network Simulated

The Luxembourg SUMO Traffic (LuST) presented in [Codeca et al., 2017] is a general purpose mobility scenario that describes a realistic traffic scenario (e.g. avoids gridlocks and unrealistic mobility patterns) containing traffic traces with various traffic densities and mobility patterns. Still according to [Codeca et al., 2017], the City of Luxembourg has a topology comparable to that of many of European cities, consisting of a central "city centre" area, surrounded by different neighbourhoods, which are linked by arterial roads. Another important aspect is that the area is big enough to simulate the congestion patterns of cities, but at the same time small to allow simulations running during a reasonable amount of time.

Figure 6.1 shows the whole LuST scenario topology, the chosen Traffic Network (TN) for the application of our Local Level Routing (LLR) system, and the location of the cooperative junctions (i.e. junctions that have a RSU to communicate with CAVs).

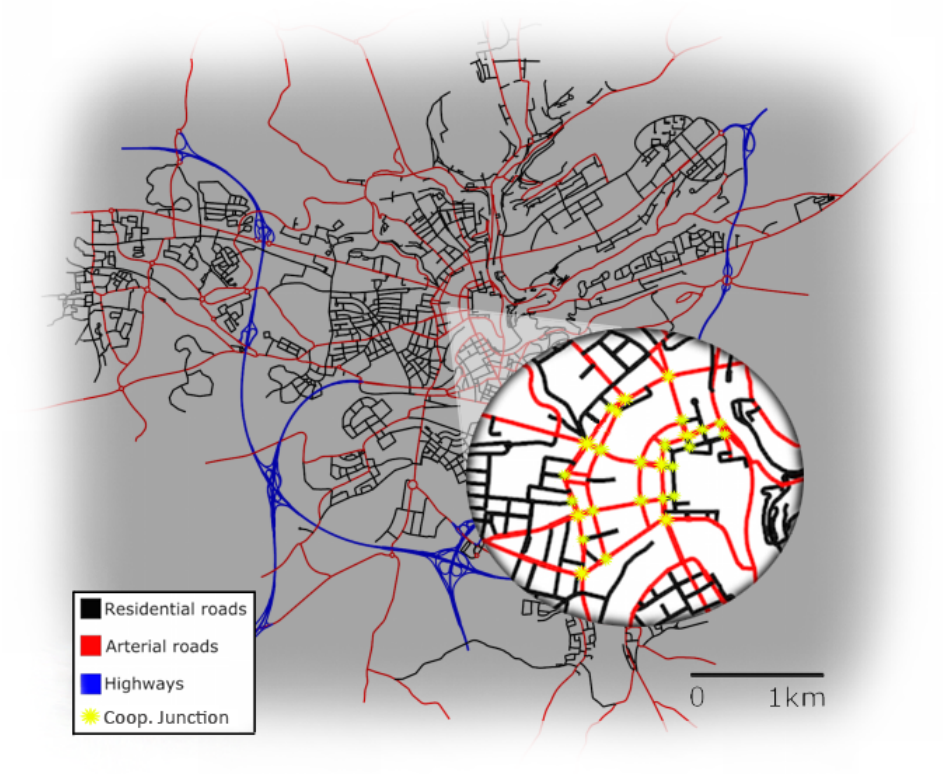


Figure 6.1: LuST scenario topology and chosen area for the application of the local level routing system.

6.2.1 Traffic Demand

Figure 6.2 shows the complete traffic demand in red, composed of buses and the following mobility patterns: a) local (origin or destination, or both inside the city) in blue; and b) transit (both origin and destination outside the city through the highways) in green. There is also a distinction between running vehicles (R) and vehicles that are waiting to be inserted in the simulation (W) [Codeca et al., 2016].

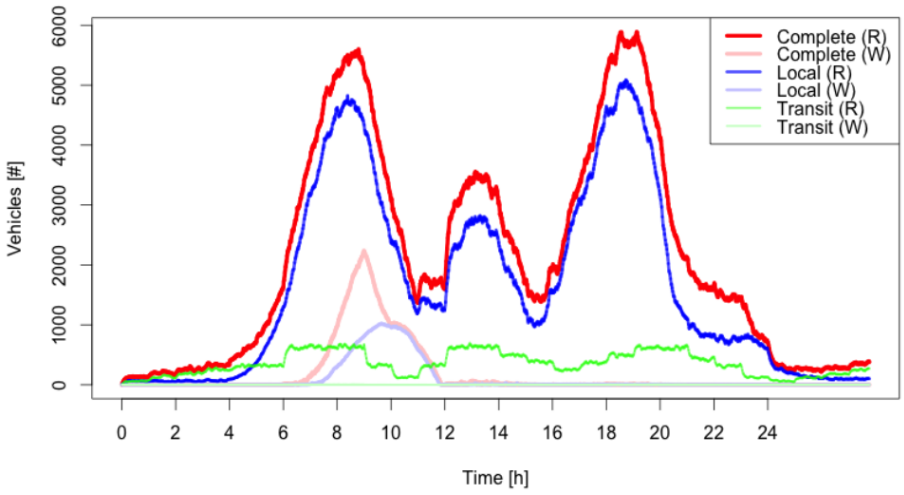


Figure 6.2: Traffic Demand over a day. (R) represents the running vehicles and (W) the ones waiting to enter the simulation at each given time [Codeca et al., 2016].

We will study the influence of our Local Level Routing system during two hours at the evening after peak from 21:30 to 23:30 hours. However, this time range correspond to the time we measure the values, as the simulation (including the function of the proposed system and vehicles routing themselves) starts 30 minutes before the measurements to act as a warm-up time. We expect that at this low flows the objective of finding alternative routes promoting green waves should be achieved, while balancing the load of vehicles through the network due the knowledge of the predicted queue lengths. The network speeds (in m/s), as road colour, and relative flow volumes, as road width, are shown in Figure 6.3 in respect to the traffic demand around 22:30 without vehicles equipped with rerouteing device.

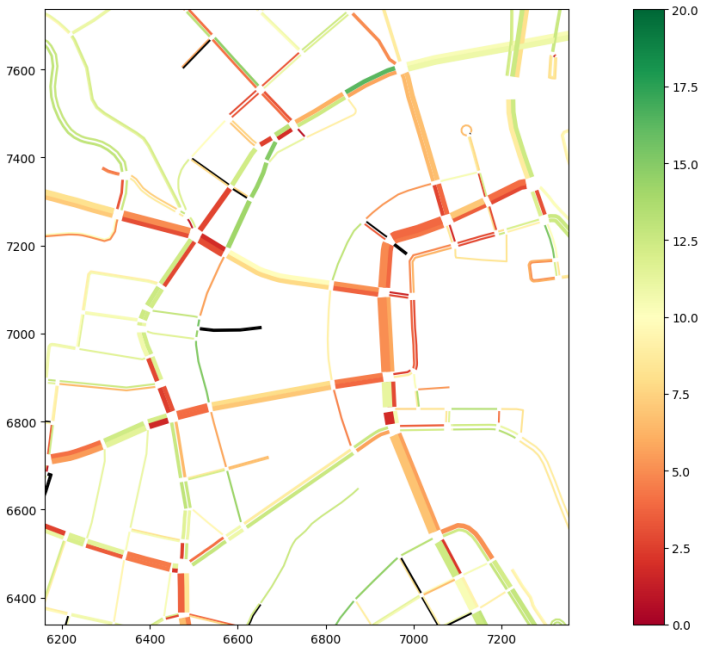


Figure 6.3: Network speeds (in m/s) and road width as relative flow volumes at around 22:30 without vehicles equipped with rerouteing device.

6.3 Scenarios

There are two main scenarios in the Luxembourg SUMO Traffic (LuST) network, each based on the predefined routes (mobility traces) of vehicles throughout the simulation over 24h. One is the Dynamic User Assignment (DUA) (all-or-nothing strategy), in which routes were defined by the time-dependent shortest-path (least time) for each vehicle with static traffic lights but without consideration of other vehicles (empty network). The second is the Dynamic User Equilibrium (DUE) (dynamic version of the Wardrop's user equilibrium principle, see Appendix B.3) that represents several iterations of the DUA considering other vehicles. Additionally, as SUMO enables the control of how many and/or which vehicles can be equipped with rerouteing device, besides the definition of the time-dependent travel times that act as edge costs, we can set a penetration rate of vehicles that may reroute using Hourly Travel Times (HTT), while others continue using their DUA routes. Therefore, the total of 16 scenarios were planned for a sensitivity analysis that tried to capture the performance of our Local Level Routing (LLR) system under seven different Penetration Rates (PRs) of CAVs, against the following base cases:

- same vehicles of the LLR but using only hourly travel times case (HTT);
- best case (DUE); and
- worst case (DUA).

6.3.1 Base Cases

The **Dynamic User Assignment (DUA)** case uses the optimum route between origin and destination in terms of travel time and length on empty network. Although this mobility is not realistic (because it does not consider other vehicles), it is useful for providing a scenario with routes ready to be optimized by rerouteing algorithms. There is no rerouteing during the simulation at this case.

The **Dynamic User Equilibrium (DUE)** case uses the optimum route where each vehicle cannot reduce its travel cost by using a different path. It iteratively optimizes each route, running many simulations and recomputing the DUA over the network. After each iteration, new routes are randomly selected among the best routes and they are added to the set of those available for a vehicle, then a new simulation is performed. There is no rerouteing during the simulation at this case.

The **Hourly Travel Time (HTT)** case uses the DUA routes for all vehicles, but if a vehicle is equipped with a rerouteing device, it uses hourly travel times (i.e discrete values of edge travel times per 1 hour interval) to reroute itself using the same time-dependent discrete travel times shortest-path algorithm (that will be defined in Section 6.4) to find the optimum route at every predefined interval (e.g. 300 seconds). The penetration rates of vehicles with rerouteing device is the same of the proposed case: 10%, 20%, 30%, 50%, 70%, 85%, 100%.

6.3.2 Proposed Case

The **Local Level Routing (LLR)** case uses the DUA routes for all vehicles, but if a vehicle is equipped with a rerouteing device, then it uses the following priority of time-dependent discrete travel times for the shortest-path algorithm (that will be defined in Section 6.4) to reroute itself:

1. edge, k, travel time for each range, rge, of the planning horizon time, t_{max} provided by the LLR system. In addition, to ensure the FIFO property and also provide some updated estimation of the travel time further than t_{max}, for any time after $actualt + t_{max}$ we use the maximum between the average value of the travel times throughout t_{max} and the travel time of the last range, $lastrge$, plus the time interval between algorithm updates, $lenrge$. In other words, $edgett_{k,>lastrge} = \max\{edgett_{k,lastrge} + lenrge, meanedgett_k\}$, where $edgett$ is the edge travel time, $numrges$ the number of ranges and

$$meanedgett_k = \frac{\sum_0^{lastrge} edgett_{k,rge}}{numrges}. \tag{6.1}$$

2. hourly travel times.

Notice that this means that vehicles may use for some edges the travel time provided by our LLR system, while for other edges not controlled by the LLR system the hourly travel times. Additionally, the updated travel times are set individually for each vehicle equipped with rerouteing device within the communication range of the RSUs within the Traffic Network of the chosen area of our LLR system. In other words, if a rerouteing device equipped vehicle doesn't enter the Traffic Network, it will not receive the updated travel times by our system. Similarly to the HTT case, vehicles find the optimum route at every predefined interval (the same value as the HTT case), and the penetration rates of vehicles with rerouteing device are the same of the HTT case: 10%, 20%, 30%, 50%, 70%, 85%, 100%. Moreover, for each penetration rate on the LLR and HTT case, the same vehicles are equipped with rerouteing device, to ensure same environment conditions.

6.3.3 Hourly Travel Times

We used travel times for every hour from the beginning until the end simulation. The collection of these travel times was done by first running one simulation for each Penetration Rate (PR) of Cooperative Automated Vehicles (CAVs) routing themselves every 300 seconds with the current edge travel times. This simulation outputs the hourly travel times for all edges in the network. Based on this travel times a new simulation was conducted, where the PR of CAVs used this travel times of the previous simulation. This second simulation also output the hourly travel times, which is the values we use for the HTT and LLR scenarios.

6.3.4 Static Data for Proposed Scenarios

As pointed out in Section 5.1.3, our Local Level Routing (LLR) requires some data for each TC. Figures 6.4, 6.5, 6.6, and 6.7 show the division of the chosen Traffic Network into 4 sub-networks, each controlled by a Traffic Controller (TC).

For the necessary data for each TC, some of them are directly retrieve from SUMO network files (e.g. lane length, lane connections, connections to yield, etc.), while for others we ran a simulation for each base case scenario using SUMO's instantaneous induction loop detectors on both the begin and end of each lane of the edges controlled by the TC. Among the outputs of this induction loop, we use the id, speed and time a vehicle leaves the detector, and new vehicles are those that are detected on the end detector but not on the begin one, while finished vehicles are the opposite. The following list of values are either standard values or collected through the simulation time but after the warm-up time (1800 s).

- Std. planning horizon time = 180 s (maximum traffic signal plan cycle time of the chosen area).

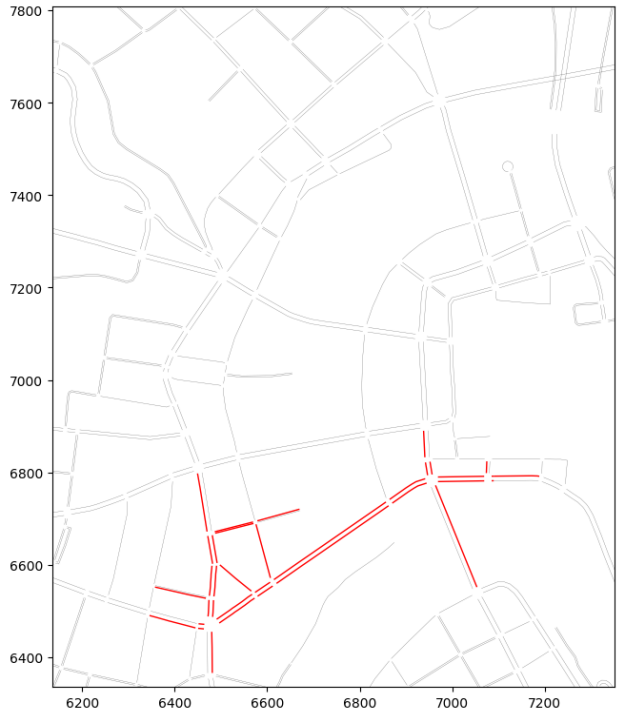


Figure 6.4: Incoming edges controlled by the traffic controller 1.

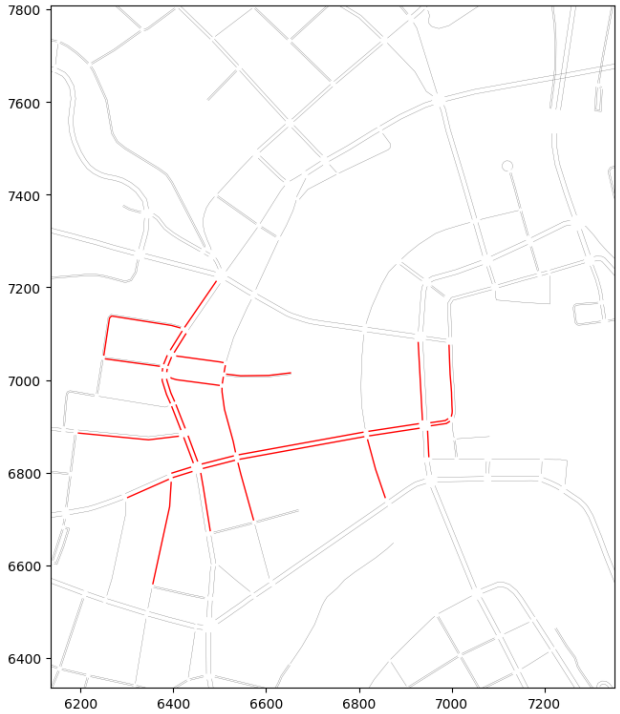


Figure 6.5: Incoming edges controlled by the traffic controller 2.

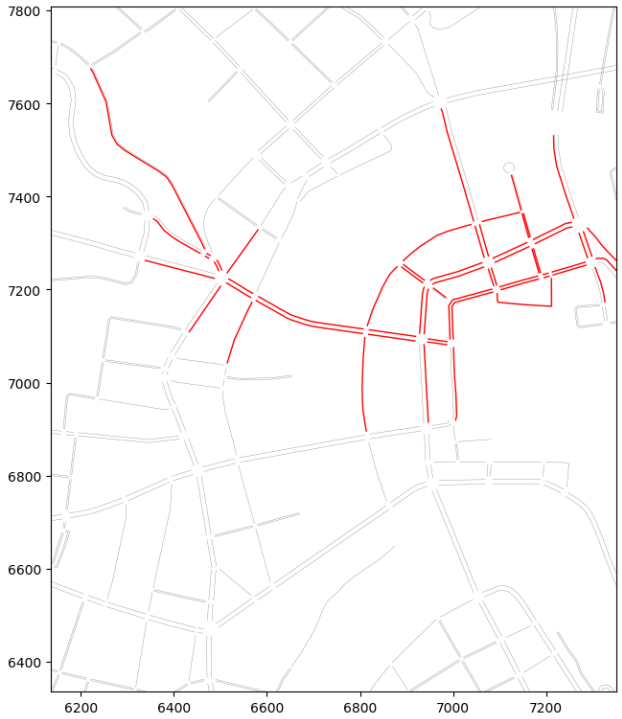


Figure 6.6: Incoming edges controlled by the traffic controller 3.

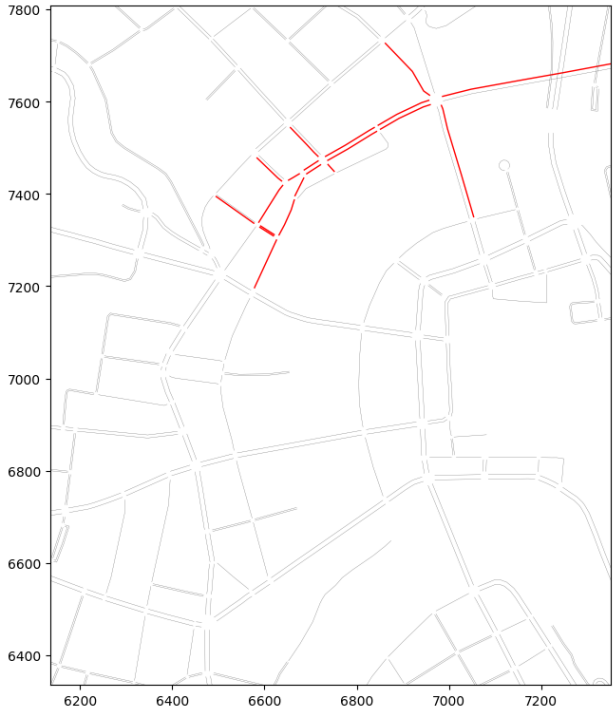


Figure 6.7: Incoming edges controlled by the traffic controller 4.

- Std. communication range = 250 m (assumed as it depends on many factors).
- Std. acceleration adjustment = 0.85 (chosen by fitting best values after many simulations).
- Std. deceleration adjustment = 1.2 (chosen by fitting best values after many simulations).
- Std. connection capacity = 0.5 veh/s = 1800 veh/h, from [Bureau of Public Roads, 1950].
- Std. lane gap time = 1 s, from [Bureau of Public Roads, 1950].
- Std. average critical gap = 5.7 s, from [Fitzpatrick, 1991].
- Std. follow-up time = 60%, from [Fitzpatrick, 1991].
- Last edge probability: ratio between finished vehicles and all detected on the begin detectors of the edge.
- Next edge probability: ratio between all detected on begin detector of each next edge and all detected on the end detectors of the edge.
- Num. vehicles beginning their trip: average number of new vehicles throughout the simulation, but converted into one range (30 s).
- Fixed inflow profile: average values from each vehice detected throughout the simulation, but converted into one range.

For the system-wide data, the following list.

- 5 types of passenger cars, from SUMO's standard vehicles.
- Routing interval = 180 s (same value of planning horizon).
- Routing pre-period (routing time before insert the simulation) = 60 s (SUMO standard value).
- Length of the ranges (division of the planning horizon) = 30 s (chosen by fitting best values after many simulations).
- Low flow threshold = 0.25 veh/s = 1 veh/4 s = 900 veh/h, from [Bureau of Public Roads, 1950].
- Mid flow threshold = 0.5 veh/s = 1 veh/2 s = 1800 veh/h, from [Bureau of Public Roads, 1950].
- Minimum headway = max flow threshold = 1 veh/s, from [Bureau of Public Roads, 1950].
- Reaction time = 1 s, from [Bureau of Public Roads, 1950].

- Minimum stop line gap (min. gap for first vehicles in the queue) = 1 m (SUMO standard value).
- CAVs route sharing probability = 100% (assumed).

6.4 Simulated Edge Cost Estimation Algorithm

The development of the algorithm was done in Python programming language and used TraCI (the interface between SUMO and Python). SUMO has an option called Context Subscriptions in which TraCI will retrieve a predefined information when an object, in our case a CAV, is within a certain range of a reference object (the junction) with the vehicular information retrieval. The following TraCI domains were used.

- Trafficlight: for getting the signal plans within the planning horizon time.
- Vehicle: for retrieval of information from CAVs and updating the travel time of controlled edges within the Traffic Network for each CAV detected by one of the RSUs.

For the HTT, DUA and DUE cases the simulation runs only SUMO and no algorithm is needed. However, an algorithm for the Local Level Routing (LLR) system is used and adapted to the simulation, as presented in Figure 6.8. The algorithm runs twice per time step of the simulation, Δt, representing each iteration, *tcit*, which is done for each traffic controller, TC, in a ordered list of traffic controllers *tcorder* (with last TC is denoted *lasttc*) and junctions to estimate arrivals and departures, *jctorder*. Until the end time of simulation, *endt*, for every time step, t, which corresponds to a real world time of the time step, *actualt*, each TC checks if there is a message received from another TC. If the time to update reaches zero, then all the estimations must be done, what occurs every *lenrge* seconds, i.e. the algorithm updates. The *tcorder* and *jctorder* define the order of junctions to have arrivals and departures to be estimated. Only at the first iteration, the algorithm generates new *Vehicles Already on Lane* (or restart the phase start values - *startt*, *startv* and *startd* - with the next alg. update values - *nuptd* and *nuptv* - for the ones from previous time steps) and *Vehicles Starting on Lane*. Then, and for both iterations, we need to define the lanes, *lanesorder*, following the order of the junctions order, generate *Vehicles Arriving on Lane* and estimate the departures of all vehicles per junction, as well as prepare the outflow profiles to the next junctions controlled by other TCs. Especially for the second iteration, we define the probe vehicles in order to be able to predict travel times (TTs) and queue lengths with the new arrivals from both lane orders (1^{st} and 2^{nd} iterations). The algorithm also remove vehicles on lanes that will be in the *lanesorder* of the next iteration as new arrivals will be generated using updated information.

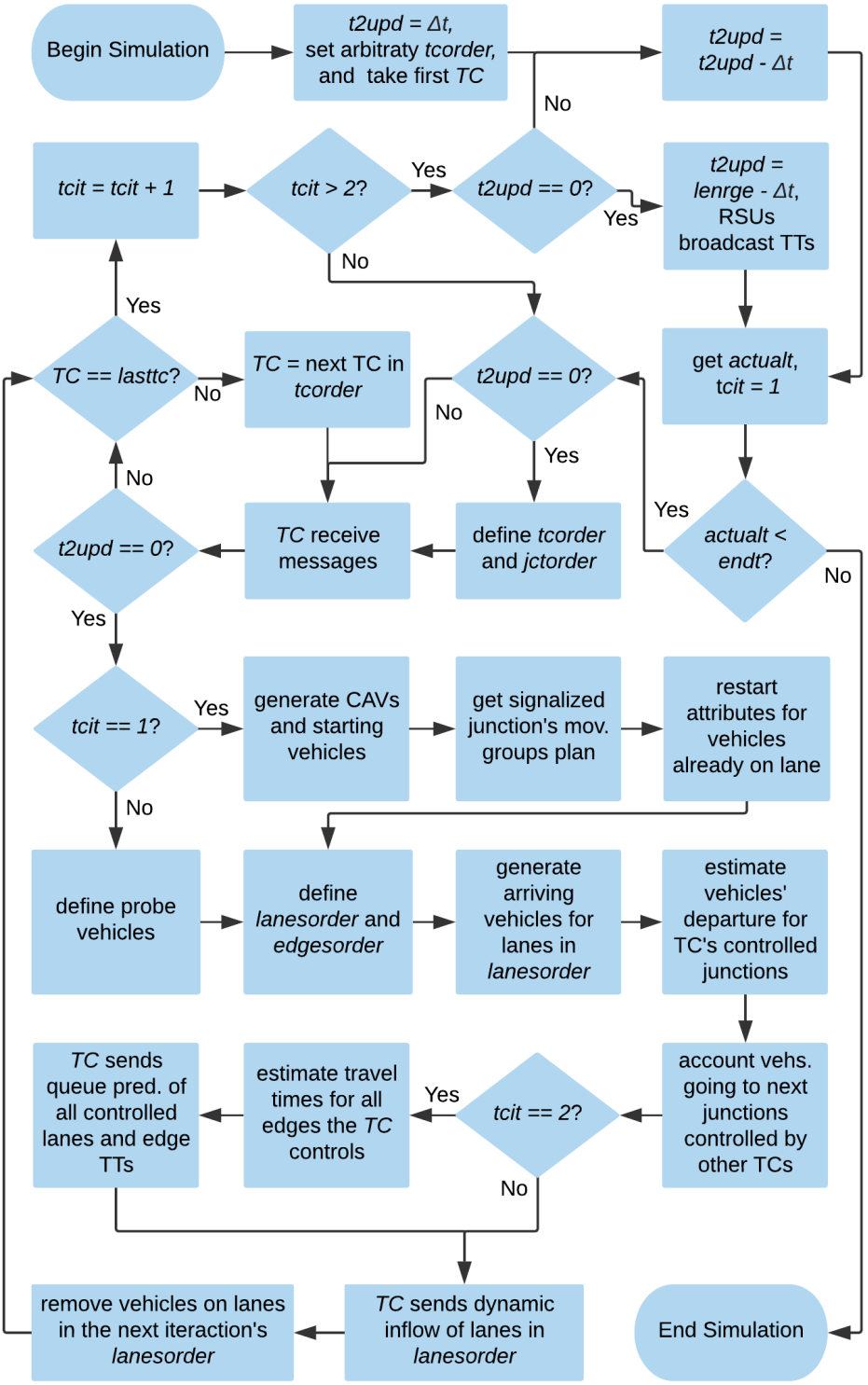


Figure 6.8: Simulated edge cost estimation algorithm.

6.5 Simulated Optimal Route Algorithm

SUMO offers a range of routing algorithms, they were described already in Section 3 and Appendix A.3:

- *Dijkstra*, the simplest and well suited to routing in time-dependent networks, but slowest one;
- *A**, similar to Dijkstra, though it uses the euclidean distance divided by the maximum speed for bounding the travel time to direct the search and speed up the process;
- *CH*, uses preprocessing and it is very efficient when a large number of queries is expected, it also considers time-dependent weights; and
- *CHWrapper*, works like CH but to be more efficient it separates preprocessing for every vehicle class that is encountered.

We decided to use the Dijkstra routing algorithm (see Algorithm 1) from SUMO as it leads to the exact solution and the best alternative for our discrete time-dependent routing.

6.6 Evaluation Method

Each scenario was analysed based on the traffic performance measures for all vehicles routed by the Local Level Routing (LLR) system for the LLR and Hourly Travel Times (HTT) cases, and the edges controlled by the traffic controllers (see Figures 6.4, 6.5, 6.6 and 6.7) for all cases. Additionally, we also introduce 5 "probe" vehicles (each with an Origin-Destination pair) inside the Traffic Network (TN) of the chosen LLR area that are generated every 15 minutes, in order to evaluate the influence of the LLR algorithm against the HTT individually for each vehicle. The origin and destination of each vehicle is seen in Figure 6.9.

Since the comparison deals with simulation, which is stochastic, we needed to develop an experiment that could create different initial conditions at each replication. Figure 6.10 shows the steps of the replication experiment in which each step was carried by each replication and the one in the last replication.

For the number of required replications of the simulation to ensure good confidence of the results, we used the information and formula from [Burghout, 2004],

$$N(m) = \left(\frac{S(m) \cdot t_{m-1,1-\alpha/2}}{X(m) \cdot \epsilon} \right)^2 . \tag{6.2}$$

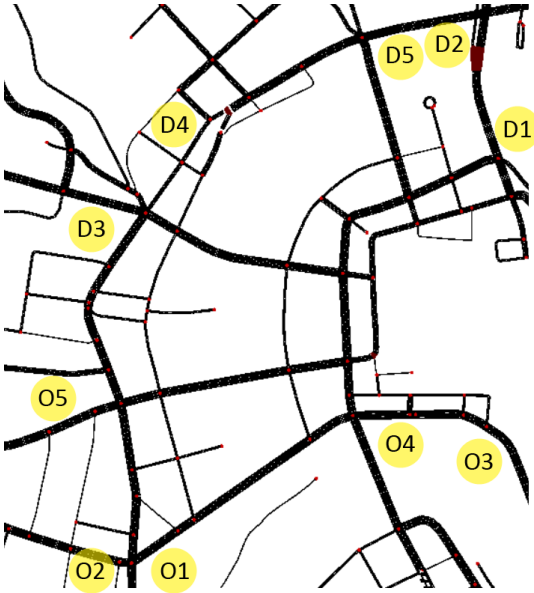


Figure 6.9: Origin (O) and destination (D) of each vehicle used to analyse individual results.

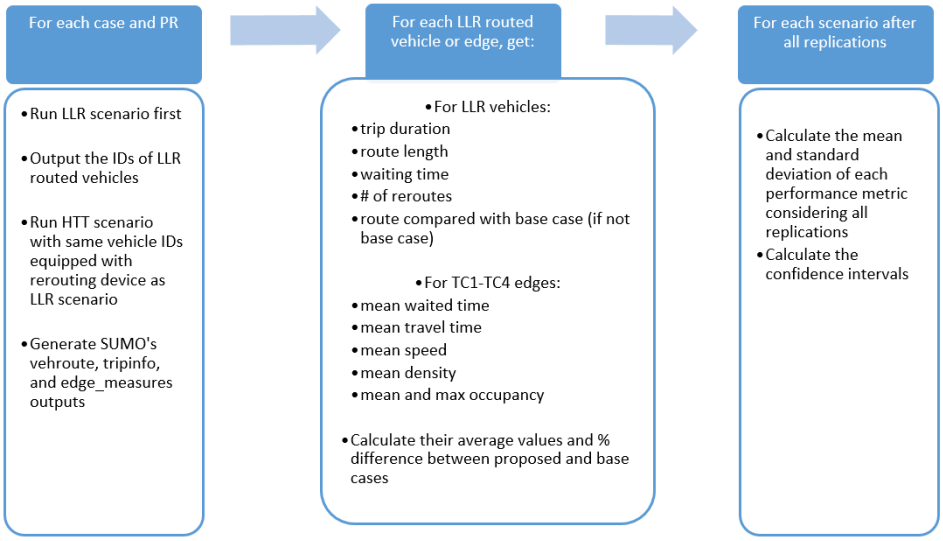


Figure 6.10: Evaluation steps.

Equation 6.2 states that the number of required replications, $N(m)$, is based on m initial replications. The other inputs include, $X(m)$, which represents the real mean from the m replications, $S(m)$, which represents the real standard deviation from the m replications. Moreover, ϵ represents the allowable percentage of error, while $t_{m-1,1-\alpha/2}$ the critical value of the two-tailed Student's t-distribution at α level of significance and $m-1$ degrees of freedom. Applying Equation 6.2 for all our measures and accepting 6% allowable error, we would need to make 12 replications to have accuracy of (at least and approximately) 95% if we consider only aggregated values. By

aggregated values we mean the measures from all LLR routed vehicles and edges, except the 5 additional vehicles for the individual evaluations. This is because if considering them the number increases to 273 replications. It would be desired to make all these replications but this would requires some weeks to get all results and, due time constraints, not possible. Based on the results of each replication, and using the *Student's t-distribution*, we can calculate the confidence interval as it follows:

$$X(n) \pm t_{n-1,1-\alpha/2} \cdot \sqrt{\frac{S(n)^2}{n}}. \tag{6.3}$$

Equation 6.3 presents the mean estimate, $X(n)$, based on n initial replications and the half of confidence interval consisting of $t_{m-1,1-\alpha/2}$. The critical value of the two-tailed Student's is the t-distribution at α level of significance and $n-1$ degrees of freedom and $S(n)^2$ which represents the standard deviation from n replications.

6.7 Results

Although we will discuss the results throughout this section, the confidence intervals, calculated using Equation 6.3, of the evaluated metrics are seen in Appendix C.1, C.2, C.3, C.4, C.5, C.6, C.7, C.8, C.9, C.10, and C.11. The first analysis we do is related to the number of vehicles that reach the area controlled by the Local Level Routing (LLR) system, in order to check if it remains the same according to the scenario. Table 6.1 shows that around 19% of all Cooperative Automated Vehicles (CAVs) enters the area of the Traffic Network in all scenarios.

According to Table C.1 the percentage of LLR routed vehicles that have a different route from the base scenario with same Penetration Rate (PR) with Hourly Travel Times (HTT) is high in all scenarios, in average 93%. Surprisingly, the impact of the LLR system on the percentage of different edges is higher when aggregated all CAVs than only the probes. However, the proportion of different edges increases much more for the probes than all CAVs at higher PRs, as Figure 6.11 illustrates. This means that as the probes' route within the chosen LLR Traffic Network (TN) is quite short, they are changing only few edges and in average only once (for any PR scenario), as seen in Table C.10, what is expected as the rerouteing interval is 180 seconds and the mean route duration is about 250 seconds. Tables C.5 and C.7 shows little difference of route duration and length (in average), but the confidence intervals are very broad for the probes, specially probe 3, which is the vehicle that almost all of its DUA route uses the most congested edges (see Figures 6.3 and 6.9). The differences between LLR and HTT scenarios remains quite the same (Tables C.5, C.7 and C.9), and although some scenarios have bigger differences we cannot see a trend. The only pattern of improvement is in the averages, seen in Figures 6.12, 6.13, and 6.14, which is interesting specially

Scenario	Num. LLR Routed Vehs. [#]	Num. All CAVs [#]	Proportion
LLR 10PR	323	1685	19%
HTT 10PR	323	1674	19%
LLR 20PR	623	3463	18%
HTT 20PR	621	3443	18%
LLR 30PR	908	5115	18%
HTT 30PR	905	5085	18%
LLR 50PR	1553	8550	18%
HTT 50PR	1550	8499	18%
LLR 70PR	2201	11983	18%
HTT 70PR	2202	11917	18%
LLR 85PR	2739	14476	19%
HTT 85PR	2736	14387	19%
LLR 100PR	3322	17078	19%
HTT 100PR	3319	16978	20%

Table 6.1: Total number of Cooperative Automated Vehicles (CAVs) and their proportion as well as number that received information from the Local Level Routing (LLR) system.

for waiting time, as the higher the PR these vehicles get less benefits (except the route length that remains quite the same). The explanation for this is that because at lower PRs CAVs have privileged information about traffic conditions and signal timings, that most of others do not, what enables them to enjoy less traffic and get green waves easier. Regarding the instability of the results for probes, we need to make a deeper analysis on why these results are variating so much, even though the average values are accordingly. As pointed out in [Krishnamoorthy, 2008], when rerouteing reacts too often and for every small variation in travel times the results may actually get worse.

Analysing the Table C.2 we see a curious behaviour, the sum of the waited time on the edges (time that vehicles were considered stopped) had a good improved but that didn't come with lower mean travel times and higher mean speeds. However, if we check the variation of the mean travel times and mean speeds from the worst case scenario (DUA) and the best one (DUE), the difference is only 5% for travel times and -3% for speeds, while the waited time in the DUA case is about 215% higher than the DUE case. What lead us to the conclusion that, at least for this scenario and traffic demands, the travel time will hardly improve much, though our LLR algorithm, for mean travel times, performed even better than the DUE case with penetration rates of 50% and 70%, shown in Figure 6.15. We can also see a slight poorer performance of our LLR system above 70% regarding travel times. In fact, this behaviour is quite common in the literature (for instance [Leistner et al., 2012], [Pan et al., 2013], [Codeca et al., 2017], [Olia et al., 2016], [Herbert and Mili, 2008]), and it is due the fact that when receiving travel times vehicles routing at the same time may go to a certain edge which would

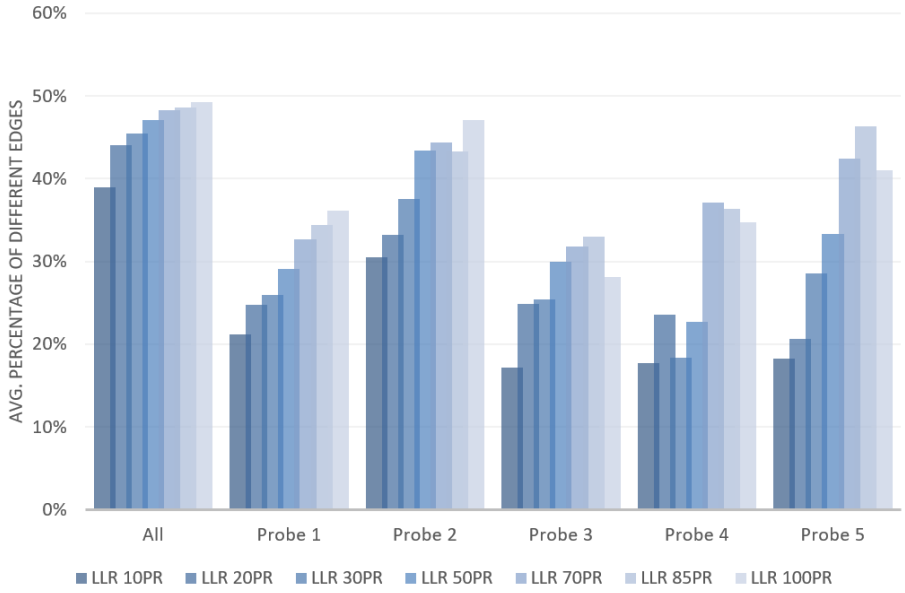


Figure 6.11: Average percentage of different edges.

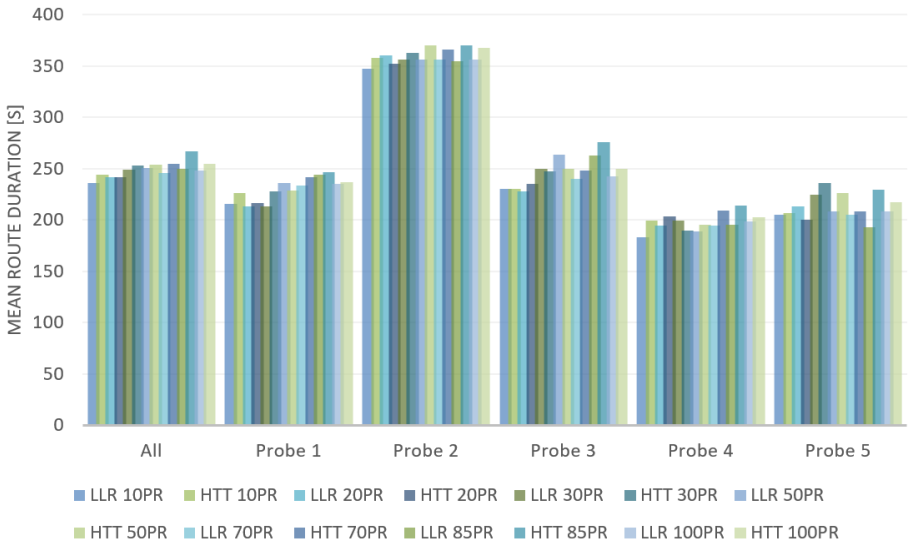


Figure 6.12: Mean route duration, in seconds.

not be congested at the time they would pass, but as the travel times are estimated per time ranges and not per each vehicle (what would demand too much computation and communication between TCs) the last travel time estimation don't consider these vehicles rerouteing to the edge using this last estimation, what might create a punctual congestion. Nevertheless, on the next estimation (next algorithm update) they would be considered and the congestion possibly predicted for that time, but as vehicles have rerouteing interval they will not reroute again, only vehicles to be routed at and after this second algorithm update would have such information when rerouteing. We

Figure 6.13: Mean route length, in meters.

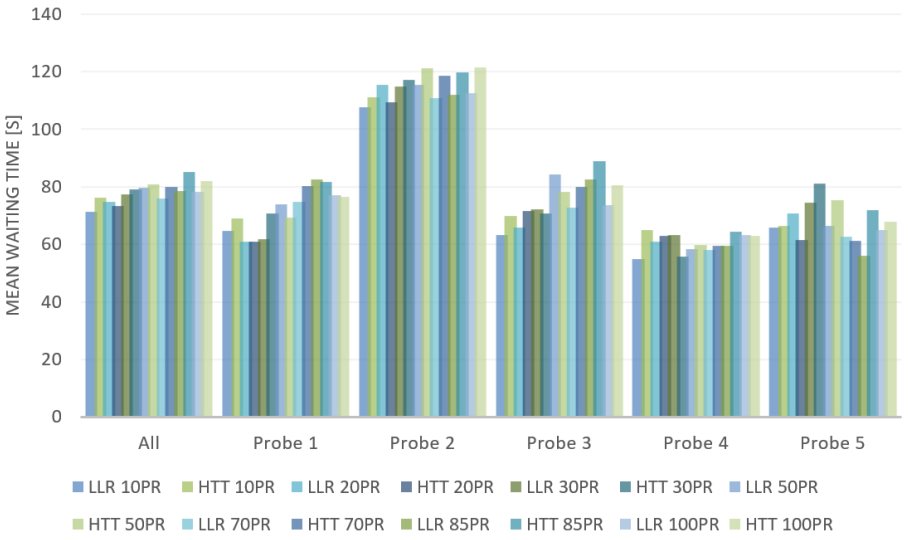


Figure 6.14: Mean waiting time, in seconds.

should notice that, if CAVs share their routes, this rerouteing interval avoids the problem of vehicles changing their routes so much that the predicted traffic become invalid and the routing information is not consistent [Herbert and Mili, 2008]. On the other hand, not only the waited time had a big room for improvement (which we achieved a great part of it), but also the density and occupancy, as seen in Table C.3 and Figures 6.16, 6.17 and 6.18. The mean occupancy gets considerable improvement but it was already low. The expressive reduction of waiting time and maximum mean occupancy (but not together with the travel time) as the PR increases, tell us vehicles may be driving longer or using alternative edges with similar distances to get green wave and avoid longer queues, what balances the load of vehicles through the

traffic network and turns it more efficient. This is the big advantage of our LLR system compared to the HTT, especially on scenarios with PR above 70%. The explanation for this is that when all vehicles use the HTT fastest travel times, they move traffic jams from one place to another, while in the LLR the travel times accounts vehicles routes and balance the load.

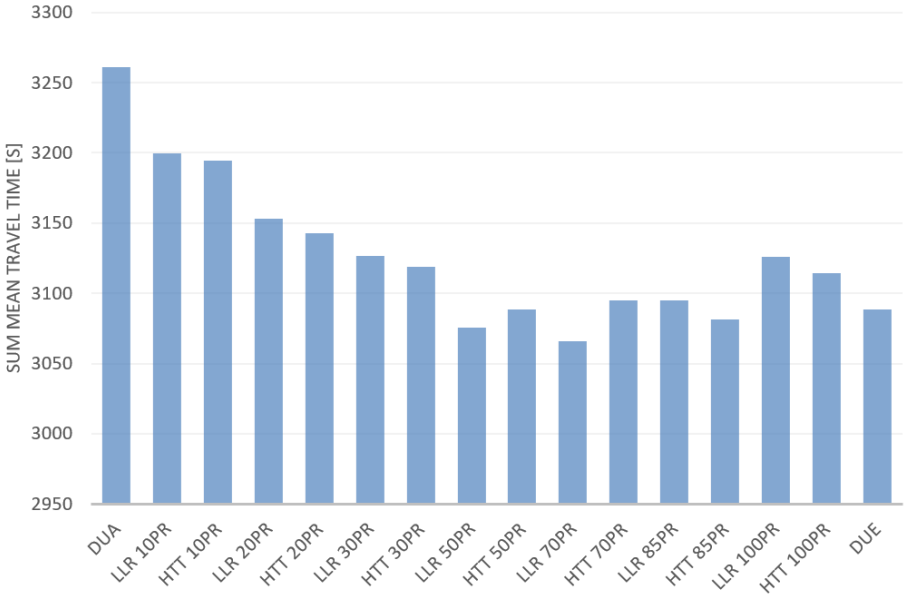


Figure 6.15: Sum of mean travel time on edges controlled by the LLR system, in seconds.

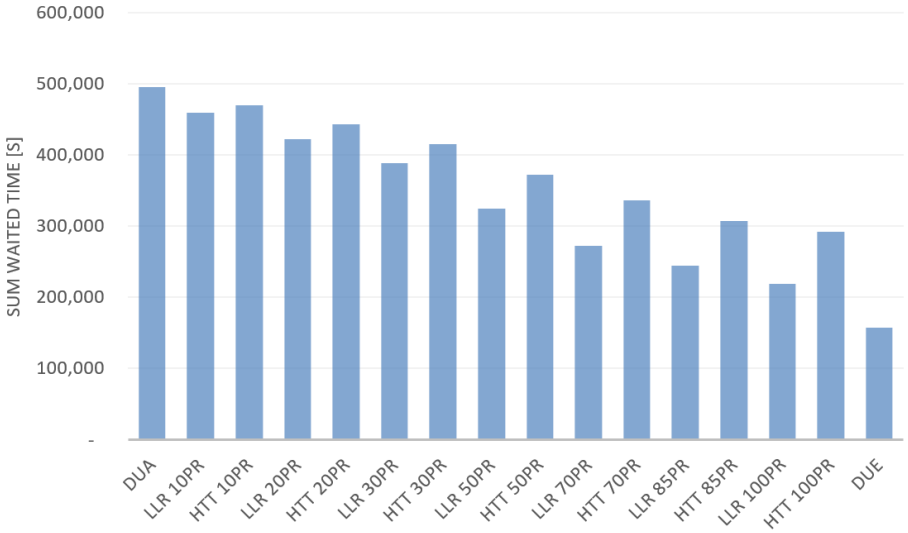


Figure 6.16: Sum of waited time on edges controlled by the LLR system, in seconds.

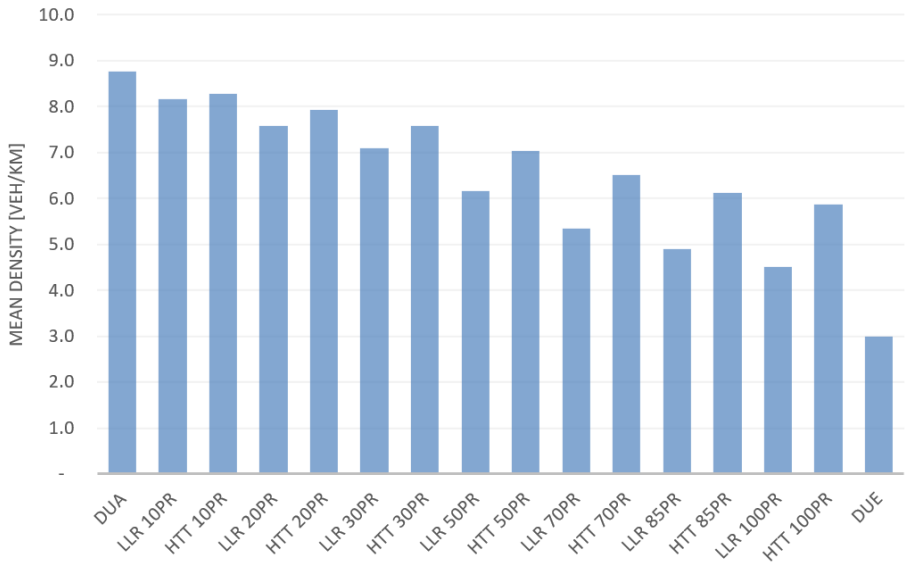


Figure 6.17: Mean density on edges controlled by the LLR system, in vehicles/km.

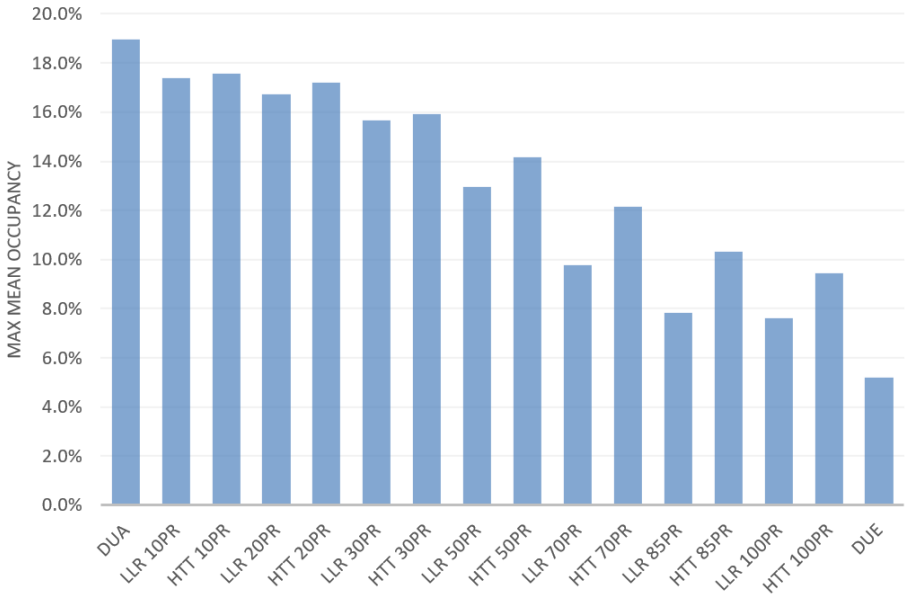


Figure 6.18: Maximum mean occupancy on edges controlled by the LLR system, in percentage.

Chapter 7

Conclusions

In this work we investigated the benefits of local level routing to the Cooperative Intelligent Transport Systems (C-ITS) environment, focusing on the opportunities related to the reduction of travel times and congestion as well as better balancing the load of vehicles throughout a traffic network. Our contributions were supported by the European Union and will be evaluated as part of the project MAVEN (Managing Automated Vehicles Enhances Network), funded by the EC Horizon 2020 Research and Innovation Framework Programme. In the following list we summarise the conclusions of this thesis according to the research objectives defined in Section 1.1:

1. **Optimal route algorithms and edge cost estimation models:** in Chapter 3 we researched algorithms for the optimal route problem (in particular shortest-path problems) that have short running time and can handle the dynamic behaviour of travel times in urban networks, especially due signal timings and variety of route alternatives. This lead to the choice of discrete FIFO (First-In-First-Out) time-dependent "shortest" (in fact minimum-time or earliest arrival) path problem. After that, in Chapter 4 we explored models for edge travel time estimation that were accurate, able to input the available traffic information, and output in the format needed for the chosen shortest-path algorithm. As we require only the arrival and departures times at edges with certain degree of prediction and quick running time, microscopic models present unnecessary complexity for only one metric (i.e. travel time), high resource usage at larger networks and limited benefits if the implementation is fully distributed (a traffic controller for almost one junction); while macroscopic models lack of enough detail; and queueing models use flows instead of known arrivals. Our main contribution is the proposed event-based traffic model to predict the movement of vehicles along an edge coupled with a deterministic queueing model for the prediction of travel times.

2. **Local level routing system:** after defining the chosen models and algorithm type, in Chapter 5 we developed a route guidance system in which

Cooperative Automated Vehicles (CAV) reroute themselves through a network using discrete time-dependent edge travel times (within a prediction time) based on accurate short-term traffic predictions considering signal timings and vehicle arrivals of each discrete intervals, as well as queue lengths and unexpected events. On this way, vehicles with a destination outside the area controlled by the system can find the optimal route considering the nearby accurate predictions and faraway stored information, avoiding route advices to an unnecessary local destination. However, one important issue is that benefits are realised when there is a time interval between reroutes, i.e. recommended from 180 up to 300 s [Codeca et al., 2017], and vehicles should reroute at different timings. The reason is due to unreliable predictions caused by too short intervals that worsen the performance ([Watling and van Vuren, 1993], [Herbert and Mili, 2008]) and many vehicles rerouteing at the same time (specially above 70% penetration rate). We implement the system in the infrastructure (i.e traffic network) composed by connected Traffic Controllers (TCs) that exchange 1) edge travel times, 2) queue prediction, and 3) inflow profile messages between each other to allow the system to be flexible as a distributed or centralized according to the required running time and communication restrictions. TCs model the movement of vehicles per green phase by estimating arrival and departure times at each edge using the proposed event-based traffic model, which identifies 5 events throughout the movement assuming that vehicles will accelerate until certain distance they need to start braking and then accelerate again when possible. TCs also define vehicles trip through the network flowing between junctions using turning rates, and at lane level using the predicted queue from the traffic model. This enables modelling microscopically at edges and statistically between edges, maintaining short running time while using key data for accurate predictions. The travel times are specific for each time window/range (a fixed length division of the prediction time), and estimated using either: the difference between departure and arrival time for vehicles that arrived within the time window; or a deterministic queuing model based on the cumulative arrivals and departures up to the end of the time window, including the delay time given departures on the preferential traffic to yield. Such estimation considers all available information, as well as the waiting due yielding to preferential traffic, even when there is no vehicle on the lane.

3. **Evaluation in a real-world scenario:** finally, in Chapter 6 we analysed the performance of our proposed local level routing system and compared it against the best (Dynamic User Equilibrium - DUE), worst (Dynamic User Assignment - DUA) and Hourly Travel Times (HTT) cases. At the Traffic Network performance level and considering the Local Level Routing (LLR) system proposed case against the HTT case, the biggest improvement was at reducing the waited time (time a vehicle is considered stopped on the edge) up to 25%, while 23% lower density and occupancy (mean and maximum mean) improvement of roughly

20%. If we compare our proposed system against the scenario without routing (Dynamic User Assignment), we have reduction up to 56% of waited time, 48% for density and roughly 60% lower maximum mean occupancy at 100 penetration rate. This let us conclude that CAVs were avoiding traffic jams they would experience if using only hourly travel times and/or getting green waves, doing so by taking a longer route or alternative path with similar length which at the end didn't bring much improvement for network travel time. Indeed, the mean speed and sum of mean travel time show almost no improvements between LLR and HTT cases, but still up to 6% lower travel time compared to the DUA case. However, the difference between worst and best cases is only about 5% and we achieve a reduction of travel time better than the best case. Additionally, CAVs have slight lower travel times of 5% and considerably less waiting times, 9%, in 10% penetration rate compared to 100%, even their route length is pretty much the same in all scenarios. This is due the fact that they get privileged information at low penetration rates (PR) and as the PR increases the benefits don't improve at the same rate for them, though at network level it turns more expressive.

7.1 Future Work

Throughout this work we identified possible directions that could complement our study, we list then in the following list.

- The proposed messages, in particular the inflow profile, to be exchanged between traffic controllers contain information that can be useful for other systems like signal control and network coordination (green wave) and, if included additional information (i.e. number of priority or emergency vehicle per range/time window of the planning horizon), it could be used by priority management and emergency situations. Therefore, the format of the messages as well as their interval of exchange could be adapted in order to reduce the communication burden and simplify the integration of all other systems.

- Regarding the travel time broadcasting, in case of an unexpected event, it would be recommended if Roadside Units also inform that vehicles should reroute again even if it is not the time of another reroute given the rerouteing interval. For instance, if an edge is fully closed due an accident a vehicle would still go there even if it is known the edge is closed. A strategy should be developed for this situation, due the fact that if all vehicles route at the same time they may create congestion on edges predicted to be not congested.

- As traffic prediction is affected by vehicle's reactions to the routing information, route choice models (discussed in Appendix B.3) would be

worth to be explored and replace the actual static turning rates that doesn't consider perceived travel times.

- The predicted travel times are averaged per edge, even though we have the precision of travel times per movement groups. This can be a big issue in busy roads where there are non-protected left-turning bay and left-turning vehicles wait long time while the ones going through have almost no delay. On this way, the left-turning and through vehicles get same edge travel times but in reality they would experience very different delays. The ideal situation would be routing and providing travel times per connections, but the feasibility of this is a problem.
- A model for lane change is recommended (a rule-based should be enough), the main usage is when a vehicle in front doesn't have green signal and there are other lanes available, or when a slow vehicle in front delays the vehicle behind and even in the case of either different lane speeds or lane closure.
- In the current setting of the system, vehicles already on lane for the next algorithm update are estimated based on the modelling of last update. However, these vehicle could use vehicle detectors between the algorithm updates and generate vehicles at the position and time they were detected on the next update. Additionally, this time interval between algorithm updates uses the same time of the time window/range's length (division of the planning horizon) but that could be changed to allow more frequent update while maintain efficient time window intervals, though this would required a quite major modification of the system.
- When there is high traffic (around the high threshold), the travel times are easier to predict due to little variation of headways, and both travel times as well as modelling of vehicles could use saturation flows and only the queueing model estimating departures.
- We also would like to make a deeper analysis on why the results for probes were more unstable, and test in more scenarios with higher flows and in cases with lane closures as well as study the performance of the system relate to vehicle's emissions.

Bibliography

[Akcelik, 1980] Akcelik, R. (1980). Time-Dependent Expressions for Delay , Stop Rate and Queue Length at Traffic Signals. pages 1–19.

[Akcelik, 1988] Akcelik, R. (1988). Highway Capacity Manual Delay Formula for Signalized Intersections. *ITE Journal (Institute of Transportation Engineers)*, 58(3):23–27.

[Akçelik and Besley, 2002] Akçelik, R. and Besley, M. (2002). Queue Discharge Flow and Speed Models for S ignalised I ntersections. *Queue*, pages 99–118.

[Anderson and Bell, 1997] Anderson, J. and Bell, M. (1997). Travel time estimation in urban road networks. *Intelligent Transportation System, 1997. ITSC'97., IEEE Conference on. IEEE.*

[Batz et al., 2010] Batz, G. V., Geisberger, R., Neubauer, S., and Sanders, P. (2010). Time-dependent contraction hierarchies and approximation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6049 LNCS:166–177.

[Batz and Sanders, 2012] Batz, G. V. and Sanders, P. (2012). Time-dependent route planning with generalized objective functions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7501 LNCS:169–180.

[Behrisch et al., 2011] Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility - an Overview. *Proceedings of the 3rd International Conference on Advances in System Simulation (SIMUL'11)*, (c):63–68.

[Bellman, 1955] Bellman, R. (1955). On a routing problem.

[Ben-Akiva and Bierlaire, 1999] Ben-Akiva, M. and Bierlaire, M. (1999). Discrete Choice Methods and Their Applications to Short-Term Travel Decisions. In *Handbook of Transportation Science*, number Draft, pages 7–37.

[Benekohal and El-Zohairy, 2001] Benekohal, R. F. and El-Zohairy, Y. M. (2001). Multi-regime arrival rate uniform delay models for signalized intersections. *Transportation Research Part A: Policy and Practice*, 35(7):625–667.

[Bureau of Public Roads, 1950] Bureau of Public Roads (1950). *Highway Capacity Manual 2000*.

[Burghout, 2004] Burghout, W. (2004). A note on the number of replication runs in stochastic traffic simulation models. (September):1–6.

[Carey, 2001] Carey, M. (2001). Dynamic Traffic Assignment with More Flexible Modelling within Links. In *Networks and Spatial Economics*, number 1955, pages 349–375.

[Cascetta et al., 1996] Cascetta, E., Nuzzolo, A., Russo, F., and Vitetta, A. (1996). A modified logit route choice model overcoming path overlapping problems. Specification and some calibration results for interurban networks. In *TRANSPORTATION AND TRAFFIC THEORY. PROCEEDINGS OF THE 13TH INTERNATIONAL SYMPOSIUM ON TRANSPORTATION AND TRAFFIC THEORY, LYON, FRANCE, 24-26 JULY 1996*, Lyon.

[Catling, 1977] Catling, I. (1977). A time-dependent approach to junction delays. *Traffic Engineering & Control*, 18(Analytic).

[Chabini and Lan, 2002] Chabini, I. and Lan, S. (2002). Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):60–74.

[Chai et al., 2017] Chai, H., Zhang, H. M., Ghosal, D., and Chuah, C. N. (2017). Dynamic traffic routing in a network with adaptive signal control. *Transportation Research Part C: Emerging Technologies*, 85:64–85.

[Chen et al., 2014] Chen, H., Hancock, K. L., and Katz, B. J. (2014). Real-Time Traffic State Prediction : Modeling and Applications.

[Chen and Hu, 2012] Chen, L. W. and Hu, T. Y. (2012). Flow equilibrium under dynamic traffic assignment and signal control—an illustration of pretimed and actuated signal control policies. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1266–1276.

[Chen et al., 2006] Chen, X., Yang, Z. S., and Wang, H. Y. (2006). A multi-agent urban traffic control system cooperated with dynamic route guidance. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, 2006(August):330–335.

[Chen and Tang, 1998] Chen, Y. L. and Tang, K. (1998). Minimum time paths in a network with mixed time constraints. *Computers and Operations Research*, 25(10):793–805.

[Chrobok et al., 2000] Chrobok, R., Kaumann, O., Wahle, J., and Schreckenberg, M. (2000). Three categories of traffic data: Historical, current, and predictive. *Proceedings of the 9th IFAC Symposium Control in Transportation Systems*, pages 250–255.

[Chun-Hsin Wu et al., 2003] Chun-Hsin Wu, Chia-Chen Wei, Da-Chun Su, Ming-Hua Chang, Jan-Ming Ho, Wu, C.-h., Wei, C.-c., Su, D.-c., Chang, M.-h., and Ho, J.-m. (2003). Travel time prediction with support vector regression. *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, 2:1438–1442.

[Codeca et al., 2016] Codeca, L., Frank, R., and Engel, T. (2016). Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. *IEEE Vehicular Networking Conference, VNC*, 2016-Janua:1–8.

[Codeca et al., 2017] Codeca, L., Frank, R., Faye, S., and Engel, T. (2017). Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63.

[Cooke and Halsey, 1966] Cooke, K. L. and Halsey, E. (1966). The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14(3):493–498.

[Daganzo, 1977] Daganzo, C. F. (1977). On stochastic models of traffic assignment.pdf.

[Dean, 1999] Dean, B. C. (1999). Continuous-Time Dynamic Shortest Path Algorithms. *Electrical Engineering*, page 116.

[Dean, 2004] Dean, B. C. (2004). Shortest paths in FIFO time-dependent networks: Theory and algorithms. *Rapport technique, Massachusetts Institute of ...*, pages 1–13.

[Delling, 2011] Delling, D. (2011). Time-dependent SHARC-routing. *Algorithmica (New York)*, 60(1):60–94.

[Delling and Wagner, 2009] Delling, D. and Wagner, D. (2009). Time-dependent route planning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5868 LNCS:207–230.

[Demiryurek et al., 2011] Demiryurek, U., Banaei-Kashani, F., Shahabi, C., and Ranganathan, A. (2011). Online computation of fastest path in time-dependent spatial networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6849 LNCS:92–111.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A Note on Two Probles in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271.

[Dong, 2011] Dong, W. (2011). An overview of in-vehicle route guidance system. *Australasian Transport Research Forum 2011 Proceedings*, (September):1–12.

[Dorigo and Stutzle, 2004] Dorigo, M. and Stutzle, T. (2004). Ant Colony Optimization Theory. *Ant Colony Optimization*, (di):121–152.

[Dreyfus, 1969] Dreyfus, S. E. (1969). An Appraisal of Some Shortest-Path Algorithms. *Operations Research*, 17(3):395–412.

[Eppstein, 1998] Eppstein, D. (1998). Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673.

[Ermagun and David Levinson, 2017] Ermagun, A. and David Levinson (2017). Spatiotemporal Traffic Forecasting: Review and Proposed Directions. In *96thAnnual Transportation Research Board Meeting*, number January 2017.

[European Commission, 2012] European Commission (2012). Road Transport - A change of gear.

[European Commission, 2016] European Commission (2016). A european strategy on cooperative intelligent transport systems, a milestone towards cooperative, connected and automated mobility.

[Faez and Khanjary, 2008] Faez, K. and Khanjary, M. (2008). Utospf: A distributed dynamic route guidance system based on Wireless Sensor Networks and Open Shortest Path First protocol. *ISWCS'08 - Proceedings of the 2008 IEEE International Symposium on Wireless Communication Systems*, pages 558–562.

[Faez and Khanjary, 2009] Faez, K. and Khanjary, M. (2009). UTOSPF with waiting times for green light consideration. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, (October):4170–4174.

[Farzaneh, 2005] Farzaneh, M. (2005). Modeling traffic dispersion. (November).

[Festag, 2014] Festag, A. (2014). Cooperative intelligent transport systems standards in Europe. *IEEE Communications Magazine*, 52(12):166–172.

[FHWA, 2001] FHWA (2001). *Traffic Flow Theory Revised. A State-of-the-Art Report*. 2001 edition.

[FHWA, 2018] FHWA (2018). *Traffic Flow Theory Revised. A State-of-the-Art Report*. 2018 edition.

[Fitzpatrick, 1991] Fitzpatrick, K. (1991). Gaps Accepted at Stop-Controlled Intersections. *Transportation Research Record*, (1303):103–112.

[Fu, 2001] Fu, L. (2001). An adaptive routing algorithm for in-vehicle route guidance systems with real-time information. *Transportation Research Part B*, 35:749–765.

[Fu et al., 2006] Fu, L., Sun, D., and Rilett, L. R. (2006). Heuristic shortest path algorithms for transportation applications: State of the art. *Computers and Operations Research*, 33(11):3324–3343.

[Gazis, 1974] Gazis, D. C. (1974). Traffic science. *A Wiley-Intersection Publication*, pages 148–151.

[Geroliminis and Skabardonis, 2005] Geroliminis, N. and Skabardonis, A. (2005). Prediction of Arrival Profiles and Queue Lengths Along Signalized Arterials by Using a Markov Decision Process. *Transportation Research Record*, 1934(1):116–124.

[Goldberg and Harrelson, 2005] Goldberg, A. V. and Harrelson, C. (2005). Computing the Shortest Path : A * Search Meets Graph Theory. Technical Report March 2003.

[Gupta et al., 2003] Gupta, M. M., Homma, N., Jin, L., and Homma, N. (2003). *Static and Dynamic Neural Networks: from fundamentals to advanced theory.*

[Herbert and Mili, 2008] Herbert, W. and Mili, F. (2008). Route guidance: State of the art vs. state of the practice. *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1167–1174.

[Hoogendoorn and Knoop, 2012] Hoogendoorn, S. and Knoop, V. (2012). Traffic flow theory and modelling. In *The Transport System and Transport Policy: An Introduction*, pages 125–159.

[Hoogendoorn and Bovy, 2001] Hoogendoorn, S. P. and Bovy, P. H. L. (2001). State-of-the-art of vehicular traffic flow modelling. *Proceedings of the I MECH E Part I Journal of Systems & Control in Engineer*, 215(4):283–303.

[Johnson, 2001] Johnson, K. (2001). *Physics for You.* Nelson Thornes.

[Kampen, 2015] Kampen, J. V. (2015). Route guidance and signal control based on the back-pressure algorithm.

[Khanjary et al., 2011] Khanjary, M., Faez, K., Meybodi, M. R., and Sabaei, M. (2011). PersianGulf: An autonomous combined traffic signal controller and route guidance system. *IEEE Vehicular Technology Conference*, (61).

[Khanjary and Hashemi, 2012] Khanjary, M. and Hashemi, S. (2012). Route guidance systems: review and classification. *Proceedings of the 6th Euro American on Telematics and Information Systems*, pages 269–275.

[Kimber and Hollis, 1979] Kimber, R. and Hollis, E. (1979). Traffic queues and delays at road junctions. Technical report.

[Krishnamoorthy, 2008] Krishnamoorthy, R. K. (2008). Travel time estimation and forecasting on urban roads. (June):283.

[L. Zhao, 2008] L. Zhao, T. Ohshima, H. N. (2008). A* algorithm for the time-dependent shortest path problem. *The 11th Japan-Korea Joint Workshop on Algorithms and Computation (WAAC08)*, pages 36–43.

[Lei and Ozguner, 1999] Lei, J. and Ozguner, U. (1999). Combined decentralized multi-destination dynamic routing and real-time traffic light control for congested traffic networks. *Proceedings of the 38th IEEE Conference on Decision and Control Cat No99CH36304*, pp(December 1999):3277–3282.

[Leistner et al., 2012] Leistner, D., Vreeswijk, J., and Blokpoel, R. (2012). Micro-routing using accurate traffic predictions. *IET Intelligent Transport Systems*, 6(4):380–387.

[Li et al., 2015] Li, P., Mirchandani, P., and Zhou, X. (2015). Solving simultaneous route guidance and traffic signal optimization problem using space-phase-time hypernetwork. *Transportation Research Part B: Methodological*, 81(P1):103–130.

[Liang and Wakahara, 2014] Liang, Z. and Wakahara, Y. (2014). Real-time urban traffic amount prediction models for dynamic route guidance systems. pages 1–13.

[Liobaite and Khokhlov, 2016] Liobaite, I. and Khokhlov, M. (2016). Optimal estimates for short horizon travel time prediction in urban areas. *Intelligent Data Analysis*, 20(6):1459–1475.

[Ma et al., 2002] Ma, S., He, G., and Wang, S. (2002). A hierarchical coordination model for control- guidance integrated systems in ITS. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2002-Janua(September):522–527.

[Madkour et al., 2017] Madkour, A., Aref, W. G., Rehman, F. U., Rahman, M. A., and Basalamah, S. (2017). A Survey of Shortest-Path Algorithms. pages 1–26.

[Martinez et al., 2011] Martinez, F. J., Toh, C. K., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2011). A survey and comparative study of simulators for vehicular ad hoc networks (VANETs). *Wireless Communications and Mobile Computing*, 11(7):813–828.

[Mathew, 2014] Mathew, T. (2014). Signalized Intersection Delay Models. In *Transportation Systems Engineering*, pages 1–18.

[Mathew, 2017a] Mathew, T. V. (2017a). Lecture Notes in Transportation Systems Engineering - Microscopic Traffic Simulation. *NPTEL Online courses*, pages 1–20.

[Mathew, 2017b] Mathew, T. V. (2017b). Lecture Notes in Transportation Systems Engineering - Traffic Progression Models. *NPTEL Online courses*, pages 1–20.

[May Jr and Keller, 1967] May Jr, A. D. and Keller, H. E. (1967). A deterministic queueing model. *Transportation research*, 1(2):117–128.

[Meneguzzer, 1997] Meneguzzer, C. (1997). Review of Models Combining Traffic Assignment and Signal Control.

[Merchant and Nemhauser, 1978] Merchant, D. K. and Nemhauser, G. L. (1978). A Model and an Algorithm for the Dynamic Traffic Assignment Problems.

[Miller, 1963] Miller, A. J. (1963). Settings for Fixed-Cycle Traffic Signals. *Operational Research Society*, 14(4):373–386.

[Mitchell, 1998] Mitchell (1998). Genetic Algorithms.

[Nannicini et al., 2008] Nannicini, G., Delling, D., Liberti, L., and Schultes, D. (2008). Bidirectional A*search for time-dependent fast paths. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5038 LNCS(2):334–346.

[Nejad et al., 2016] Nejad, M. M., Mashayekhy, L., Chinnam, R. B., and Phillips, A. (2016). Hierarchical time-dependent shortest path algorithms for vehicle routing under ITS. *IIE Transactions (Institute of Industrial Engineers)*, 48(2):158–169.

[Ni, 2016] Ni, D. (2016). Multiscale Traffic Flow Modeling. *Traffic Flow Theory*, pages 361–377.

[Ohno, 1978] Ohno, K. (1978). Computational algorithm for a fixed cycle traffic signal and new approximate expressions for average delay.

[Ojeda et al., 2013] Ojeda, L. L., Kibangou, A. Y., and de Wit, C. C. (2013). Online dynamic travel time prediction using speed and flow measurements. *Control Conference (ECC), 2013 European*, pages 4045–4050.

[Olia et al., 2016] Olia, A., Abdelgawad, H., Abdulhai, B., and Razavi, S. N. (2016). Assessing the Potential Impacts of Connected Vehicles: Mobility, Environmental, and Safety Perspectives. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 20(3):229–243.

[Pacey, 1956] Pacey, G. (1956). The progress of a bunch of vehicles released from a traffic signal. *London: Road Research Laboratory*.

[Pan et al., 2012] Pan, J., Khan, M. A., Popa, I. S., Zeitouni, K., and Borcea, C. (2012). Proactive vehicle re-routing strategies for congestion avoidance. *Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2012*, (1):265–272.

[Pan et al., 2013] Pan, J., Popa, I. S., Zeitouni, K., and Borcea, C. (2013). Proactive vehicular traffic rerouting for lower travel time. *IEEE Transactions on Vehicular Technology*, 62(8):3551–3568.

[Peter E. Hart et al., 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael (1968). Formal Basis for the Heuristic Determination of Minimum Costs Paths. *Systems Science and Cybernetics*, (2):100–107.

[Prato, 2009] Prato, C. G. (2009). Route choice modeling: Past, present and future research directions. *Journal of Choice Modelling*, 2(1):65–100.

[Rahman and Kaiser, 2016] Rahman, S. and Kaiser, M. S. (2016). FNN based Adaptive Route Selection Support System. 7(10).

[Robertson, 1969] Robertson, D. I. (1969). TRANSYT: A Traffic Network Study Tool.

[Rouphail M and Akcelik, 1992] Rouphail M, N. and Akcelik, R. (1992). Oversaturation Delay Estimates With Consideration of Peaking. *Transportation Research Record*, (1365):71–81.

[Schmitt and Jula, 2006] Schmitt, E. and Jula, H. (2006). Vehicle Route Guidance Systems: Classification and Comparison. *IEEE Intelligent Transportation Systems Conference Proceedings*, pages 242–247.

[Srinivas Peeta et al., 2015] Srinivas Peeta, Henry Liu, and Xiaozheng He (2015). Traffic Network Modeling. In *The Routledge Handbook of Transportation*, number September, pages 25–41.

[Taale et al., 2015] Taale, H., Van Kampen, J., and Hoogendoorn, S. (2015). Integrated signal control and route guidance based on back-pressure principles. *Transportation Research Procedia*, 10(July):226–235.

[Tak et al., 2014] Tak, S., Kim, S., and Yeo, H. (2014). Travel time prediction for Origin-Destination pairs without route specification in urban network. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, (May 2015):1713–1718.

[TARKO et al., 1993] TARKO, A., ROUPHAIL, N., and Akcelik, R. (1993). Overflow Delay At a Signalized Intersection Approach Influenced By an Upstream Signal: an Analytical Investigation.

[Tatomir and Rothkrantz, 2006] Tatomir, B. and Rothkrantz, L. (2006). Hierarchical Routing in Traffic Using Swarm-Intelligence. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 230–235. IEEE.

[Treiber and Kesting, 2013] Treiber, M. and Kesting, A. (2013). *Traffic Flow Dynamics*, volume 2013.

[Vetter, 2009] Vetter, G. V. B. D. S. (2009). Time-Dependent Contraction Hierarchies. *Proceedings of the 11th Workshop on Algorithm Engineering and Experiments (ALENEX'09)*, pages 97–105.

[Wardrop, 1952] Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. In *Inst Civil Engineers Proc London/UK/.*

[Watling and van Vuren, 1993] Watling, D. and van Vuren, T. (1993). The modelling of dynamic route guidance systems. *Bilingualism*, 1:159–182.

[Webster, 1958] Webster, F. V. (1958). Traffic signal settings. Technical report.

[Wu et al., 2004] Wu, C. H., Ho, J. M., and Lee, D. T. (2004). Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281.

[Ximin, 2006] Ximin, L. (2006). Based on Hybrid Genetic Algorithm and Cellular Automata Combined Traffic Signal Control and Route Guidance. *2007 Chinese Control Conference*, pages 53–57.

[Yamashita et al., 2005] Yamashita, T., Izumi, K., Kurumatani, K., and Nakashima, H. (2005). Smooth traffic flow with a cooperative car navigation system. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, page 478.

[Yang and Miller-Hooks, 2004] Yang, B. and Miller-Hooks, E. (2004). Adaptive routing considering delays due to signal operations. *Transportation Research Part B: Methodological*, 38(5):385–413.

[Yang et al., 2006] Yang, Z., Lu, S., and Liu, X. (2006). Combined traffic signal control and route guidance: Multiple User Class Traffic Assignment Model versus Discrete Choice Model.

[Ziliaskopoulos, 2001] Ziliaskopoulos, A. K. (2001). Foundations of Dynamic Traffic Assignment : The Past , the Present and the Future. *Networks and Spatial Economics*, 1(3):233–265.

Appendix A

The Optimal Route Problem

Algorithm 2 is a standard procedure, when it is assumed that the algorithm starts from the origin node, for shortest path problems.

Algorithm 2 Standard pseudocode for shortest path algorithms, adapted from [Fu et al., 2006].

Input: V, E, C
Output: P, L

```
l_s = 0
l_{v_{j+1}} = ∞   ∀ v_{j+1} ∈ V \ {s}
p_{v_j} = NULL
Q = {s}
while Q ≠ ∅ do
    Select a junction v_j ∈ Q
    Q = Q \ {v_j}
    for each edge e_k from junction v_j, where e_k = (v_j, v_{j+1}) ∈ E do
        if l_{v_j} + c_{e_k} < l_{v_{j+1}} then
            l_{v_{j+1}} = l_{v_j} + c_{e_k}
            p_{v_{j+1}} = e_k
            Q = Q ∪ {v_{j+1}}
        end if
    end for
end while
```

Where

$p_{v_{j+1}}$: is the preceding edge $e_k = (v_j, v_{j+1})$ on the shortest path to junction v_{j+1},

$l_{v_{j+1}}$: is the cost, commonly referred as the "distance label", of visiting the junction v_{j+1},

c_{e_k} : is the cost to transverse edge k, where $e_k = (v_j, v_{j+1})$, and

Q : is the scan eligible junction set which manages the junctions to be examined during the search procedure.

A.1 Earliest Arrival for Fixed Departure Time

Algorithm 3 Pseudocode to solve the earliest arrival from source s to a destination d using a label correcting algorithm [Dean, 2004].

Input: $t, V,\ E,\ a_{v_j,v_{j+1}}(t)\ \forall\ e_k = (v_j, v_{j+1}) \in E$
Output: $EA_{s,v_{j+1}}(t)$

```
EA_{s,v_j}(t) = ∞   ∀ v_j ∈ V \ {s}
EA_{s,s}(t) = t
Q = {s}
while Q ≠ ∅ do
    Select a junction v_j ∈ Q
    Q = Q \ {v_j}
    for each junction v_{j+1} connected to v_j, where e_k = (v_j, v_{j+1}) ∈ E do
        f(t) = min{EA_{s,v_{j+1}}(t), a_{v_j,v_{j+1}}(EA_{s,v_j}(t))}
        if EA_{s,v_{j+1}}(t) ≠ f(t) then
            EA_{s,v_{j+1}}(t) = f(t)
            Q = Q ∪ {v_{j+1}}
        end if
    end for
end while
```

A.2 Earliest Arrival for Every Departure Time

While the computation of $EA_{s,*}(t)$ gives a single scalar distance label $EA_{s,v_j}(t)$, for every junction $v_j \in V$ (similar to the static shortest path problem); on $EA_{s,*}(*)$, the distance labels for every $v_j \in V$ are the functions EA_{s,v_j} over time that, in discrete time, can be represented by vectors with $t_{max} + 1$ integer components of a finite window with time duration t_{max}. This is better understood by a time-expanded network, G_T, presented in [Dean, 2004] and illustrated in Figure A.1; where each column represents a single junction in V and the rows the earliest arrival at junction v_j. On this way, the network contains $n(t_{max} + 1)$ junctions of the form (v_j, t), where $v_j \in V$ and $t \in \{0, 1, ..., t_{max}\}$; while $O(mt_{max})$ edges of the form $((v_j, t), (v_{j+1}, a_{v_j,v_{j+1}}(t)))$, where $(v_j, v_{j+1}) \in E$, $t \geq 0$, and $a_{j,j+1}(t) \leq t_{max}$.

For the $EA_{s,*}(*)$ problem (in discrete time), besides proposing a decomposition of $EA_{s,*}(*)$ by time into $t_{max}+1$ computations of $EA_{s,*}(t)$ (which would lead to running time of $O(t_{max}(m + n \log n))$), [Dean, 2004] presented three reference solution algorithms: one using a label correcting algorithm; another that is analogous to a label setting algorithm; and the last one based on the parallel computation of two disjoint regions of time and space partitioned based on the $EA_{s,v_j}(t)$, introduced by himself in [Dean, 1999]. However, as the author stated that label-correcting algorithm is not the most efficient for this problem, we present on Algorithm 4 only the algorithm similar to the

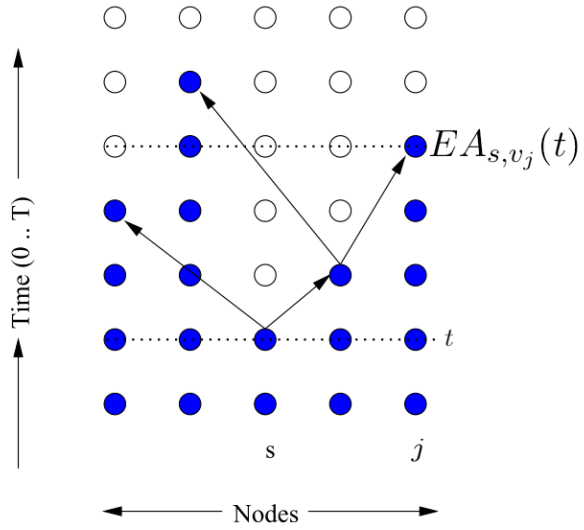


Figure A.1: Illustration of shortest path trees on a time-space diagram (time-expanded network), adapted from [Dean, 2004].

label-setting algorithm. This algorithm finds the shortest path from a set of source junctions $\zeta = \{(s, \tau) : 0 \geq \tau \geq t_{max}\}$ to every junction (v_j, t) by computing the actual values of the EA_{s,v_j} functions. These functions are given by the inverse of the last departure from s to junction (v_j, t), i.e. $(t - D_{s,v_j}(t))^{-1}$; where $D_{s,v_j}(t)$ denotes a distance label function from ζ to each junction (v_j, t) in G_T, while assuming $a_{v_j,v_{j+1}}(t) \geq t \; \forall \; ((v_j, t), (v_{j+1}, a_{v_j,v_{j+1}}(t)))$, and enumerating junctions one "level" at a time in chronological order. It achieves a running time of $O(mt_{max})$, the same as the lower bound for computing $EA_{s,*}(*)$ in discrete time.

Algorithm 4 Pseudocode to solve the earliest arrival from source s to all other junctions [Dean, 2004].

Input: $T, V,\ E,\ a_{v_j,v_{j+1}}(t)\ \forall\ v_j, v_{j+1} \in E$
Output: $D_{s,v_j},\ EA_{s,v_j}$
  for $t \leftarrow 0$ to t_{max}, where $t \in T$ **do**
    $D_{s,v_j}(t) = \infty \quad \forall\ v_j \in V \setminus \{s\}$
    $D_{s,s}(t) = 0$
  end for
  for $t \leftarrow 1$ to t_{max}, where $t \in T$ **do**
    for each $v_j \in V$ **do**
      $D_{s,v_j}(t) = \min\{D_{s,v_j}(t), D_{s,v_j}(t-1)\}$
    end for
    for each $(v_j, v_{j+1}) \in E$ **do**
      if $a_{v_j,v_{j+1}}(t) \geq t_{max}$ and $D_{s,v_{j+1}}(a_{v_j,v_{j+1}}(t)) > D_{s,v_j}(t) + a_{v_j,v_{j+1}}(t) - t$ **then**
        $D_{s,v_{j+1}}(a_{v_j,v_{j+1}}(t)) = D_{s,v_j}(t) + a_{v_j,v_{j+1}}(t) - t$
      end if
    end for
  end for

A.3 Speed-up Techniques

Speed-up techniques have been developed to avoid unnecessary computation. Such techniques have an offline phase (called preprocessing), which computes additional data, that accelerates queries during the online phase. Usually the offline phase run much longer times by exploit several properties of a transportation network during, while the online phase obtains the quickest path within microseconds. There are three common speed-up techniques, according to [Delling and Wagner, 2009], they are briefly described in the following list.

- *Landmark-based ALT (A*, Landmarks, Triangle inequality) algorithm* [Goldberg and Harrelson, 2005], defines a set of junctions, called landmarks, and precomputes and stores the shortest paths between all the junctions and all landmarks in the network to answer a query faster than a normal calculation. Its query is unidirectional and similar to Dijkstra, but the priority queue is also determined by the lower bound of distance $h(v_j)$ to s given by the landmarks.
- *Contraction Hierarchies (CH)* ([Vetter, 2009], [Batz et al., 2010]), are based on the idea of removing unimportant junctions from the graph and adding shortcuts to preserve distances between non-removed junctions, while restricting the scope of a search from the source by a set of junctions defined a search from the destination. During preprocessing, each junction is assigned a priority based on it importance in an n-level hierarchy, then the graph is split into two: one storing edges directing

from unimportant to important junctions, and another vice-versa. Its query is conducted of two Dijkstra searches, a forward search on the unimportant to important junctions graph, while a backward search on the graph with the opposite.

- *SHARC (Shortcut+ArcFlags) routing* [Delling, 2011], uses an arc-flag approach, which first computes a partition of the graph and then attaches a label to each edge e_k (true or false) if a shortest path to at least one junction in the partition starts with e_k. The algorithm extracts the maximal junction induced subgraph of minimum junction degree 2 and perform a multi-level partition of the graph. Then, iteratively, it removes edges and sets their arc-flags as true if their junctions are a core junction or if a shortest path starts with the edge. Afterwards, it refines the arc-flags of edges removed and finally reattach the junctions not in the core removed at the beginning. Its query is similar to Dijkstra, but needs to compute the common level of the current junction and the destination, and the arc-flags evaluation.

Some issues brought by [Demiryurek et al., 2011] is that even though the Contraction Hierarchies (CH) and SHARC methods were modified to time-dependent road networks, the importance of the junctions can be also time-varying and extensive preprocessing times. In SHARC, whenever an edge cost function changes, arc flags should be recomputed even when the graph partition don't need to be updated. In CH, the space consumption is at least 1000 bytes per junction for less varied edge-weights where the storage cost increases with real-world time-dependent edge weights. However, their concerns were over application of such techniques in continental size road networks. For instance, [Delling and Wagner, 2009] states that although a SHARC variant called Aggressive SHARC (which uses exact flags instead of approximate flags) has, indeed, high preprocessing times, but also the lowest query times. On the other hand, [Nejad et al., 2016] points out that ALT-based methods have a trade off between choosing well-positioned landmarks and preprocessing time, requiring large memory space, making them ineffective for large road networks and vehicles without online information. Additionally, although some speed-up techniques are based on A*, time-dependent A* algorithms do not require storage of preprocessed shortest paths, short-cuts, or lower bounds.

The time-dependent shortest path is approached in [Nejad et al., 2016] by exploiting an efficient agglomerative approach to construct road network representations through hierarchical community structure detection. The so called Time-Dependent Goal Directed (TNGD) algorithm reduces the search space in each level of the hierarchy, determining a spectrum of promising communities for exploration in each level of the hierarchy. Then, a Hierarchical Time-Dependent Goal Directed (HTNGD) algorithm searches over the entire hierarchical representation of the road network, recursively calling the TNGD for the level below, from the highest level and terminating at the lowest level with the TNGD identifying the shortest path. Each upper level has

a virtual junction as the centre of a community where communities are connected through inter-community arcs, while communities are chosen if the lowest level of the hierarchy structure contains at least one path from source to destination. The procedure to find the minimum arrival time from the community with the source to the communities to be relaxed are similar to A* (including heuristic function for lower bounds), but using also S and N to store a set of visited communities and a set of communities to visit in the next iteration, respectively. This avoid removing some promising communities by extending the search space by adding neighbour communities to the selected communities in the core set. The proposed algorithm (HTNGD) generates routes in real time in terms of milliseconds on large-scale networks without storing a large number of precomputed shortest paths and lower bounds.

[L. Zhao, 2008] presents a novel of a non-trivial generalization framework of the A* algorithm. It generalizes the heuristic function for each junction v_j, $h(v_j)$, by the time-dependent version of it (using the notation from [Dean, 2004]) $h(v_j, EA_{s,v_j})$. This heuristic function is a lower bound of the shortest travel time from an intermediate junction v_j to a destination d with departure time (distance label) EA_{s,v_j}. They apply this modified A* into the landmark-based ALT algorithm [Goldberg and Harrelson, 2005], in which they also employs the triangle inequality to calculate the heuristic function, given by the difference between the shortest travel time from a landmark junction z to d and shortest travel time from z to a intermediate junction v_j (if positive), but considering the latest departure time $\hat{t}$ (found by sampling of time) from z to v_j. Their algorithm is several times faster than the generalized Dijkstra algorithm for the time-dependent problem.

Based on the same concept ALT algorithm (but using a bidirectional search), [Nannicini et al., 2008] proposed a novel bidirectional A* Search for time-dependent fast paths that works in three phases: 1) the forward search (from s to d) is run on the graph with time-dependent arc cost function c, while the backward search (from d to s) run on the graph weighted by the lower bounding function $h(v_j)$ including in a set M all finished junctions; 2) when the two search scopes meet, both search scopes are allowed to proceed, but until the minimum element of the backward search queue exceeds the time dependent cost of the path (from s to d through the junctions of the heaps of the forward and backward search), and including in a set M all finished junctions by the backward search; 3) only the forward search continues until d is finished, with the additional constraint that only junctions M can be explored. The results of their algorithm are several times faster than the Dijkstra's algorithm, but only when finding slightly suboptimal solutions, while the exact version is slower than unidirectional ALT.

Another bidirectional time-dependent A* Search algorithm for fast paths is presented in [Demiryurek et al., 2011]. Beginning with a offline phase, the road network is partitioned into non-overlapping areas based on the hierarchy of roads (though the algorithm don't depend on this step), and assign at least one partition to each junction (junctions with more than one partition

are called border junctions). Then, it is precomputed the lower-bound travel time (i.e. distance label) border-to-border, junction-to-border, and border-to-junction of the lower-bound graph to find the heuristic functions, while updating the distances labels only when the minimum travel time of an edge changes. On the online phase (similar to the rules used in [Nannicini et al., 2008]), they run the backward search on the lower-bound graph to filter-in the set of the junctions that needs to be explored by the forward search. Meanwhile, the forward search is a best-first search which scans junctions based on their time-dependent cost label and lower-bound of the distance $h(v_j)$(maintained in a priority queue). The results of their analysis showed that their algorithm provides better heuristic function (i.e. lower-bound estimation) compared to the standard ALT algorithm, and outperform the other common approaches in storage and response time.

Appendix B

Edge Cost Estimation

Typically, edge travel time is an increasing function of the edge flow due to congestion, as shown in Figure B.1, using the Bureau of Public Roads (BPR) function as edge performance function [Srinivas Peeta et al., 2015]:

$$TT_k = TT_k^0 \left[1 + \beta \left(\frac{Q_k}{C_k}\right)^{\gamma}\right], \tag{B.1}$$

where TT_k is the total travel time, TT_k^0 the free flow travel time, Q_k the flow, and C_k the capacity of edge k (that comes from edge k, e_k). The parameters, generally $\beta = 0.15$ and $\gamma = 4$, determine the edge travel time increase in proportion to the volume-to-capacity ratio.

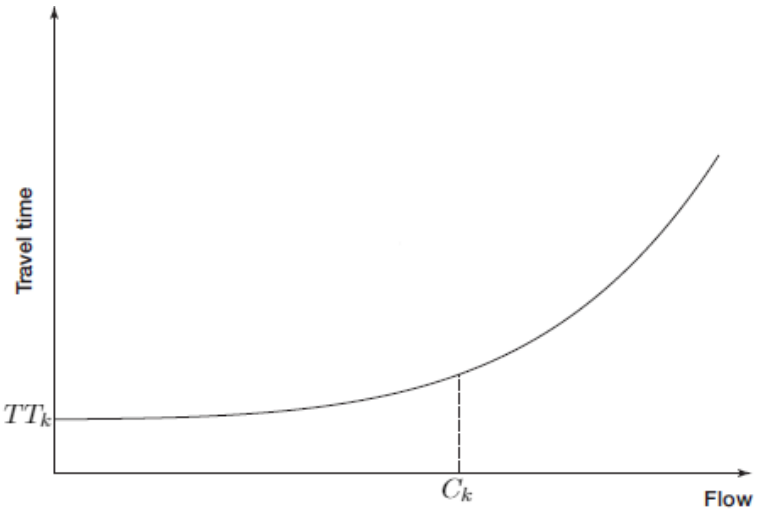


Figure B.1: Bureau of Public Roads (BPR) function, adapted from [Srinivas Peeta et al., 2015].

However, the BPR function (Eq. B.1) is not realistic because it does not consider queue delays, either by interactions, road capacity constraints and traffic signals. [Srinivas Peeta et al., 2015] presented a dynamical system for a general time-dependent edge travel time function TT_k (referred as edge performance function) based on the dynamic assignment formulation

in [Carey, 2001], to describe the flow dynamics when depending only on the traffic volume:

$$\begin{cases} \dot{Q}_k(t) = in_k(t) - out_k(t) \\ TT_k = f(Q_k(t)) \\ out_k(t + TT_k) = in_k(t) / \left[1 + \dot{TT}_k\right], \end{cases} \tag{B.2}$$

where $\dot{Q}_k(t)$ is the variation of the flow of vehicles, $in_k(t)$ the inflow, and $out_k(t)$ the outflow on edge k at time t. Moreover, the travel time function $TT_k = f(Q_k(t))$ can be specified by:

- a linear delay function,

$$f(Q_k(t)) = \alpha + \beta Q_k(t), \quad \alpha > 0,\ \beta > 0; \tag{B.3}$$

- a piece-wise linear function,

$$f(Q_k(t)) = \begin{cases} \alpha & if \quad Q_k(t) < \alpha/\beta \\ Q_k(t)/\beta & otherwise; \end{cases} \tag{B.4}$$

- a smooth and convex functions satisfying the conditions,

$$\frac{f(Q_k(t_2)) - f(Q_k(t_1))}{Q_k(t_2) - Q_k(t_1)} \cdot \frac{Q_k(t_2) - Q_k(t_1)}{t_2 - t_1} > -1, \quad \forall t_2,\ t_1, \tag{B.5}$$

$$f''(Q_k)\ is\ continuous, \tag{B.6}$$

$$f(0) = \alpha, \tag{B.7}$$

$$f' > 0,\ f''(Q_k) > 0 \quad \forall\ Q_k > 0,\ and \tag{B.8}$$

$$f'(Q_k) \to \beta\ as\ Q_k \to \infty. \tag{B.9}$$

A further step is to calculate the travel time function that includes traffic signal delay, formulated as it follows: [Srinivas Peeta et al., 2015]:

$$TT_k = TT_k^0 + d_k, \tag{B.10}$$

where $d_k = f(Q_k, \boldsymbol{\lambda}_j, t)$ is the signal delay function, and $\boldsymbol{\lambda}_j$ is the green split of the traffic signal at junction (vertex) $v \equiv v_{j+1}$. The estimation of delay d_k and queue length q_k is part of the theory of traffic signals, in which the adoption of a signal control strategy at either individual junctions or a sequence of them may influence the formation of queues [FHWA, 2018]. This formulation of the modelling travel time problem is generic. To solve the problem considering either changes of traffic signals or queueing, it is necessary to use traffic flow models.

B.1 Traffic Theoretic Models

B.1.1 Macroscopic (Continuum Traffic Flow) Models

Macroscopic flow models (also called *hydrodynamic models* or *continuum traffic flow models* in [Srinivas Peeta et al., 2015]) describe traffic at a high level of aggregation as a flow without distinguishing its constituent parts, i.e. the traffic stream characteristics is locally aggregated (they vary over time and space — which can be a section of a edge k — forming *dynamic fields* represented by flow-rate $Q(t)$, density $\rho(t)$, mean speed $V(t)$, and speed variance $\sigma_V^2(t)$. The traffic flow is analogous to the motion of liquids or gases, modelling the collective phenomena such as evolution of congested regions or the propagation velocity of traffic waves between junction or long segment of a road, while used particularly for traffic state estimation. On this way, macroscopic models may assume the traffic stream is properly allocated to the roadway lanes, neglecting individual vehicle manoeuvres, such as a lane change. Macroscopic models have been developed to describe the speed-density-flow relationship (the so called fundamental diagram illustrated in Figure B.2), in which such models can be classified according the number of partial differential equations and its order on modelling speed or flow ([Treiber and Kesting, 2013], [Hoogendoorn and Bovy, 2001]).

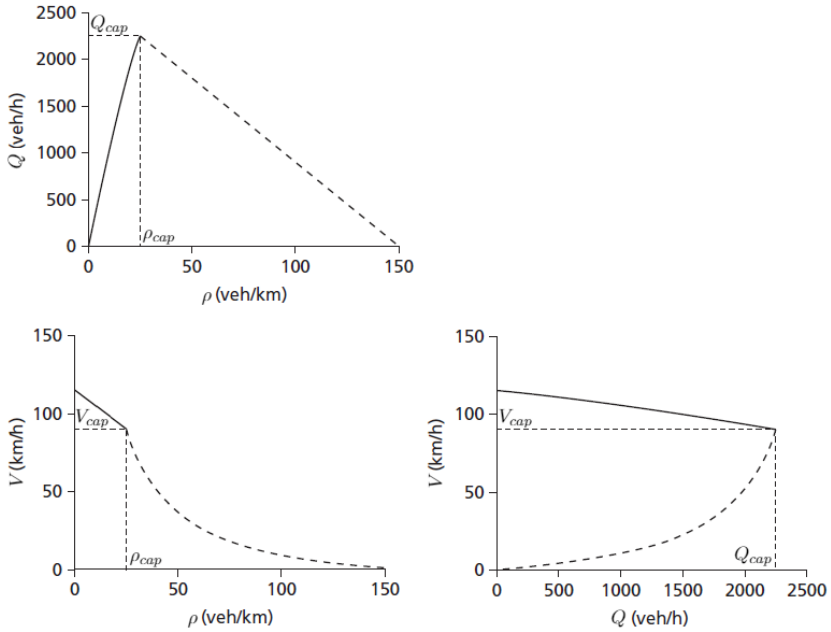


Figure B.2: Speed-density-flow relationship, example of the fundamental diagram adapted from [Hoogendoorn and Knoop, 2012].

One underlying principle of macroscopic models is the hydrodynamic flow-density relation:

$$Q = \rho \cdot V, \tag{B.11}$$

where Q is the flow, ρ the density, and V the speed. The hydrodynamic flow-density relation can be seen in Figure B.2, in which the macroscopic vehicle speeds satisfy the hydrodynamic relation (Equation B.11). In addition, the subscript "cap" represents the variable values at maximum capacity levels, as a common practice is to divide the road edge k into small segments (also referred as cells or sections) $(s, s + ds)$ dependent on time t, e.g. $Q_k(s, t)$.

Another one is the continuity equation, which derives the conservation of vehicles in terms of the traffic density according to the geometry of the road infrastructure (e.g. on and off ramps and change in number of lanes). It describes the rate of change of density in terms of differences of the flow, either from the point of view of a stationary observer (Eulerian representation) or a vehicle driver (Lagrangian representation). Its simplest form is (without entrance and leaving of vehicles within the edge and change of lane number) [Treiber and Kesting, 2013]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial Q}{\partial s} = 0. \tag{B.12}$$

Equation B.12 and Figure B.3 show that the number of vehicles in segment s increases according to the balance of inflow at the boundaries s and $s + ds$ of the segment. However, one may notice that the continuity equation would hold only when the speeds are constant, however if we use the space-mean speed on segment s of edge k, $V(s, t)$, the continuity equation is still valid [Hoogendoorn and Knoop, 2012].

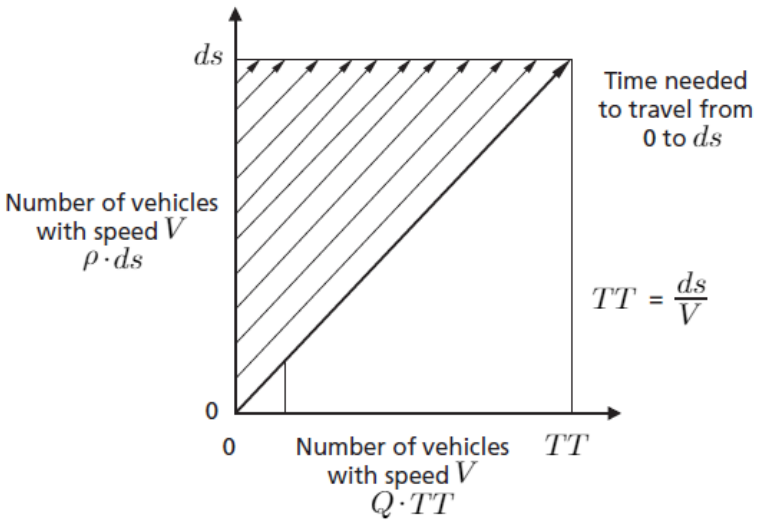


Figure B.3: Derivation of continuity equation and travel time on edge segment, adapted from [Hoogendoorn and Knoop, 2012].

Additionally, we can simply define the travel time as a function of the speed on the edge given each segment length ds:

$$TT(s,t) = \frac{ds}{V(s,t)}. \tag{B.13}$$

However, as Equations B.11 and B.12 form a system of two independent equations and three unknown variables, a third independent model equation is needed, either for traffic flow or for speed [Hoogendoorn and Bovy, 2001].

The *Lighthill–Whitham–Richards (LWR) models* assume that traffic flow and local speed change instantaneously accordingly to density at any situations when the edge is homogeneous. This class of models (also called *first-order models*) have only one dynamic equation (the continuity equation). This means that the relationships seen in Figure B.2 are usually determined by fitting speed-density or flow-density data, as follows [Treiber and Kesting, 2013]:

$$Q(s,t) = Q_e(\rho(s,t)), \tag{B.14}$$

or

$$V(s,t) = V_e(\rho(s,t)). \tag{B.15}$$

These static relations complement the continuity equation, in which Q_e and V_e correspond to the local flow equilibrium and local speed equilibrium models, respectively. However, such relations might coincide with field collected data if there are systematic errors during the measurements process, and the traffic flow is neither at equilibrium nor homogeneous. One simple model for $V_e(\rho(s,t))$ is the modified *Greenshield's model*, presented in [Srinivas Peeta et al., 2015]:

$$V_e(\rho(s,t)) = V_0 + (V_f - V_0)\left(1 - \frac{\rho(s,t)}{\rho_{max}}\right)^{\beta}, \tag{B.16}$$

where for edge k we have the following values for each segment s: V_0 as the minimum speed, V_f the free-flow speed, $\rho(s,t)$ the segment density at time t, ρ_{max} the maximum density, and β a user-specific parameter. There are many other models to compute $V(s,t)$; however, as pointed out in [Liang and Wakahara, 2014], the speed information should be measured rather than calculated using the conventional speed-density-flow relationship. This is due the fact that the speed in a urban road edge depends mainly on the type and the geometry of the edge. For the flow-density relation $Q_e(\rho(s,t))$, the simplest model is the *Triangular Fundamental Diagram* (TFD) [Treiber and Kesting, 2013]:

$$Q_e(\rho(s,t)) = \begin{cases} V_0 \cdot \rho(s,t) & if\ \rho(s,t) \leq \rho_{cap}(s,t) \\ \frac{1}{TG}(1 - \rho(s,t) \cdot l_{eff}) & if\ \rho_{cap}(s,t) < \rho(s,t) \leq \rho_{max}, \end{cases} \tag{B.17}$$

where the first case represents free flow, the second congested,

TG = mean desired time gap parameter of 1.4s for highways and 1.2s for city traffic (s/lane),

ρ_{max} = maximum density on the road of 0.12 for highways and city traffic (veh/(m · lane)),

$\rho_{cap} = {}^{1}/_{[(V_0 \cdot TG)+l_{eff}]}$ = density at capacity level of the segment s,

V_0 = mean desired speed of the edge which depends on each case but usually 27.8 for highways and 13.9 for city traffic (m/s),

$l_{eff} = DG + l_{ave} = {}^{1}/_{\rho_j}$ = mean effective vehicle length (m),

DG = average minimum gap in stopped traffic (m), and

l_{ave} = average vehicle length (m).

The solution for such model relies on the application of Eqns. B.14 and B.15 to the continuity equation B.12, what enables us to get simplest LWR model (without ramps and change of lane number) [Treiber and Kesting, 2013]:

$$\frac{\partial \rho}{\partial t} + \frac{dQ_e(\rho)}{d\rho} \cdot \frac{\partial \rho}{\partial s} = 0. \tag{B.18}$$

This partial differential equation is a non-linear wave equation that describes the propagation of kinematic waves. This is particular important to compute the propagation velocity of density variations along the edge because such velocity is proportional to the gradient of the steady-state flow-density relation. By applying the continuous version of Eq. B.18 with the triangular fundamental diagram (TFD), we get the so called *section-based model*. As there are only two propagation velocities of density variations (free flow and congested traffic), it is possible to define each section (segment) end border at an inhomogeneity or bottleneck (but each road section there is at most one jam front). Then, Eq. B.18 is solved only for a single integral for the motion of the jam front, because the inflow at the upstream end of each section is given either by the outflow of an adjacent section or by the source boundary conditions of the simulated system. Another possibility is the discrete version known as *Cell-Transmission Model* (CTM), where time and space are discretized into time steps and cells/segments (road is represented as a collection of equal length cells), and supplemented by a "supply-demand" update rule (illustrated in Figure B.4) [Treiber and Kesting, 2013]. As CTM is a convergent numerical approximation to the hydrodynamic model and replicates kinematic waves, queue formation, and dissipation, it could be suitable for modelling dynamic traffic [Yang et al., 2006]. Usually, on CTM the length of each segment/cell, ds, is equal to the distance that a single vehicle traverses in one time step at the free-flow speed [Yang et al.,

2006]. However, the time step duration Δt should comply with the following limitation [Treiber and Kesting, 2013]:

$$\Delta t < \frac{ds}{V_0}. \tag{B.19}$$

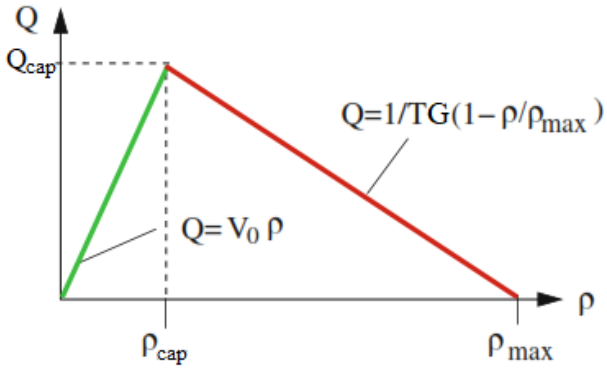


Figure B.4: Triangular fundamental diagram used in the cell-transmission model and the section-based model, adapted from [Treiber and Kesting, 2013].

Equation B.17 requires many average parameters that may lead to systematic errors. Under our scenario that we have information about signal timings, queue length estimation and future arrivals, we can estimate the parameters by local measurements [Treiber and Kesting, 2013]:

$$\rho_{max} = Q_{max}\left(\frac{1}{V_0} - \frac{1}{\omega}\right), \tag{B.20}$$

$$TG = \frac{1}{Q_{max}\left(1 - \frac{\omega}{V_0}\right)}, \tag{B.21}$$

where Q_{max} is the dynamic capacity of a lane in road section based on the outflow of moving traffic waves (propagation of density changes) whose velocity is ω. This can be very important once we know that discharging vehicles influenced by upstream traffic light will not drive as in steady-state with section capacity Q_{cap}. We should notice that this outflow corresponds to variations of the capacity the homogeneous section Q_{cap} and before the bottleneck, it is different from the capacity of the bottleneck (e.g a traffic light), C that could represent the saturation flow. The propagation velocity ω can be determined by the time interval which is needed by the downstream front of a moving traffic wave to pass two consecutive stationary detectors. Nevertheless, they can be both computed as [Treiber and Kesting, 2013]:

$$Q_{max} = \frac{1}{TG + \left(\frac{l_{eff}}{V_0}\right)} = \frac{1}{TG\left(1 + \frac{|\omega|}{V_0}\right)}, \tag{B.22}$$

$$\omega = -\frac{1}{\rho_{max} \cdot TG} = -\frac{l_{eff}}{TG} = \frac{\Delta Q_e}{\Delta \rho}, \tag{B.23}$$

where ΔQ and $\Delta \rho$ represent the variations of flow and density respectively. Therefore, the triangular model (in this case, the CTM model) can be also represented as:

$$Q_e(\rho(s,t)) = \begin{cases} V_0 \cdot \rho(s,t) & if\ \rho(s,t) \leq \frac{Q_{max}}{V_0} \\ Q_{max}\left[1 - \frac{\omega}{V_0}\right] + \omega \cdot \rho(s,t) & if\ \rho(s,t) \leq \frac{Q_{max}}{V_0}, \end{cases} \tag{B.24}$$

in which the first case represents free flow, the second congested. Additionally, to model bottlenecks in LWR models, we need to use reduction of the section capacity, denoted as ΔQ_{cap}. For this, the idea is to model changing of drivers' behaviour, i.e. the mean desired speed V_0 and/or the time gap TG. While the average vehicle length l_{eff} and hence ρ_{max}, both remains the same. However, we should consider this reduction on the lane level ΔQ_{max}, computed as:

$$\Delta Q_{max} = \frac{V_1^0}{V_1^0 \cdot TG_1 + l_{eff}} - \frac{V_2^0}{V_2^0 \cdot TG_2 + l_{eff}}, \tag{B.25}$$

where the subscripts 1 and 2 represents the fronts (shockwaves) of the different density areas. On this way differences on desired speeds and time gaps model bottlenecks. There are other bottleneck situations, but for our urban edge environment, the most important is the consideration of traffic light, as follows [Yang et al., 2006]:

$$\Delta Q_{max} = \begin{cases} Q_{max} & if\ effective\ red \\ 0 & if\ effective\ green. \end{cases} \tag{B.26}$$

We see that traffic lights are modelled by a time-dependent flow-conserving bottleneck: during the red the bottleneck capacity is equal to zero, otherwise the bottleneck disappears.

Finally, as we aim to calculate the travel time on a edge, we can obtain the The Total Travel Time Spent (TTS_{total}) by the sum of the TTS for each segment TTS_s. It represents the time that all vehicles spent crossing the segment during the analysed period $[0, t_{max}]$, calculated as follows [Treiber and Kesting, 2013]:

$$TTS_s = \int_0^{t_{max}} ds \cdot \rho(s,t) dt = \int_0^{t_{max}} \frac{ds}{V(s,t)} Q(s,t) dt. \tag{B.27}$$

As higher-order models have limited benefit and does not justify their added complexity [Ni, 2016], we will consider only these first-order models. However, on one hand, models with dynamic speed (second-order macroscopic models and most microscopic models) even a local increase of capacity can lead to congestion, due speed perturbations and traffic instabilities. On the other hand, first-order models assume congestion only due bottleneck caused by lowered capacity [Treiber and Kesting, 2013]. Another issue with macroscopic

models is that even with perfect information of the signal timings, and time of arrival of a vehicle at an junction, the delay due to queue clearance is a function of the vehicle's location in the queue [Yang and Miller-Hooks, 2004], and the vehicle position in the queue cannot be known with certainty due to the high aggregation level of macroscopic models.

B.2 Queueing Models

B.2.1 Steady-State Queuing Models

Steady-state queuing (or delay) models include randomness of arrivals and (some of them overflows and non-constant departure process), which compute junction delays based on statistical distributions of the arrival and departure processes. They assume that, after sufficient time, the state of the system is independent from its initial state and the elapsed time. In this case, stochastic equilibrium is achieved and the expected queues and delays are finite and can be estimated (the so called *undersaturated* conditions). The condition of stochastic equilibrium occurs only if the degree of saturation x is below 1 [FHWA, 2018]:

$$x = \frac{Q/S}{g/c} < 1. \tag{B.28}$$

Otherwise they generate infinite delays at saturation degree ($x = 1$). Specifically for undersaturated conditions, the first traffic signal delay model was proposed by [Webster, 1958] combining theoretical and numerical simulation approaches:

$$d = \frac{c\,(1 - g/c)^2}{2\,[1 - (g/c)\,x]} + \frac{x^2}{2Q\,(1 - x)} - 0.65\left(\frac{c}{Q^2}\right)^{1/3} x^{2+5(g/c)}, \tag{B.29}$$

where

d = average delay per vehicle (sec),

c = cycle length (sec),

g = effective green time (sec),

x = degree of saturation (flow to capacity ratio, 1 - dimensionless), and

Q = arrival rate (veh/sec).

The first term in Eq. B.29 corresponds to the delay when traffic is arriving at a uniform rate (average values), d_u. The second term represents the random nature of the arrivals (i.e. Poisson) and departures (outflow) at constant rate of the signal capacity, d_o. The third term is a calibration (based on simulation experiments) that typically ranges 10 percent of the first two

terms. However, Eq. B.29 only approximates the expression for delay, and the higher the saturation degree the more inaccurate it is [Ohno, 1978].

Later work, still assuming undersaturated conditions (but accounting explicitly some expected overflow queue), provides a more precise calculation of delay considering a compound Poisson arrival process and general departure process ([Gazis, 1974] apud [FHWA, 2018], [Miller, 1963]):

$$d = \frac{(c-g)}{2c(1-Q/S)}\left\{(c-g) + \frac{2}{Q}\left[1 + \frac{(1-Q/S)(1-B^2)}{2S}\right]q_0 + \frac{1}{S}\left(1 + \frac{I + B^2 Q/S}{1-Q/S}\right)\right\}, \tag{B.30}$$

where

d = average delay per vehicle (sec),

c = cycle length (sec),

g = effective green time (sec),

Q = arrival rate (veh/sec),

S = saturation flow (veh/sec),

I = index of dispersion for the arrival process (dimensionless),

B = index of dispersion for the departure process (dimensionless), and

q_0 = (average) overflow queue at beginning of green due randomness (veh).

The indexes of dispersion are defined as:

$$B = \frac{\mathrm{Var}(D)}{\mathrm{E}(D)}, \ \textit{and} \tag{B.31}$$

$$I = \frac{\mathrm{Var}(A)}{\mathrm{E}(A)} \approx \frac{\mathrm{Var}(\Delta D)}{\mathrm{E}(D-A)}, \tag{B.32}$$

in which A is the number of arrivals and D the maximum number of departures in one cycle. The ΔD represents the reserve capacity (veh) in one cycle, calculated as:

$$\Delta D = \begin{cases} D - q(0) - A & if\ q(0) + A < D \\ 0 & otherwise, \end{cases} \tag{B.33}$$

where $q(0)$ is the initial queue at the begin of green. If there is no overflow ($q_0 = 0$) and no random arrivals ($I = 0$), Equation B.30 calculates the uniform component of the delay d_u seen in Figure 4.8. Additionally, Equation B.30

requires the (average) overflow queue q_0, which can be approximated for any arrival and departure distributions by [Miller, 1963]:

$$q_0 \approx \begin{cases} \frac{(2x-1)I}{2(1-x)} & if\ x \geq 0.5 \\ 0 & otherwise, \end{cases} \tag{B.34}$$

where x is degree of saturation (flow to capacity ratio) and I is the index of dispersion for arrivals. The delay calculations on Equations B.29 and B.30 are average values considering the average number of vehicles in the cycle $(c \cdot Q)$, thus an application intended to compute total delay in the cycle for a specific signal should make $d_{all} = d \cdot c \cdot Q$.

Steady-state delay models are applicable at low flow to capacity ratios $(x < 0.8)$, when equilibrium is reached in a reasonable period of time.

B.2.2 Time-Dependent Queuing Models

A popular approach to derive a time dependent formula for delays more realistic than those in steady-state queueing theory, specially when the flow approaches the signal capacity is called *coordinate transformation technique*. Originally proposed by Whiting (unpublished) [FHWA, 2018] and further explored by [Kimber and Hollis, 1979], this technique shifts the original asymptotic steady-state curve to the deterministic oversaturation delay line as seen in Figure B.5. However, there are two important restriction of the coordinate transformation technique, listed in the following list.

- Flow (inflow) Q is a constant rate over the interval $[0, t_{max}]$, what is possible to deal with step-functions.
- No initial queue at the beginning of the interval $[0, t_{max}]$, though the [Kimber and Hollis, 1979] model (described in the following Equation B.35) assumes a possible initial queue $q(0)$ and the initial queue component of delay d_q is used to compensate this issue on other time-dependent models when applied with a correction of the uniform delay d_u (see Equation B.42).

[Kimber and Hollis, 1979] proposed a model that accounts how the queue length evolves over time, by splitting the delay into the steady-state delay and the average delay per arriving vehicle during time interval $[0, t_{max}]$ (as a deterministic model). The idea is that the non-stationary arrival process is approximated with a step-function, and the total delay calculated by integrating the queue size over time. The final transformed time dependent equation is:

$$d = \frac{1}{2}\left[(\beta^2 + \gamma)^{1/2} - \beta\right], \tag{B.35}$$

$$\beta = \frac{t_{max}}{2}(1-x) - \frac{1}{C}\left[q(0) - B + 2\right], \tag{B.36}$$

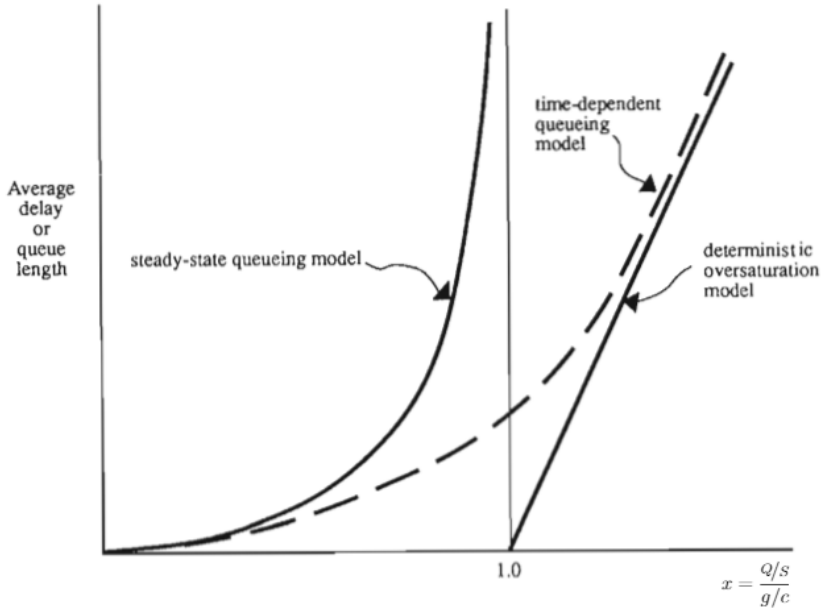


Figure B.5: Applicability of queueing models, adapted from [Rouphail M and Akcelik, 1992].

$$\gamma = \frac{4}{C}\left[\frac{t_{max}}{2}(1-x) + \frac{1}{2}(x \cdot t_{max} \cdot B) - \left(\frac{q(0)+1}{C}\right)(1-B)\right], \qquad \text{(B.37)}$$

where

d = average delay per vehicle (sec/veh),

x = degree of saturation (dimensionless),

C = signal capacity (veh/sec),

$q(0)$ = initial queue at time $\tau = 0$ (veh),

B = index of dispersion for departure process,(dimensionless), and

t_{max} = interval constant demand (sec).

Later, [Akcelik, 1988] introduced a generalized time-dependent expression for delay, where the first term represents the uniform delay d_u (the same as in Webster's formula B.29) and the second term the overflow delay d_o:

$$d = \frac{0.5c(1-g/c)^2}{1-(g/c)x} + 900t_{max}(x^{\gamma})\left[(x-1) + \sqrt{(x-1)^2 + \frac{\beta(x-x_0)}{C \cdot t_{max}}}\right], \qquad \text{(B.38)}$$

where

d = average overall delay (including stop-start delays) per vehicle (sec/veh),

$x_o = \alpha + [(S \cdot g)/\delta]$,

C= signal capacity (veh/sec),

t_{max} = interval constant demand (sec), and

$\alpha, \beta, \gamma, \delta$ calibration parameters (dimensionless) available in [Akcelik, 1988].

Even though Equation B.38 is generalized, [Bureau of Public Roads, 1950] does not recommend to use it for the delay calculation for protected-plus-left-turn movements (meaning both) This is due the fact that permitted movements can be based on saturation flow or probability of accepted time gaps while protected left-turning movements behave exactly as through movements, what is the case of the equation. Additionally, Equation B.38 (and also other time-dependent models such Equations 4.10 and 4.4) does not account the delay due an initial queue $q(0)$, considering only the uniform and overflow components (d_u and d_o, respectively) of the delay d. For the initial queue component d_q, [Bureau of Public Roads, 1950] suggests:

$$d_q = \frac{1800 \cdot q(0) \cdot (1+\epsilon) \cdot t_{unmet}}{C \cdot t_{max}}, \tag{B.39}$$

where ϵ is a delay parameter and t_{unmet} the duration of unmet demand during $[0, t_{max}]$. It is given by:

$$t_{unmet} = \begin{cases} 0 & if q(0) = 0 \\ \min\left\{t_{max}, \frac{q(0)}{C[1-\min(1,x)]}\right\} & otherwise, \end{cases} \tag{B.40}$$

$$\epsilon = \begin{cases} 0 & if\ t_{unmet} < t_{max} \\ 1 - \frac{C \cdot t_{max}}{q(0)[1-\min(1,x)]} & otherwise. \end{cases} \tag{B.41}$$

We should mention that [Bureau of Public Roads, 1950] also recommends that when using Equation B.38 with the presence of initial queue $q(0)$ and $d_q > 0$, it is necessary to change the calculation of the first term of Equation B.38 (which corresponds to the uniform component of the delay d_u) by:

$$d_u = d_s \cdot \left(\frac{t_{unmet}}{t_{max}}\right) + d_{us} \cdot PF \cdot \left(\frac{t_{max} - t_{unmet}}{t_{max}}\right), \tag{B.42}$$

where

d_s = saturated delay (d_u evaluated with $x = 1$),

d_{us} = undersaturated delay (d_u evaluated with actual x value), and

PF = progression factor parameter that we will discuss on the next section.

As $q(0)$ can be defined by the distribution of the overflow q_0, [Akcelik, 1980] proposed an estimate to the average overflow queue q_0 for oversaturated conditions:

$$q_0 = \begin{cases} \frac{C \cdot t_{max}}{4} \left[(x-1) + \sqrt{(x-1)^2 + \left(\frac{12(x-x_o)}{C \cdot t_{max}} \right)} \right] & when\ x > x_o \\ 0 & otherwise, \end{cases} \tag{B.43}$$

where $(x_o = \alpha + [(S \cdot g)/\delta])$ represents the degree of saturation below which the overflow delay of the delay formula is zero, and α, δ calibration parameters available in [Akcelik, 1988]. However, Equation B.43 is relevant only for high degrees of saturation $(x > 0.8)$, being Equation B.34 more suitable when $(0.5 \leq x < 0.8)$. Meanwhile in [Bureau of Public Roads, 1950], it is proposed a calculation for the time instant in which the initial queue $q(0)$ is totally cleared from the start of the time period t_{max}:

$$t_{clear} = \max \left(t_{max}, \frac{q(0)}{C} + t_{max} \cdot x \right), \tag{B.44}$$

where t_{clear} is the time instant $q(0)$ is totally discharged, t_{max} the time instant to begin the period time, C the signal capacity, and x the degree of saturation. One may notice that in case it is desired only the clearance time of initial queue (suppose $t_{clearance}$, we need to make:

$$t_{clearance} = t_{clear} - t_{max}. \tag{B.45}$$

B.2.3 The Influence of Arrival Profile

The arrival profile in real urban networks express the following characteristics [FHWA, 2018]:

- vehicles cross the junction in "bunches" with similar headways in which such groups are separated by the red signal time (platooning effect); and
- the number of vehicles discharged from upstream signalized junction is constrained by its corresponding throughput in one cycle D (filtering effect).

In order to deal with the influence of upstream traffic control on vehicle delays but maintaining the assumptions for isolated junctions, the [Bureau of Public Roads, 1950] uses a Progression Factor (PF) applied to the delay computed for an isolated junction signal:

$$d = d_u(PF) + d_o + d_q, \tag{B.46}$$

where

d = average delay per vehicle (sec/veh),

PF = uniform delay progression adjustment factor that accounts for effects of signal progression (dimensionless),

d_u = uniform delay under uniform arrival (sec/veh),

d_o = overflow delay due random arrivals and oversaturation queues, which is adjusted for an specific time interval and signal control strategy $[0, t_{max}]$, and

d_q = initial queue delay (sec/veh).

The PF value is computed based on field measurement and applying the following formula:

$$PF = \frac{(1 - P_g) f_{PA}}{1 - (g/c)}, \tag{B.47}$$

where

P_g = proportion of vehicles arriving (at stop line or join the moving or stationary queue) during green time, measured in field or estimated from arrival type as $(P_g = R_p \cdot (g/c))$,

R_p = approximate ranges of platoon ratio based on progression quality (level of coordination between junctions that defines the arrival type), available in [Bureau of Public Roads, 1950], page 16-20,

f_{PA} = supplemental adjustment factor for platoon arriving during green, also available in [Bureau of Public Roads, 1950], page 16-20,

g = effective green time, and

c = cycle length.

Alternatives for the PF approach were provided in [Benekohal and El-Zohairy, 2001], where an arrival-based (AB) approach also considers the quality of progression but includes it into a specific uniform delay model for each arrival type. Additionally, [TARKO et al., 1993] confirms that the progression quality has no effect on the overflow delay.

We have just discussed how random arrivals can be implemented in non-random arrivals scenarios using the progression factor, as well as how to deal with inflows that change over time (see section B.2.2). However, we still need to present a possibility to estimate the arrival flow profile given upstream outflow. The model proposed in ([Robertson, 1969] apud [Farzaneh, 2005]) (implemented in the TRANSYT simulation and optimization model) is the most well-known model of platoon dispersion, it uses a histogram of the outflow coming from the upstream junction to obtain the pattern of arrival on the downstream junction:

$$Q_k(t) = SF_k^t \cdot Q_{k-1}(t - TT_k^{min}) + (1 - SF_k^t) \cdot Q_k(t - \Delta t), \tag{B.48}$$

where

k = downstream edge,

$(k-1)$ = upstream edge,

t = time step (integer),

Δt = duration of time step t (sec), corresponding to the length of the discrete interval of the arrival histogram,

$Q_{k-1}(t)$ = outflow from edge $(k-1)$ at time t (veh/Δt),

$Q_k(t)$ = inflow to edge k at time t (veh/Δt),

$TT_k^{min} = (\gamma_t \cdot TT_k^{ave})/\Delta t$ = minimum travel time on edge k (time step unit, multiple of Δt),

TT_k^{ave} = average travel time on edge k (sec),

SF_k^t = smoothing factor on edge k at time step t (sec), and

γ_t = travel time factor at time step t (dimensionless).

The arrival profile at the downstream junction at instant t is as a linear combination of the downstream flow one time step earlier ($Q_k(t-\Delta t)$) with the upstream departure flow TT_k^{min} steps earlier ($Q_{k-1}(t-TT_k^{min})$). This characteristic means that the arrival flow follows a shifted geometric distribution, and using its properties the parameters (factors) can be estimated as [Mathew, 2017b]:

$$\gamma_t = \frac{1}{1+\beta_t} = \frac{2TT_k^{ave} + \Delta t - \sqrt{(\Delta t)^2 + 4(\sigma_{TT}^t)^2}}{2TT_k^{ave}}, \tag{B.49}$$

$$SF_k^t = \frac{1}{1+\beta_t \cdot \gamma_t \cdot TT_k^{ave}} = \Delta t \left(\frac{\sqrt{(\Delta t)^2 + 4(\sigma_{TT}^t)^2} - \Delta t}{2(\sigma_{TT}^t)^2} \right), \tag{B.50}$$

where

σ_{TT}^t is the standard deviation of edge travel times at time step t on edge k, and

β_t = platoon dispersion factor at time step t (dimensionless), typically from 0.25 (tight platoons of suburban high-speed arterials) to 0.5 (dispersed platoons typical of central areas) [Geroliminis and Skabardonis, 2005], but also computed by $\beta_t = \frac{1-\gamma_t}{\gamma_t}$.

The model assumes binomial distribution of vehicle travel time as well as arrivals are not influenced by an initial queue downstream, which implies it works well only in undersaturated conditions. Another possibility is introduced in [Pacey, 1956], where the platoon dispersion can be modelled from the differences in speed between vehicles in the platoon, leading to normally distributed travel times used to transform upstream flow into the traffic flow profile ([Farzaneh, 2005],[FHWA, 2018]):

$$f(TT_k) = \left(\frac{L_k}{(TT_k)^2 \cdot \sigma_v \cdot \sqrt{2\pi}} \right) \cdot \exp \left[- \frac{\left(\frac{L_k}{TT_k} - \frac{L_k}{TT_k^{ave}} \right)^2}{2(\sigma_v)^2} \right], \tag{B.51}$$

$$Q_k(\tau_1 \Delta t) = \sum_{\tau_0 \Delta t} Q_{k-1}(\tau_0 \Delta t) \cdot f(\tau_1 \Delta t - \tau_0 \Delta t), \tag{B.52}$$

where

L_k = downstream edge k distance (meters),

TT_k = individual vehicle travel time along distance L_k (duration flow observation unit, multiple of Δt),

TT_k^{ave} = travel time corresponding to the average speed on edge k (duration flow observation unit, multiple of Δt),

σ_v = standard deviation of speed on edge k,

$Q_k(\tau_1 \Delta t)$ = number of vehicles arriving downstream (edge k) during time interval $\tau_1 \Delta t$ (veh/Δt),

$Q_{k-1}(\tau_0 \Delta t)$ = number of vehicles leaving upstream (edge $k-1$) during time interval $\tau_0 \Delta t$ (veh/Δt),

Δt = duration of flow observation (between each time interval), and

$f(\tau_1 \Delta t - \tau_0 \Delta t)$ = probability of the travel time is $(\tau_1 \Delta t - \tau_0 \Delta t)$.

The (discrete) Δt long intervals of the arrival histogram, τ_0 and τ_1, represent the first and second observations, respectively. For instance, the flow in the τ_1-th interval observed downstream is the summation of the flows for all τ_0-th previous intervals observed upstream, and then multiplied by the probability of the travel time is $(t_{\tau_1} - t_{\tau_0})$. One issue with this model is that it assumes vehicles travel at a constant speed and it is necessary to know the speeds of vehicles without any interference between vehicles. Additionally, these last two platoon dispersion/diffusion models needs the departure profile, being the common practice using constant discharge flows based on the saturation flow, start loss and end gain time ([FHWA, 2018], [Geroliminis and Skabardonis, 2005]). An alternative for constant discharge flow is introduced in [Akçelik and Besley, 2002], where queue discharge models consider the queue discharge speed to derive relationships for other traffic parameters.

B.3 The Influence of Route Choice

When every vehicle with same O-D (Origin-Destination) pair get local travel times and choose their optimal route without considering others path choice (and hence congestion due capacity), this is known as *all-or-nothing* path choice principle. As congestion effects the travel time, the choices made by other travellers also trying to choose the least travel time (or disutility), may impact the travel time [Srinivas Peeta et al., 2015]. The two Wardrop's principles introduced in [Wardrop, 1952] states two fundamental path decision principles in the following list.

1. First principle (*User Equilibrium*), "*The journey times of all routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route*". This corresponds to the Nash Equilibrium, in which travellers modify their path choice until they unilaterally minimize their travel cost. The problem with this principle is that it assumes that all travellers are homogeneous, fully rational, and they have perfect knowledge of travel costs [Srinivas Peeta et al., 2015].

2. Second principle (*System Optimum*): "*At equilibrium the average journey time is minimum*". In other words, travellers cooperate with another to minimize total system travel time. This principle may not be consistent because the selfish behaviour of travellers seeking to minimize their individual path travel times [Srinivas Peeta et al., 2015].

Later, [Daganzo, 1977] suggested to modify the first principle as it follows:

1. Modification o First principle (*Stochastic User Equilibrium*), "*no user believes he can improve his travel time by unilaterally changes*". This assumes that each driver has a different perception of costs on any given route and that the trips between each O-D pair are divided among the routes with the most cheapest route attracting most trips.

Cooperative Automated Vehicles (CAVs) are assumed to aim to reduce their travel time but not achieving User Equilibrium principle. The process to incorporates the travel demand and supply by assigning trips through the network, and edge (path) flows with its corresponding edge (path) travel times is called *traffic assignment*. In fact, this term mainly used on the representation of average or steady-state conditions of a time period long enough to allow all traffic flows to arrive at their destinations, also known as *static traffic assignment* models. The issue with these models are that they don't account variations of travel times and flows over the analysis period, such as oversaturated traffic flow, queue spill-back, dynamic routing [Srinivas Peeta et al., 2015]. In reality, the signal control strategy affects the travel time on the roads and influences the drivers' route choice behaviour, what also contribute to define signal timings, illustrated in Figure B.6 [Chen

and Hu, 2012]. Another class of traffic assignment models was first first introduced in [Merchant and Nemhauser, 1978], the so called *dynamic traffic assignment* (DTA) models. These models try to capture the dynamism and realism of traffic, in which each DTA has its underlying behavioural and system assumptions as well as control actions. However, among their common features are time-dependent O-D matrices, and the time-dependent flow-density relationship [Ziliaskopoulos, 2001]. Additionally, analytical models have been developed under the time-dependent flow conservation and propagation, boundary and non-negativity constraints. The Dynamic User Equilibrium (DUE) uses edge travel times (edge performance) models, such as point-queue model, whole edge model, and cell transmission model. Another type are simulation-based DTA models, which avoid challenges of analytical models.

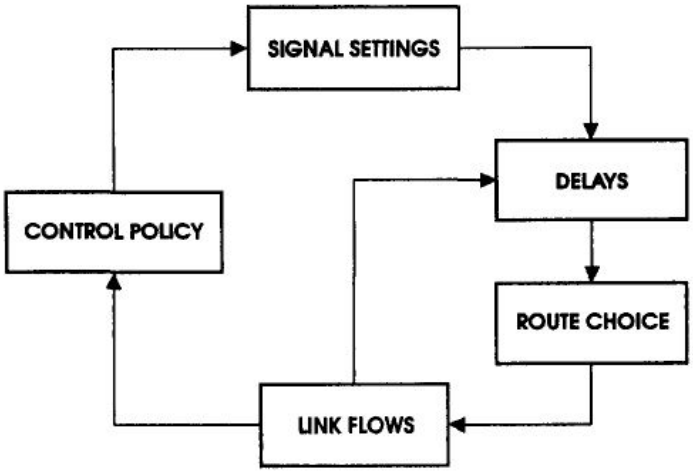


Figure B.6: Conceptual structure of combined traffic assignment and control problem [Meneguzzer, 1997].

Although there are evidences that drivers may make decisions based on minimum perceived travel time differences (thresholds) [Watling and van Vuren, 1993], the emulation of driver's decision of selecting from one of the available paths can be done with a Route Choice model. These models are usually based on discrete choice theory, they determine the probability of choosing an alternative as function of its disutility. The utility function U_{r_a} can be represented in many forms. For instance, average route travel time on the route r_a between the O-D pair (s, d) [Yang et al., 2006], or a function that includes preference of the driver, and "pressure" on the next edge of the route and on the route itself [Kampen, 2015]. Under stochastic traffic assignment and utility maximization, a rational traveller will choose the path with the minimum perceived generalized cost [Srinivas Peeta et al., 2015]:

$$U_{r_a} = \overline{u}_{r_a} + \varepsilon_{r_a}, \tag{B.53}$$

where U_{r_a} represents the perceived route travel cost (utility), $\overline{u}_{r_a}$ the

deterministic (or measured) travel time, and ε_{r_a} the random error on the route r_a. There are mainly two distribution for the random error, multinomial normal distribution (but the final model is not in an analytical closed form) and Gumbel distribution [Srinivas Peeta et al., 2015]. Using the latter and assuming travel times as the travel costs, and as we need to use the disutility, $\overline{u}_{r_a} = -TT_{r_a}$, we can assume the utility function U_{r_a} as [Ben-Akiva and Bierlaire, 1999]:

$$U_{r_a} = \theta \cdot (-TT_{r_a}) + \varepsilon_{r_a} = -\theta \cdot TT_{r_a} + \varepsilon_{r_a}, \tag{B.54}$$

where θ is a positive shape or scale factor parameter, TT_{r_a} the expected travel time on route r_a, and ε_{r_a} the random term. From the Multinomial Logit model:

$$Var(TT_{r_a}) = \frac{\pi^2}{6 \cdot \theta^2}, \tag{B.55}$$

that means if $(\theta < 1)$ drivers have high perception of the variance of travel times (notice that is variation between each route and not variation of the random error), and $(\theta > 1)$ drivers may choose an specific set of alternative routes.

The most common discrete choice models for travel behaviour modelling are the Multinomial Logit (MNL) and Nested Logit (NL). However, MNL does not assume similarity (correlation) among alternatives (routes are independent), while in NL each route alternative belongs exclusively to one nest, but in real networks routes share many different other paths. Therefore, modifications of such models are proposed and classified in [Prato, 2009] as it is presented in the following list.

- *Logit* structures, they are modifications of MNL models but introducing a correction term within the deterministic part of the utility function account the correlation among alternative routes.
- *Generalized Extreme Value* (GEV) structures, which allow similarities in the stochastic part of the utility function, and relate the network topology to the specific coefficients, but do not consider taste variation or correlation over time of unobserved factors.
- *Non-GEV* structures, where unrestricted substitution patterns, random taste variation, and correlation in unobserved factors over time are allowed, closed-form expression for the choice probabilities are not not present.

As there are many proposed models, we will only explore simple models that have satisfactory performance and they are suitable for our local level routing system. The first and simplest model is the proportional decision rule, in which the frequency of a selected route is used to estimate the probability, what for us would mean turning rates.

The C-Logit model [Cascetta et al., 1996] uses a commonality factor, CF_{r_a}, represents the degree of similarity of each route (path) r_a with the other paths r_c in the route choice set from source s to destination d, $R_{s,d}$, where subscripts a means the analysing route and c a choice route. The probability P_{r_a} of choosing route r_a within the choice set R is the simple Logit structure of the model [Ben-Akiva and Bierlaire, 1999]:

$$P_{r_a} = \frac{\exp\left[\theta(\overline{u}_{r_a} - CF_{r_a})\right]}{\sum_{c \in R_{s,d}} \exp\left[\theta(\overline{u}_{r_c} - CF_{r_c})\right]} \quad \forall\ r_a \in R_{s,d}, \tag{B.56}$$

where U_{r_a} and U_{r_c} are the perceived utility functions of route r_a and r_c, respectively, and CF_{r_c} the commonality factor for r_c. This commonality factor has many formulations in the literature. However, as cited in [Prato, 2009], the following expression computed expected and satisfactory results above the others:

$$CF_{r_a} = \beta_{CF} \cdot \ln\left[1 + \sum_{\substack{r_c \in R_{s,d} \\ r_a \neq r_c}} \left(\frac{L_{r_a,r_c}}{\sqrt{L_{r_a} L_{r_c}}}\right)\left(\frac{L_{r_a} - L_{r_a,r_c}}{L_{r_c} - L_{r_a,r_c}}\right)\right], \tag{B.57}$$

in which L_{r_a} and L_{r_c} are the length of routes r_a and r_c, respectively, L_{r_a,r_c} the common length between routes r_a and r_c, and β_{CF} an estimated parameter.

Generally, the Path-Size Logit model [Ben-Akiva and Bierlaire, 1999] outperforms the C-Logit model [Prato, 2009]. It models the expression of the probability of choosing route r_a within the alternative routes in a simple Logit structure [Prato, 2009]:

$$P_{r_a} = \frac{\exp\left\{\theta\left[\overline{u}_{r_a} + \beta_{PS} \cdot \ln(PS_{r_a})\right]\right\}}{\sum_{c \in R_{s,d}} \exp\left\{\theta\left[\overline{u}_{r_c} + \beta_{PS} \cdot \ln(PS_{r_c})\right]\right\}} \quad \forall\ r_a \in R_{s,d}, \tag{B.58}$$

where PS_{r_a} and PS_{r_c} are the path sizes of routes r_a and r_c, respectively. Here, there are two also alternatives for computation of PS_{r_a}, the original and generalized form, and β_{PS} is a parameter to be estimated. We will present the original because the generalized needs an additional parameter to be estimated, and even though it can produce better results, it may also produce counter-intuitive results ([Prato, 2009], [Ben-Akiva and Bierlaire, 1999]). The original form is:

$$PS_{r_a} = \sum_{k \in \Gamma_{r_a}} \frac{L_k}{L_{r_a}} \cdot \frac{1}{\sum_{r_c \in R_{s,d}} \delta_{k,r_c}}, \tag{B.59}$$

in which L_k is the length of edge k, Γ_{r_a} is the set of edges belonging to route r_a, and δ_{k,r_c} is the edge-path incidence matrix (equal to one if route r_c uses edges k and zero otherwise).

It is important to notice that each of those models gives a different interpretation regarding the correction term (CF or PS) of the utility function. The commonality factor (CF) reduces the utility of a path due its similarity with other routes, while the path size (PS) corresponds to the fraction of the path that constitutes a "full" alternative [Prato, 2009]. Additionally, those factors are needed to be calculated only once. The θ parameter adjusts the effect that small changes in the travel times may have on the driver's decisions. This parameter got best-fit value of 10 in [Kampen, 2015], while in [Yang et al., 2006], the authors defined their values using uniform distribution and set fixed values of standard deviation of travel time perception. [Prato, 2009] states that β_{CF} and β_{PS} should be negative to express the reduction of the utility of paths with common edges with respect to other routes. Finally, the flow calculation from source junction s to destination d on each route r_a is

$$Q_{r_a} = Q \cdot P_{r_a} \quad \forall\ r_a \in R_{s,d}, \tag{B.60}$$

where Q is the inflow at the junction and Q_{r_a} is the flow through the route r_a. The outflow from the junction to each downstream junction could be based on the sum of the flows for each r_a with same first edge.

Appendix C

Simulation and Results

C.1 Results

The first table can be found on the next page.

Scenario	Avg. Different Edges [%]						Avg. Diff. Route [%]
	All	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5	
LLR 10PR	39.0±0.6	21.2±3.1	30.5±5.3	17.2±5.1	17.7±3.7	18.3±4.7	85.9±0.6
LLR 20PR	44.0±0.6	24.8±5.5	33.2±5.9	24.9±5.6	23.5±5.2	20.6±4.0	90.5±0.7
LLR 30PR	45.5±0.5	26.0±2.9	37.6±5.3	25.4±3.9	18.3±4.2	28.5±5.6	92.8±0.4
LLR 50PR	47.1±0.6	29.1±4.1	43.4±4.9	29.9±4.4	22.7±4.5	33.3±6.8	94.2±0.3
LLR 70PR	48.3±0.5	32.7±3.9	44.4±6.6	31.8±6.1	37.2±4.4	42.5±6.8	95.0±0.2
LLR 85PR	48.7±0.5	34.4±2.6	43.3±6.1	33.0±4.9	36.4±3.3	46.4±4.9	95.1±0.3
LLR 100PR	49.2±0.6	36.2±4.5	47.1±6.1	28.1±6.8	34.7±6.8	41.0±3.1	95.3±0.2

Table C.1: Confidence intervals of the average percentage of different edges and routes between the LLR routed vehicles' final route compared to their Hourly Travel Times (HTT) routes with same Penetration Rate (PR).

Scenario	Sum Waited Time [s]		Sum Mean Travel Time [s]		Mean Speed [m/s]	
	Ave.	% Diff.	Ave.	% Diff.	Ave.	% Diff.
DUA	495,131.3±0.0	-	3,261.3±0.0	-	08.2±0.0	-
LLR 10PR	459,218.2±1311.4	-2.2%±0.4%	3,199.5±13.2	0.2%±0.4%	08.3±0.0	-0.4%±0.3%
HTT 10PR	469,726.1±1336.7	-	3,194.4±8.5	-	08.3±0.0	-
LLR 20PR	422,185.4±1229.7	-4.7%±0.3%	3,153.2±5.4	0.3%±0.3%	08.4±0.0	-0.1%±0.3%
HTT 20PR	442,930.1±2171.9	-	3,142.9±8.6	-	08.4±0.0	-
LLR 30PR	388,603.7±1335.7	-6.4%±0.5%	3,126.9±4.6	0.3%±0.3%	08.5±0.0	-0.2%±0.3%
HTT 30PR	415,379.7±1735.7	-	3,118.8±5.7	-	08.5±0.0	-
LLR 50PR	325,057.8±2684.8	-12.8%±0.4%	3,075.3±12.4	-0.4%±0.5%	08.5±0.0	0.3%±0.2%
HTT 50PR	372,730.7±1965.9	-	3,088.8±8.3	-	08.5±0.0	-
LLR 70PR	272,761.4±3517.1	-18.9%±1.0%	3,066.1±9.2	-0.9%±0.5%	08.5±0.0	0.2%±0.3%
HTT 70PR	336,390.5±1944.6	-	3,094.8±8.0	-	08.5±0.0	-
LLR 85PR	244,381.6±4717.6	-20.5%±1.1%	3,094.9±42.1	0.4%±1.4%	08.5±0.0	0.0%±0.4%
HTT 85PR	307,482.3±2950.8	-	3,081.7±8.4	-	08.5±0.0	-
LLR 100PR	218,528.1±3851.3	-25.1%±1.0%	3,126.2±28.5	0.4%±1.2%	08.4±0.0	0.0%±0.5%
HTT 100PR	291,949.1±3563.6	-	3,114.5±17.2	-	08.4±0.0	-
DUE	157,866.1±0.0	-	3,088.7±0.0	-	08.5±0.0	-

Table C.2: Confidence intervals of the average and percentage difference of sum of waited time, mean travel time, and mean speed on the edges controlled by the LLR system per scenario. The percentage difference values are compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).

Scenario	Mean Density [veh/km]		Mean Occupancy [%]		Max Mean Occupancy [%]	
	Ave.	**% Diff.**	**Ave.**	**% Diff.**	Ave.	**% Diff.**
DUA	08.8±0.0	-	01.5±0.0	-	19.0±0.0	-
LLR 10PR	08.2±0.0	-1.5%±0.5%	01.4±0.0	-1.7%±0.5%	17.4±0.2	-1.0%±1.7%
HTT 10PR	08.3±0.0	-	01.5±0.0	-	17.6±0.2	-
LLR 20PR	07.6±0.0	-4.4%±0.3%	01.4±0.0	-4.4%±0.5%	16.7±0.2	-2.7%±1.3%
HTT 20PR	07.9±0.0	-	01.4±0.0	-	17.2±0.2	-
LLR 30PR	07.1±0.0	-6.5%±0.4%	01.3±0.0	-6.4%±0.4%	15.7±0.2	-1.5%±1.5%
HTT 30PR	07.6±0.0	-	01.4±0.0	-	15.9±0.2	-
LLR 50PR	06.1±0.1	-12.5%±0.4%	01.2±0.0	-11.9%±0.6%	13.0±0.1	-8.4%±1.7%
HTT 50PR	07.0±0.0	-	01.3±0.0	-	14.1±0.2	-
LLR 70PR	05.3±0.1	-18.0%±1.0%	01.1±0.0	-16.7%±1.3%	09.8±0.2	-19.5%±2.5%
HTT 70PR	06.5±0.0	-	01.3±0.0	-	12.1±0.3	-
LLR 85PR	04.9±0.1	-20.0%±1.0%	01.0±0.0	-17.8%±1.1%	07.8±0.1	-24.0%±1.8%
HTT 85PR	06.1±0.1	-	01.2±0.0	-	10.3±0.2	-
LLR 100PR	04.5±0.1	-23.3%±1.0%	01.0±0.0	-20.5%±1.0%	07.6±0.5	-19.3%±4.0%
HTT 100PR	05.9±0.1	-	01.2±0.0	-	09.4±0.4	-
DUE	03.0±0.0	-	00.6±0.0	-	05.2±0.0	-

Table C.3: Confidence intervals of the average and percentage difference mean density, occupancy and maximum mean occupancy on the edges controlled by the LLR system per scenario. The percentage difference values are compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).

Scenario	Mean Route Duration [s]					
	All	**Probe 1**	**Probe 2**	**Probe 3**	**Probe 4**	**Probe 5**
LLR 10PR	236.1±4.4	215.4±4.0	347.1±9.4	230.1±15.0	183.0±4.3	204.9±9.9
HTT 10PR	244.0±5.5	226.1±8.7	357.4±13.4	230.3±12.1	199.5±7.7	206.7±8.0
LLR 20PR	241.7±4.4	213.2±6.4	360.4±10.5	227.5±16.0	194.1±8.3	213.1±11.4
HTT 20PR	241.3±7.3	216.5±7.3	351.9±11.2	234.8±16.4	202.9±11.8	200.1±7.2
LLR 30PR	248.5±6.5	212.9±13.5	356.2±11.1	249.5±25.8	199.4±11.3	224.5±13.6
HTT 30PR	252.5±9.4	227.8±10.2	362.8±12.4	246.9±17.5	189.7±7.0	235.4±20.5
LLR 50PR	250.2±8.5	235.4±15.8	356.5±15.1	263.1±25.1	188.3±9.4	208.0±7.9
HTT 50PR	253.8±8.7	228.2±11.0	369.9±13.0	249.7±19.7	195.2±10.2	226.1±13.1
LLR 70PR	245.7±7.4	233.3±10.0	355.9±8.4	240.1±21.5	194.4±10.7	205.0±13.0
HTT 70PR	254.5±7.0	241.6±12.9	365.8±18.8	248.0±19.2	209.1±14.1	208.0±20.3
LLR 85PR	249.8±9.6	244.2±15.3	354.1±15.8	262.7±30.1	195.3±8.6	192.6±12.7
HTT 85PR	267.1±6.7	246.0±15.3	370.3±13.6	275.9±18.1	214.0±10.0	229.1±11.9
LLR 100PR	247.9±10.2	235.2±12.7	356.5±15.0	242.1±24.1	198.1±6.9	207.8±11.8
HTT 100PR	254.5±6.7	236.4±12.0	367.8±18.1	249.3±20.8	202.1±13.4	216.9±12.9

Table C.4: Confidence intervals of the average mean route duration of all LLR routed vehicles and per probe vehicle.

Scenario	Percentage Difference Mean Route Duration [%]					
	All	**Probe 1**	**Probe 2**	**Probe 3**	**Probe 4**	**Probe 5**
LLR 10PR	-3.2%±2.5%	-4.5%±3.8%	-2.7%±2.9%	0.7%±9.3%	-8.0%±3.8%	-0.5%±7.3%
LLR 20PR	0.3%±3.0%	-1.5%±2.4%	2.6%±4.3%	-2.2%±9.5%	-3.6%±7.3%	6.7%±6.0%
LLR 30PR	-1.3%±4.4%	-6.4%±5.4%	-1.5%±5.5%	2.2%±12.8%	5.3%±6.4%	-3.7%±7.1%
LLR 50PR	-1.1%±5.3%	3.4%±7.1%	-3.3%±5.8%	7.3%±15.2%	-2.9%±7.6%	-7.5%±5.6%
LLR 70PR	-3.2%±4.7%	-3.0%±5.6%	-2.1%±5.9%	-2.3%±10.2%	-6.2%±7.8%	0.5%±12.0%
LLR 85PR	-6.3%±4.9%	-0.1%±7.4%	-4.0%±6.4%	-4.0%±12.5%	-8.1%±7.2%	-15.2%±8.8%
LLR 100PR	-2.5%±4.3%	0.1%±7.8%	-2.6%±6.5%	-2.0%±11.0%	-1.0%±8.2%	-3.5%±7.9%

Table C.5: Confidence intervals of the percentage difference of mean route duration of all LLR routed vehicles and per probe vehicle, compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).

Scenario	Mean Route Length [m]					
	All	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5
LLR 10PR	1,942.1±31.9	1,789.0±8.4	2,844.0±67.6	2,021.3±117.8	1,499.9±39.8	1,556.4±40.6
HTT 10PR	1,978.3±31.8	1,865.5±39.4	2,910.3±64.5	1,950.4±83.8	1,589.7±53.7	1,575.7±43.0
LLR 20PR	1,964.9±32.9	1,804.0±28.0	2,895.1±66.2	1,966.6±109.6	1,560.5±69.2	1,598.2±49.6
HTT 20PR	1,985.9±46.4	1,845.7±28.0	2,857.7±56.7	1,980.1±113.9	1,666.6±93.3	1,579.4±36.2
LLR 30PR	2,009.3±43.3	1,790.0±15.1	2,831.6±53.7	2,141.8±169.7	1,607.2±86.5	1,676.0±64.6
HTT 30PR	2,030.9±67.8	1,831.5±20.7	2,876.1±58.1	2,117.1±128.4	1,591.6±69.5	1,738.2±147.2
LLR 50PR	1,982.7±45.9	1,856.3±66.2	2,818.9±72.2	2,098.7±147.4	1,530.0±43.1	1,609.6±30.8
HTT 50PR	2,021.0±65.4	1,856.8±50.8	2,890.8±86.1	2,054.2±163.4	1,586.0±75.3	1,717.2±103.2
LLR 70PR	1,989.3±48.1	1,835.7±52.6	2,871.5±73.1	2,007.8±158.8	1,597.0±78.5	1,634.3±72.1
HTT 70PR	2,028.6±41.4	1,840.4±38.8	2,882.5±117.5	1,994.0±139.7	1,738.5±84.8	1,687.7±123.0
LLR 85PR	2,001.7±58.4	1,837.6±45.9	2,835.9±97.1	2,142.9±216.0	1,595.4±58.5	1,596.6±61.3
HTT 85PR	2,095.4±56.3	1,869.1±63.7	2,901.3±82.0	2,191.1±176.2	1,717.3±65.7	1,798.1±103.6
LLR 100PR	1,975.3±66.9	1,798.4±35.1	2,851.8±83.1	1,998.3±192.7	1,573.1±67.4	1,654.6±62.2
HTT 100PR	1,988.9±56.6	1,816.3±40.6	2,833.5±106.5	1,986.3±152.9	1,602.5±74.8	1,705.6±99.3

Table C.6: Confidence intervals of the average mean route length of all LLR routed vehicles and per probe vehicle.

Scenario	Percentage Difference Mean Route Length [%]					
	All	**Probe 1**	**Probe 2**	**Probe 3**	**Probe 4**	**Probe 5**
LLR 10PR	-1.8%±2.3%	-4.0%±2.1%	-2.2%±2.6%	4.2%±8.4%	-5.5%±2.6%	-1.1%±3.7%
LLR 20PR	-0.9%±2.9%	-2.2%±2.4%	1.4%±3.5%	0.0%±7.9%	-5.7%±7.3%	1.3%±3.4%
LLR 30PR	-0.9%±3.7%	-2.3%±1.1%	-1.4%±3.1%	2.1%±10.7%	1.3%±6.6%	-2.6%±6.1%
LLR 50PR	-1.6%±4.5%	0.2%±4.9%	-2.3%±4.1%	3.6%±11.0%	-3.1%±5.6%	-5.7%±5.0%
LLR 70PR	-1.8%±3.6%	-0.1%±3.8%	-0.1%±3.9%	1.5%±9.3%	-7.9%±4.3%	-2.4%±6.3%
LLR 85PR	-4.3%±3.6%	-1.4%±4.4%	-2.0%±5.2%	-1.1%±11.9%	-6.7%±6.0%	-10.5%±7.0%
LLR 100PR	-0.5%±4.4%	-0.8%±3.8%	0.9%±4.0%	1.8%±12.3%	-1.4%±6.4%	-2.3%±6.9%

Table C.7: Confidence intervals of the percentage difference of mean route length of all LLR routed vehicles and per probe vehicle, compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).

Scenario	Mean Waiting Time [s]					
	All	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5
LLR 10PR	71.2±2.3	64.8±3.1	107.7±5.5	63.2±6.6	54.8±3.1	65.8±6.3
HTT 10PR	76.3±3.4	68.9±6.3	111.2±8.4	69.9±5.7	64.9±4.3	66.3±5.4
LLR 20PR	74.8±2.2	60.9±4.7	115.5±5.6	65.7±8.0	61.0±4.5	70.7±7.0
HTT 20PR	73.3±3.9	60.9±6.2	109.3±6.5	71.7±7.7	62.9±4.6	61.6±4.9
LLR 30PR	77.3±3.6	61.8±10.3	114.7±6.9	72.2±10.6	63.1±5.3	74.6±7.8
HTT 30PR	79.1±3.9	70.8±7.6	117.1±7.8	70.7±6.8	55.8±4.2	81.1±8.6
LLR 50PR	79.6±4.7	73.9±10.2	115.5±9.8	84.2±12.1	58.2±6.0	66.3±6.5
HTT 50PR	80.8±4.5	69.3±8.0	121.2±7.9	78.3±7.6	59.8±4.4	75.3±7.2
LLR 70PR	75.8±3.7	74.8±7.2	110.9±5.8	72.8±9.3	58.1±5.3	62.7±10.7
HTT 70PR	79.9±4.6	80.3±9.5	118.7±9.1	80.0±8.0	59.5±8.2	61.2±10.4
LLR 85PR	78.5±5.3	82.5±10.5	112.0±9.1	82.5±12.3	59.6±6.0	56.1±8.0
HTT 85PR	85.2±3.1	81.7±10.7	119.6±6.7	88.8±8.3	64.4±6.4	71.7±5.9
LLR 100PR	78.3±4.2	77.0±10.4	112.7±9.9	73.6±12.4	63.1±3.6	64.9±8.0
HTT 100PR	81.9±3.2	76.6±8.2	121.6±8.4	80.7±8.5	62.8±7.1	67.8±8.8

Table C.8: Confidence intervals of the average mean waiting time of all LLR routed vehicles and per probe vehicle.

Scenario	Percentage Difference Mean Waiting Time [%]					
	All	**Probe 1**	**Probe 2**	**Probe 3**	**Probe 4**	**Probe 5**
LLR 10PR	-6.3%±4.6%	-4.4%±10.2%	-2.3%±6.8%	-8.4%±12.3%	-14.6%±8.9%	1.0%±14.7%
LLR 20PR	2.5%±5.3%	1.1%±7.1%	6.3%±7.2%	-6.0%±15.9%	-1.8%±11.0%	16.4%±15.3%
LLR 30PR	-1.8%±6.4%	-12.2%±12.1%	-0.9%±9.8%	4.2%±18.3%	14.2%±11.1%	-6.2%±11.8%
LLR 50PR	-0.7%±8.3%	8.7%±17.2%	-3.9%±10.1%	11.9%±24.3%	-1.2%±13.9%	-10.9%±8.9%
LLR 70PR	-4.2%±8.8%	-5.1%±11.0%	-4.9%±11.2%	-7.5%±14.0%	3.1%±21.2%	11.4%±32.8%
LLR 85PR	-7.4%±8.3%	3.5%±15.3%	-5.6%±10.0%	-5.9%±15.4%	-5.5%±12.6%	-20.3%±13.8%
LLR 100PR	-4.2%±5.9%	2.9%±17.1%	-6.0%±12.4%	-7.0%±18.4%	4.6%±18.8%	0.5%±21.8%

Table C.9: Confidence intervals of the percentage difference of mean waiting time of all LLR routed vehicles and per probe vehicle, compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).

Scenario	Avg. Rerouteing Times [#]					
	All	Probe 1	Probe 2	Probe 3	Probe 4	Probe 5
LLR 10PR	1.1±0.1	1.1±0.1	1.4±0.1	1.1±0.1	1.0±0.1	0.9±0.1
HTT 10PR	1.1±0.0	1.1±0.1	1.5±0.1	1.1±0.1	1.0±0.0	1.0±0.1
LLR 20PR	1.1±0.1	1.1±0.1	1.6±0.1	1.1±0.1	1.0±0.0	1.0±0.0
HTT 20PR	1.1±0.1	1.1±0.1	1.5±0.1	1.1±0.1	0.9±0.0	1.0±0.0
LLR 30PR	1.2±0.0	1.1±0.1	1.5±0.1	1.2±0.2	1.0±0.0	1.1±0.1
HTT 30PR	1.2±0.0	1.2±0.1	1.5±0.1	1.1±0.1	0.9±0.0	1.1±0.1
LLR 50PR	1.2±0.0	1.2±0.1	1.5±0.1	1.3±0.1	1.0±0.0	1.0±0.1
HTT 50PR	1.1±0.1	1.1±0.1	1.5±0.1	1.1±0.2	0.9±0.1	1.1±0.1
LLR 70PR	1.1±0.0	1.1±0.1	1.5±0.1	1.2±0.1	1.1±0.1	1.0±0.1
HTT 70PR	1.1±0.1	1.2±0.1	1.4±0.2	1.0±0.1	1.0±0.1	1.0±0.1
LLR 85PR	1.1±0.1	1.2±0.1	1.4±0.1	1.1±0.2	1.0±0.1	0.9±0.1
HTT 85PR	1.2±0.0	1.2±0.1	1.5±0.1	1.1±0.1	1.0±0.1	1.1±0.1
LLR 100PR	1.1±0.1	1.1±0.1	1.5±0.1	1.1±0.2	1.0±0.1	1.0±0.1
HTT 100PR	1.1±0.1	1.1±0.1	1.5±0.2	1.0±0.1	1.0±0.1	1.1±0.1

Table C.10: Confidence intervals of the average mean rerouteing times of all LLR routed vehicles and per probe vehicle.

Scenario	Percentage Difference Avg. Rerouteing Times [%]					
	All	**Probe 1**	**Probe 2**	**Probe 3**	**Probe 4**	**Probe 5**
LLR 10PR	-0.2%±4.8%	2.6%±7.3%	-5.9%±8.3%	4.8%±13.9%	3.7%±6.3%	-0.2%±8.2%
LLR 20PR	2.8%±2.5%	4.3%±8.5%	1.5%±5.1%	3.0%±10.5%	6.7%±6.3%	1.4%±7.3%
LLR 30PR	1.3%±4.1%	-9.7%±6.5%	-0.6%±6.1%	10.3%±18.3%	8.3%±6.2%	3.1%±8.8%
LLR 50PR	5.6%±6.7%	5.8%±7.5%	1.3%±12.1%	26.9%±22.7%	9.7%±6.8%	-5.5%±10.7%
LLR 70PR	2.8%±6.4%	-6.1%±10.4%	6.6%±13.2%	13.0%±14.5%	6.7%±10.6%	2.7%±12.1%
LLR 85PR	-3.3%±5.9%	-1.1%±5.2%	-0.2%±10.6%	3.9%±17.3%	2.4%±12.0%	-16.3%±12.5%
LLR 100PR	-0.4%±7.1%	-2.9%±12.0%	3.0%±13.2%	8.8%±14.8%	0.0%±10.7%	-5.0%±10.0%

Table C.11: Confidence intervals of the percentage difference of mean rerouteing times of all LLR routed vehicles and per probe vehicle, compared to their respective Hourly Travel Times (HTT) scenarios with same Penetration Rate (PR).