FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Dynamic Traffic Assignment and Routing Algorithms with Applications in Smart Mobility

Diogo Miguel Ferreira Rodrigues

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Rosaldo José Fernandes Rossetti

July 28, 2023

Dynamic Traffic Assignment and Routing Algorithms with Applications in Smart Mobility

Diogo Miguel Ferreira Rodrigues

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Ana Paula Rocha
Referee: Prof. Marin Lujak (Rey Juan Carlos University)

July 28, 2023

This work is a result of project DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions, with reference NORTE-01-0145-FEDER-000073, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Cofinanciado por:

Abstract

Transportation forecasting is the area concerned with analyzing, modeling, simulating and validating mobility models for people in a constructed environment, typically in an urban setting where the most pressure is exerted on the transportation infrastructure. The main problem that transportation forecasting aims to solve is one of optimization, where the goal is minimizing infrastructure spending while maximizing the global utility, reduce road congestion, and minimize time spent in transit, upkeep and fuel costs of vehicles, or carbon emissions. It is concerned with crowd dynamics as well as with social, economical and cultural factors that impact users' decisions concerning means of transportation and possible routes.

Traffic assignment consists of allocating a route to each trip (pair of origin-destination), and is the basis of all investigation into transportation forecasting. The system-optimal traffic assignment is one that minimizes the global cost function, and is only feasible if there is a single decision center with perfect knowledge. However, each individual typically adopts a greedy strategy by choosing the route that minimizes their own cost function. This strategy reduces the amount of data and reasoning required by each user for effective decision-making. If this behavior is applied by every user, a sub-optimal user equilibrium is reached where there is no faster route other than the selected route for a given trip.

The main concern of this work is the difference between system-optimal and user equilibrium, namely to study the phenomenon, and to propose new cost functions and new ways for inter-user interaction so as to change user decisions, with the ultimate goal of meeting certain system-wide objectives.

The suggested changes to user behavior are relevant in the study of global optimization of urban traffic, and contribute to better addressing urgent needs of contemporary societies such as environmental sustainability, productivity, fuel- and energy-saving, while still considering the importance of social justice in a system where people make concessions for the greater good but are still entitled to a fair share of resources.

The employed methods comprised the development of a high-performance mesoscopic traffic simulator that served as a simulated environment to study the way people move and different strategies for route assignment. The results of this study point towards better global performance metrics when assignment strategies stray away from greedy behavior, and that the developed methodology and technology constitute significant contributions for researchers and decision-makers to analyze the impact of solutions or decisions, so as to allow a better insight into how people move and better data-driven public policy and decision making.

Keywords: Operations. Traffic assignment. Dynamic traffic assignment. Optimization. Mesoscopic simulation. Reinforcement Learning. Difference reward.

Resumo

A área científica de previsão de tráfego refere-se à análise, modelação, simulação e validação de modelos de transporte de pessoas num ambiente edificado, tipicamente em zonas urbanas onde existe mais pressão sobre a infraestrutura de transporte. O problema principal da previsão de tráfego constitui-se como um problema de otimização, onde o objetivo é minimizar gastos em infraestrutura enquanto se maximiza a sua utilidade global, reduz a congestão nas estradas, e minimiza o tempo perdido em trânsito, os custos de manutenção dos veículos e de combustíveis, ou emissões de gases com efeito de estufa. Os conceitos essenciais desta área são a dinâmica de populações, assim como os fatores sociais, económicos e culturais que afetam as decisões dos utilizadores relativamente a meios de transporte e rotas alternativas.

O problema de afetação de tráfego consiste na atribuição de uma rota a cada viagem (par origem-destino), e constitui a base da previsão de tráfego. A afetação de tráfego que minimiza a função global de custo designa-se *ótima*, e só é possível com um único centro de decisão com conhecimento perfeito. No entanto, cada indivíduo tipicamente adota uma estratégia *greedy* (gananciosa) ao escolher a rota que minimiza a sua própria função de custo. Esta estratégia reduz a quantidade de informação e reflexão necessárias para uma tomada de decisão eficaz. Se este comportamento for adotado por todos os utilizadores, resulta num equilíbrio do utilizador subótimo, onde não existe nenhuma rota mais rápida do que a rota selecionada para uma certa viagem.

O tema principal deste trabalho é a diferença entre o ótimo do sistema e o equilíbrio do utilizador, nomeadamente o estudo do fenómeno, e a proposta de novas funções de custo e novas formas de interação entre utilizadores de forma a alterar as suas decisões, com o objetivo último de atingir certos objetivos globais do sistema.

As alterações sugeridas ao comportamento dos utilizadores são relevantes no estudo da otimização global do tráfego urbano, e contribuem para a satisfação de necessidades urgentes das sociedades contemporâneas como a sustentabilidade ambiental, produtividade e eficiência energética, considerando, no entanto, a importância da justiça social, num sistema onde as pessoas fazem concessões em prol do bem comum, mas têm ainda assim direito a uma utilização justa dos recursos.

Os métodos aplicados incluem o desenvolvimento de um simulador de tráfego mesoscópico de alto desempenho computacional, que serviu como ambiente de simulação para estudar o movimento dos veículos e diferentes estratégias para atribuição de rotas. Os resultados deste estudo apontam para melhores métricas globais de performance quando as estratégias de afetação se afastam do comportamento *greedy*, sendo que a metodologia e tecnologia desenvolvidas constituem contribuições importantes para os investigadores e decisores públicos analisarem o impacto de soluções ou decisões específicas, permitindo uma melhor compreensão da forma como as pessoas se movem, e contribuindo para melhores políticas públicas e tomada de decisão guiada por dados.

Palavras-chave: Investigação Operacional. Afetação de tráfego. Afetação de tráfego dinâmica. Otimização. Simulação mesoscópica. *Reinforcement Learning*. *Difference reward*

Acknowledgments

To Professor Rosaldo Rossetti for his essential support and contribution to this work as my thesis supervisor through his vast knowledge of traffic simulation and assignment, as well as his enthusiasm and expertise as a lecturer of modules CAL (Algorithm Design and Analysis) and EDAA (Advanced Data Structures and Algorithms), the latter of which was especially focused on traffic problems, which ultimately led me to accept Professor Rossetti's challenge to join project DynamiCITY and to elaborate this Master Thesis under his supervision.

I am also grateful for the opportunity to participate in the DynamiCITY Final Workshop, organized in Porto on the 27th of June 2023, with the presentation of this work abstract, whose extended version will be published as a book chapter on the DynamiCITY Special Springer Book Edition.

To the members of the project "DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions", with reference NORTE-01-0145-FEDER-000073, under which this work was elaborated, as well as to Norte Portugal Regional Operational Programme (NORTE 2020) for financial support to the project under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

To my parents and brother, for their unconditional support throughout my academic pathway, and insightful discussions.

To the municipality of Porto (Câmara Municipal do Porto) for providing project DynamiCITY with their traffic demand estimates, which this work greatly benefited from.

To the OpenStreetMap contributors (2023) (of which I am also a member as a side-effect of this work), for the extensive and up-to-date map data that made this work possible. OpenStreetMap depends largely on voluntary work to curate its immense dataset. It finds use in many commercial, research, and public policy projects, which goes to show how powerful and significant free, open collaboration data is in modern times.

Diogo Rodrigues

"If you wanted to discuss economics,
you shouldn't have invited an economist.
You should have invited a psychiatrist."

Unknown author, on a TV debate
about the Portuguese economy,
2021-2022

Contents

1 Introduction **1**
- 1.1 Problem and aims . . . 2
- 1.2 Hypotheses . . . 3
- 1.3 Research opportunities . . . 3
- 1.4 Thesis outline . . . 4

I State of the art **5**

2 Traffic assignment **7**
- 2.1 Traffic assignment as an economic problem . . . 7
- 2.2 Vehicle dynamics . . . 9
- 2.3 Modeling . . . 10
  - 2.3.1 Capacity and speed models . . . 11
  - 2.3.2 Capacity and travel time estimation . . . 15
  - 2.3.3 Signalized intersections . . . 16
  - 2.3.4 Gap acceptance model . . . 18

3 Static traffic assignment **21**
- 3.1 Formalization . . . 21
- 3.2 All-or-nothing assignment . . . 22
- 3.3 Non-uniqueness of path flows . . . 23
- 3.4 Equilibrium conditions . . . 23
  - 3.4.1 System optimum . . . 23
  - 3.4.2 User equilibrium . . . 23
  - 3.4.3 Comparison between SO and UE . . . 25
  - 3.4.4 Price of anarchy and Braess's paradox . . . 26
- 3.5 As a convex optimization problem . . . 27
- 3.6 Frank-Wolfe algorithm . . . 28
  - 3.6.1 Frank-Wolfe algorithm applied to traffic assignment . . . 29
  - 3.6.2 Stopping criteria . . . 30

4 Traffic simulation models **33**
- 4.1 Network description . . . 33
- 4.2 Level of detail . . . 34
  - 4.2.1 Macroscopic models . . . 34
  - 4.2.2 Microscopic models . . . 34
  - 4.2.3 Mesoscopic models . . . 35

4.3 Time-driven vs event-driven . . . 35
4.4 Models of interest . . . 36
4.4.1 SUMO . . . 36
4.4.2 DynaMIT . . . 38

5 Assignment strategies **41**
5.1 Tolls . . . 41
5.2 Auctions and negotiation . . . 43
5.3 Reinforcement learning . . . 43
5.3.1 Q-learning . . . 44
5.3.2 Difference reward . . . 46

II Methodology **49**

6 Data **51**
6.1 Traffic network data . . . 51
6.2 Demand and TAZ data . . . 53

7 Static simulator **55**
7.1 Intersections . . . 55
7.2 Signalized intersections . . . 55
7.3 Unsignalized intersections . . . 56

8 Dynamic simulator **57**
8.1 Parameters . . . 58
8.2 Environment . . . 58
8.2.1 Uncongested flow . . . 59
8.2.2 Queues . . . 59
8.2.3 Prioritized intersections . . . 60
8.2.4 Lane changing . . . 61
8.3 Events . . . 62
8.4 Vehicle spawning . . . 64
8.5 Environment-policy interaction . . . 65
8.6 Deployment and visualization . . . 65
8.7 Parallelism . . . 68

9 Assignment strategies **69**
9.1 Q-learning . . . 69
9.1.1 Parameter estimation . . . 70
9.1.2 Metrics . . . 70
9.2 Reward function . . . 71

10 Results & Discussion **73**
10.1 Performance comparison . . . 73
10.2 Assignment strategies . . . 74

11 Conclusions **79**

References **81**

A Source code **89**

B Gap acceptance model **91**

C DynaMIT supply simulation **95**

D Parameter estimation **97**
D.1 Vehicle gap 97
D.2 Queue dissipation speed 99

E Movement equations under uncongested flow **101**
E.1 On driver behavior in the face of speed limits 101
E.2 The movement model 104
E.2.1 Lane changing 106

F Order-statistic queue **107**

List of Figures

2.1 Definition of spacing and headway in a traffic time-space diagram. Adapted from Luttinen (1996, fig. 2.1). 9
2.2 Fundamental diagram of traffic flow 14
2.3 Link performance functions according to different models 15

3.1 Braess's network, demonstrating Braess's paradox. Adapted from Braess (1968). 26

4.1 Simplified classification of traffic models by level of detail. 34
4.2 Position at time t in the presence of a queue, as used by DynaMIT 40

6.1 Data retrieval and processing pipeline. 52
6.2 Traffic network of the municipality of Porto with only relevant road classes . . . 53
6.3 Exploratory visualization of demand data from 09:00-10:00. 54

8.1 Data flow diagram of DynamiNATOR 57
8.2 Queue spill-back when destination lane is full 61
8.3 Event diagram for the dynamic simulator 63
8.4 Screen captures of a netstate file rendered for same region (VCI between N14 and A3). 66
8.5 Screen capture of the traffic network of Porto, rendered with DynamiNATOR GUI. 66
8.6 Detailed screen captures of two intersections, rendered with DynamiNATOR GUI. 67
8.7 Four frames of the same intersection at different time instants in DynamiNATOR. 67

10.1 Number of vehicles and average travel time of vehicles leaving the simulation. . . 74
10.2 Diagrams demonstrating how a vehicle may become stuck in a loop due to Q-value underestimation. 76
10.3 Travel time for agents using difference demand with distinct values of greediness factor w. 76

D.1 Vehicle length and gap measurements in the city of Porto. 98

E.1 Scenario of highway and offramp with progressively lower speed limits. 103
E.2 Three scenarios where different speed limit layouts affect vehicle speed profiles . 103

List of Tables

3.1 Equations to convert SO and UE cost functions 25

10.1 Execution times of simulators SUMO and DynamiNATOR for similar scenarios . 73
10.2 Scenario to evaluate different assignment strategies. 74
10.3 Experiments with different greediness factors 75
10.4 Final travel times in experiments with different values of greediness factor w. . . 77

D.1 Gap length measurements' results. 97

List of Listings

F.1 Order-statistic queue 108

Abbreviations

API	Application Programming Interface
BPR	Bureau of Public Roads
CBD	central business district
CDF	cumulative distribution function
DP	Dynamic Programming
DTA	dynamic traffic assignment
EU	European Union
EU27	European Union 27 member states as of 2023
EU28	European Union 28 member states as of 2013, including the United Kindgom
FFS	free-flow speed
FIFO	First-In, First-Out
FW	Frank-Wolfe algorithm
GDP	Gross Domestic Product
GUI	Graphical User Interface
HCM	Highway Capacity Manual
ITS	Intelligent Transport Systems
OD	origin-destination
OSM	OpenStreetMap
PDF	probability distribution function
RL	reinforcement learning
SARSA	State-Action-Reward-State-Action
SI	International System of Units
SimAAS	Simulation-as-a-Service
SO	system optimum
STA	static traffic assignment
SUMO	Simulation of Urban MObility
TA	traffic assignment
TAP	traffic assignment problem
TAZ	traffic assignment zone
TRB	Transportation Research Board
UE	user equilibrium
UN	United Nations

Chapter 1

Introduction

Since the 1950s, the world has gone through a process of rapid urbanization. In 1950 the urbanization rate in Europe was only 51.7% (UN, 2018, 2019), which has risen to 74.5% by 2018 (UN, 2019), and is estimated to reach 83.7% in 2050 (UN, 2018, 2019). This trend is common to the entire world, with urban population accounting for 29.6% of the world population in 1950, 53.9% in 2015, and 68.4% by 2050 (UN, 2018), and is particularly expressive in Asia and Africa, which are expected to contribute with 90% of the 2.5 billion increase in urban population until 2050 (UN, 2019). Car ownership is also steadily increasing in the EU27, at a yearly rate of 1.2%, totaling 246.3 million passenger cars (ACEA, 2022) for a population of 446.7 million in 2022 (Eurostat, 2023). These two trends contribute to a high and ever-increasing traffic congestion burden urban residents incur. The European Commission estimates that total passenger congestion costs amount to 0.98% of the EU28 GDP, or nearly $140.0\,\text{B€}/\text{year}$; in the case of Portugal, this share is 1.37%, or $2.3\,\text{B€}/\text{year}$ at the time of the report (Brannigan et al., 2017, pp. 21-22). And congestion costs are only expected to continue increasing in the EU, by about 50% by 2050 to nearly $200\,\text{B€}/\text{year}$ (European Comission, 2011, p. 18). It is, therefore, paramount to find ever-smarter solutions so that people can live harmoniously in dense communities, as it is only expected that cities will grow in size and population, accentuating congestion problems.

The way people move in a traffic network is often guided by greedy principles, where each user selects the most convenient route. This is the result of applying the concept of Nash equilibrium (Nash, 1951) to the traffic assignment problem to arrive at an economic equilibrium. There are some advantages associated with this strategy, namely not requiring communication with fellow users (Speranza, 2018; Morandi, 2023). Because it is greedy, it is also intrinsically fair as it allows each user to have a distinct way of evaluating possible paths, and every user is equally entitled to make selfish decisions (Morandi, 2023; Lujak et al., 2015). The equilibrium situation in which everyone is greedy is called the user equilibrium, for no user may improve their individual outcomes by unilaterally changing paths, and was initially formalized by Wardrop (1952).

This situation differs from the system optimum (Wardrop, 1952), sometimes called the *city manager* optimum (Tumer et al., 2008), where a global utility function is defined and the system is optimized using that function. This global utility function can be defined by the infrastructure

manager, a city authority, or implicitly defined by society as a whole, so as to attain relevant societal goals such as better infrastructure usage through reduced congestion costs, pollution, and infrastructure maintenance (Morandi, 2021).

A consequence of the definition of system optimum is that any other solution is suboptimal. In fact, Klein et al. (2018, as cited by Morandi, 2021) assessed that the system optimum solution is optimal but 'unstable', due to the inherent unfairness (Lujak et al., 2015) of some users making concessions for the greater good, and that any of those users' move towards an individually better path would inevitably stray from the system optimum. This phenomenon also occurs in the *prisoner's dilemma*, where two accomplices would rather betray each other than cooperate towards the globally optimal outcome, hinting at the inherent fact that users' selfish unilateral choices lead to a sub-optimal situation that constitutes an opportunity cost to society. This is also called the *price of anarchy*, which can be quantified as the difference between the user equilibrium and the system optimum. As pointed out by Lujak et al. (2015), user equilibrium can be arbitrarily more costly than system optimum, and some studies have noted that this price of anarchy can typically vary between 0.5% in low-load scenarios and 15% in high-load scenarios (Peeta and Mahmassani, 1995).

Given there is a difference between the current situation (which most often resembles user equilibrium) and the system optimum, there is a margin for improvement. Thus the relevance of this problem is connected to the fact that so many people live in cities that even small improvements in each driver's experience can lead to a large overall benefit in terms of, for instance:

- Reduced fuel consumption and time wasted in traffic, leading to greater economic rationality and productivity;
- Less greenhouse and toxic gas emissions in urban canyons with poor air ventilation;
- Less congestion-induced stress, leading to better living standards.

These issues are directly related to the United Nations Sustainable Development Goals (United Nations, 2015), namely to goal 3 ("ensure healthy lives and promote well-being for all at all ages"), goal 8 ("promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all"), goal 12 ("ensure sustainable consumption and production patterns"), and most importantly goal 11 ("make cities and human settlements inclusive, safe, resilient and sustainable").

1.1 Problem and aims

The main issue this work tackles is the gap between the user equilibrium and system optimum scenarios, since drivers' greedy behaviors negatively affect the overall system.

This overarching issue branches into three problems. Firstly, there are no specific large-scale, efficient models or studies that simulate or predict traffic conditions in the city of Porto since there is no city-wide model or simulator tuned for the city that is both designed for performance, accurate but without unjustified levels of detail. Secondly, most traffic congestion studies either

apply static traffic assignment or simple dynamic traffic assignment approaches, whereas it is known that users of a traffic network do not have unbounded knowledge and rationality, contrary to what is commonly assumed. The final question is how different ways of modeling user behavior contribute to global system metrics.

Also, accurate traffic and congestion models are essential for studying a traffic network, as they allow for simulating congestion conditions for well-informed public policy and accurate travel time estimates in travel planning.

1.2 Hypotheses

The hypotheses that we pose in this dissertation are twofold. The first is that reinforcement learning techniques, namely using Q-learning with distinct reward functions based on the concept of difference reward (Tumer and Wolpert, 2004), can provide better traffic assignment solutions, with reduced global cost values. Secondly, to perform those experiments we need a new, mesoscopic high-performance traffic simulator over which we have full control, that is accurate enough to represent the real-world process to the necessary degree, but simple enough that its formalization is analytically tractable so the necessary estimates required by difference reward methods can be computed.

1.3 Research opportunities

Mentioned below are the scientific gaps that this work aims to bridge.

1. The objective function is the single criterion for optimality. This function usually represents travel time. All externalities that constitute inconveniences for the user have to be explicitly translated to time; some have direct conversions—converting tolls to time by knowing the average hourly salary of people and using that as the *value* of people's time—, others do not—some people may prefer slower paths but with the advantages of having more comfortable road geometry, or because they are not as agile (as is the case of senior citizens) and therefore prefer less frequented roads or routes with fewer traffic lights and roundabouts. This makes conversion cumbersome and does not allow new variables to be included in the metrics used in the decision process.

2. The small size of datasets used by other authors allows them to use microscopic simulators easily, but a city-wide simulation with individual agents can only be achieved in a performant way if a mesoscopic simulator is used.

3. Authors of assignment strategies (Rossetti et al., 2002; Rossetti and Liu, 2005a,b; Schepperle and Böhm, 2007; Tumer and Agogino, 2007; Schepperle et al., 2008; Tumer et al., 2008; Grunitzki et al., 2014; Monteiro et al., 2014; Lujak et al., 2015) typically validate their findings with small networks; this work aims to apply assignment strategies to real scenarios, to evaluate the impacts of those strategies in a large city.

1.4 Thesis outline

This work is organized into two parts. Part I presents the state of the art; it is a four-chapter literature review, providing background on the traffic assignment problem (Chap. 2), an overview of classical methods for solving the problem (Chap. 3), some relevant specific models from literature (Chap. 4), and a review of some of the most used assignment strategies (Chap. 5).

The core of this dissertation is part II, where a methodology is proposed to solve the problems at hand. The developed data pipeline is described (Chap. 6), followed by a simple static simulator (Chap. 7), and then the two most important chapters: the dynamic traffic assignment model (Chap. 8) and the traffic assignment strategies (Chap. 9).

Finally, results (Chap. 10) and conclusions (Chap. 11) are presented that support the contributions of this work.

Part I

State of the art

Chapter 2

Traffic assignment

This chapter introduces the area of traffic assignment (TA) and lays out a set of fundamental concepts that are common to all traffic assignment analysis and models. It serves as a theoretical and general background to the specific solutions presented in Chap. 3 and Chap. 4. Throughout this document, right-hand traffic is assumed.

Given a description of a traffic network and the trips that users in the traffic network want to perform, encoded as origin-destination (OD) pairs, traffic assignment is the problem of allocating a route to each user such that certain criteria are met. If the goal is descriptive, the criterion is the accuracy of the model when compared to real-world data and empirical evidence. This type of model is mostly used to evaluate or predict congestion conditions and assist users in making decisions by providing accurate information. If the goal is prescriptive, the criterion is an objective function that encodes a specified objective. This is the case if the objective is to influence how users choose routes in a traffic network so that a certain global objective function is optimized (Sloboden et al., 2012, p. 3-6).

2.1 Traffic assignment as an economic problem

Traffic assignment was initially regarded as an economic problem before progressively being adopted by and moving into the fields of statistics and operations research, and later into simulation. In fact, the first observations reporting to this field were made by economists Artur Pigou (1920, p. 194) and Frank Knight (1924) by remarking that roads are negatively affected by congestion, that "the congestion and interference resulting from the addition of any particular truck to the stream of traffic on [a road] affects in the same way the cost and output of all the trucks using that road", and that "whenever there is a difference in the cost, to an additional truck, of using the two roads, the driver of any truck has an incentive to use the [less congested road], until the advantage is reduced to zero for all the trucks" (Knight, 1924), a hint at the concept of Wardrop equilibrium (Wardrop, 1952), now known as user equilibrium.

This analogy of the traffic assignment problem (TAP) as an economic problem is further elaborated by Sheffi (1985, pp. 4-10) and Boyles et al. (2019, pp. 6-11), which posit that:

- Supply is the traffic network, which is composed of a finite amount of infrastructure, and each road can only serve a finite amount of traffic before it becomes heavily congested. The resources in this system are the allowed vehicle flows through the roads.

- Demand is the OD pairs that encode trips that the users want to perform; those trips must be served by the resources of the system.

Supply and demand interact when a user selects a route to meet their will to perform a trip. By doing so, the user is *acquiring* a portion of flow from each road that is part of the route; therefore, in this scenario, the resource can also be restated as being a *slot* of flow in a specific road.

One of the fundamental assumptions about the supply is that the *cost* it takes to cross a road is dependent on how many vehicles are currently on that road (see Eq. (2.3) for a formal definition). Cost is typically assumed to be the time it takes to cross a road, but it may also be any other cost function. This is empirically obvious from the fact a user selects a speed to travel on a road depending on the number of expected interactions with other vehicles (approximately proportional to the number of vehicles on that road). If a road is almost empty, users may select the maximum legally allowed speed since they do not expect to interact with other users. If a road has some vehicles, the user has to take caution to avoid possible accidents, and therefore the user has to drive slower to make it easier to come to a full stop in the event another user does a dangerous maneuver. If a road is heavily congested, all vehicles come to a near-full stop, and a queue starts to form; the queue density is the maximum density of vehicles on that road, since all vehicles are near-fully stopped, so concerns for accidents are reduced, and vehicles can queue up as close as possible. According to Klein et al. (2018), this is a reflex of applying the concept of *tragedy of the commons*[1] to vehicular traffic, in the form of a *social mobility dilemma* where "too many cars on too little road space, [end] up in clogged traffic". It may therefore be asserted that the *flow through a road* is a scarce resource, and as such, its cost increases as the available flow through that road becomes increasingly scarce. When a new user chooses a route, the cost of routes never decreases, indicating that users are competing for this scarce resource.

Some formulations of the TAP problem extend this model by considering elastic demand (Beckmann et al., 1956) where users may choose to perform a trip or not depending on the cost of the trips, or by considering user knowledge as *stochastic* where user knowledge is bounded, each piece of knowledge is associated to an error or noise factor, and user decisions are not limited to the best option (Sheffi, 1985, pp. 261-344).

[1] The *tragedy of the commons* is a metaphor that conveys the concept that individuals with unlimited access to a common finite resource tend to over-use it, destroying its value. In the formulation by Hardin (1968), the individuals are herders, and the common resource is a pasture.

2.2 Vehicle dynamics

This short section provides a brief insight into how people drive, how vehicles physically move and interact in a road network, and how that should be taken into account in traffic models.

Consider two adjacent vehicles $i-1$ and i on a road. The *spacing* between the two vehicles at time instant τ is the length (in meters) that separates the positions $x_{i-1}(\tau)$ and $x_i(\tau)$ of the two adjacent vehicles at time instant τ. The *headway* $t_i(\tau)$ between the two vehicles at time instant τ is the time it takes for vehicle i to get to the position $x_{i-1}(\tau)$ that vehicle $i-1$ was at time τ; it is, therefore, a *time-distance* (Sheffi, 1985, p. 347; Luttinen, 1996, pp. 21-22). These concepts can be visualized in Fig. 2.1, which shows a traffic time-space diagram, where time is represented in the horizontal axis and vehicle positions in the vertical axis, and the trajectories of vehicles are represented in the time-space coordinate system as they traverse a road, with the position of each vehicle at each time instant.

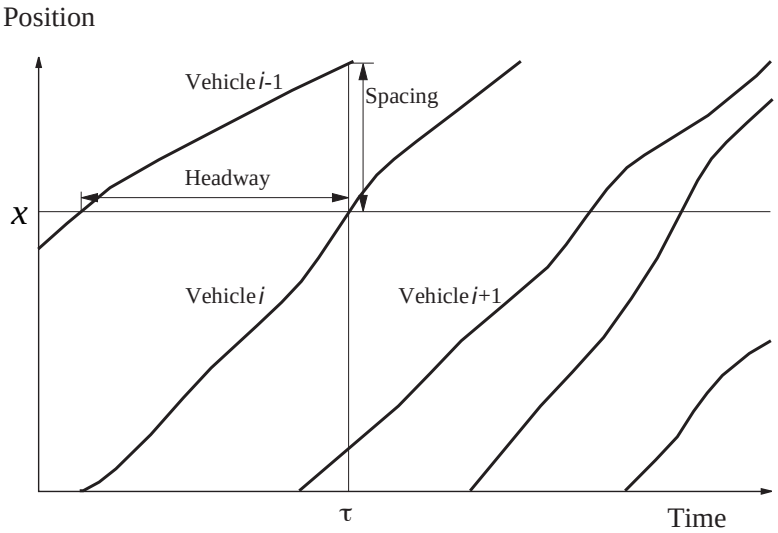


Figure 2.1: Definition of spacing and headway in a traffic time-space diagram. Adapted from Luttinen (1996, fig. 2.1).

A well-known concept in traffic engineering is that of a *movement*, extensively used in the Highway Capacity Manual (HCM) (Transportation Research Board, 2000). This concept is akin to a *connection* in traffic simulator SUMO (Simulation of Urban MObility) (Krajzewicz et al., 2002), and is of fundamental importance since it encodes the concept of which transitions between roads and lanes are allowed at junctions. It also provides a typification of similar actions in many junctions into a common designation, which the HCM (TRB, 2000) leverages to provide a general methodology for analyzing junctions (e.g., it is relevant to study and measure the time needed for a left-turning vehicle to cross oncoming traffic in the presence of a dedicated left-turning lane).

A traffic queue is a situation where vehicles are stopped or nearly stopped on a road. It can occur at a red traffic light or at a prioritized intersection, where the queue grows backwards; or it can occur on any road when there is an excessive amount of vehicles attempting to traverse

that road, in which case a breakdown in traffic flow occurs, and the queue typically propagates backwards as more vehicles arrive at the congested section, but the front of the queue is already slowly dissipating.

Queue spill-back occurs when a queue takes up all the available road length, so incoming vehicles start queuing up on upstream roads that lead up to the original queue (Ben-Akiva et al., 2002). A gridlock is a scenario where multiple continuous queues spill back onto each other and block discharges of different junctions, causing all traffic to stop in a certain area (TRB, 2000, p. 35; Boyles et al., 2019, p. 16). This scenario often occurs when vehicles enter a junction without checking for enough time or space to leave the junction, therefore blocking the junction for other users. It can also occur as an artifact in certain traffic simulators.

Vehicles frequently change lanes (also called lateral movements) for a variety of reasons. The literature (Yang and Koutsopoulos, 1996, p. 123-124; Liu, 2010; Ferrara et al., 2018, p. 123; Kessels, 2019, p. 47) makes a distinction between two main types of lane changes: (1) mandatory lane change, where a driver moves to a different lane because of their route choice, where staying in the same lane would not lead to the next intended edge; (2) discretionary lane change, where a driver changes lanes for any other reason, often to increase throughput, avoid merging vehicles or seek a speed advantage by changing to a lane where traffic is moving faster.

2.3 Modeling

The traffic network can be formalized as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$. Nodes $\mathcal{N}$ are physical locations with a definite position and serve as interchanges between edges; they usually represent road intersections. Arcs $\mathcal{A}$ are pairs of nodes, which denote a road connecting two intersections. We denote N as the size of $\mathcal{N}$, and A as the size of $\mathcal{A}$. Nodes $n \in \mathcal{N}$ are uniquely identified by sequential integers starting at zero, such that $n \in \mathbb{Z}_0^+$ and $n < N$. Each arc $a(u, v) \in \mathcal{A}$ is denoted as a pair of nodes $(u, v) \in \mathcal{N}^2$.

Since OD data is generally estimated from census data and other similar methods with limited granularity, trip origins and destinations are grouped into traffic assignment zones (TAZs), which are sets of nodes or edges where vehicles start or end their trips. Although the scale of TAZs highly depends on available data, it is commonly defined as a region aggregating populations with common geographic, demographic, and socioeconomic characteristics (Martínez et al., 2009), containing a set of census blocks with around 3000 people (Miller and Shaw, 2001).

By analogy with thermodynamics, it is possible to use macroscopic state variables to study vehicles' macroscopic and microscopic behavior on a road. The most important state variables in traffic networks are:

- Flow q, the average number of vehicles passing a fixed point in the road per unit of time. Measured in vehicles/second.
- Density K, the average number of vehicles present at some instant on a section of the road of unit length. Measured in vehicles/meter.

- Space mean speed $\overline{v}_s$, the average speed of vehicles along a road segment at some instant, as defined by Sheffi (1985, p. 347).

The fundamental relation between these state variables is

$$q = \overline{v}_s K \tag{2.1}$$

Let the flow of vehicles crossing an arc a at a certain instant t be x_a, and the cost for a vehicle crossing that same arc assuming the vehicle entered arc a at instant t to be t_a. As stated before, the greater the flow through an arc, the more time vehicles take to cross it due to an increase in inter-vehicular interactions. This phenomenon is captured by the *link performance function* $t_a(x_a)$, which relates the travel time on an arc to the flow traversing that arc (Sheffi, 1985, p. 16).

As stated by Sheffi (1985, p. 61), the assumptions on the link performance curves can be written mathematically as

$$\frac{\partial t_a}{\partial x_b} = 0 \qquad \forall a \neq b \tag{2.2}$$

$$\frac{\partial t_a}{\partial x_a} \geq 0 \qquad \forall a \tag{2.3}$$

Assumption 2.3 is the reason why users of a traffic network are considered to be competing since an increase of users choosing a particular road to cross causes its available flow to become more scarce, and as such, its cost increases accordingly. Therefore, user choices have only negative impacts on other users' outcomes: adding another user to the network will never lead to an improvement in global congestion conditions.

The most commonly considered parameters for link performance functions are the road's *length* L, *capacity* c and *free-flow speed (FFS)* v_0 (or alternatively *free-flow time* $t_0 = L/v_0$). Road length is the length of the road from the intersection where it starts to the intersection where it ends, and it can be calculated from the geometry of the road. Capacity is the maximum vehicle flow that can be transported by a road, expressed in vehicles per unit time; the related term *saturation flow rate* is the "rate at which previously queued vehicles can traverse an intersection approach [...] assuming that the green signal is available at all times and no lost times are experienced" (TRB, 2000, p. 5-14). FFS is the average speed of vehicles over a road without signalized intersections under conditions of low flow, so in practice, it translates to the average speed drivers select for a road when only considering its geometry, functional classification, and speed limit.

2.3.1 Capacity and speed models

There are many models for the speed and flow of vehicles in a road, so we will present some of the simplest (as a theoretical background) and some of the most used.

Greenshields model The first and simplest model was proposed by Greenshields (1935):

$$v = v_0 \left(1 - \frac{K}{K_{\text{jam}}}\right) \tag{2.4}$$

$$t = t_0 \left(\frac{K_{\text{jam}}}{K_{\text{jam}} - K}\right) \tag{2.5}$$

where v_0 and t_0 are the free-flow speed and time respectively, and K_{jam} is the jam density at road a. Greenshields' formula is not very used nowadays, in favor of other formulas.

BPR model One of the most well-known and used models was introduced by the U.S. Bureau of Public Roads (1964, p. V-20):

$$t = t_0 \left[1 + \alpha \left(\frac{q}{c'}\right)^{\beta}\right] \tag{2.6}$$

$$v = v_0 \frac{c'^{\beta}}{c'^{\beta} + \alpha q^{\beta}} \tag{2.7}$$

where t is the travel time, t_0 is the free-flow travel time, q is the flow through the road, c' is the "practical capacity" (Sheffi, 1985, p. 375), which is different from the capacity of the road, and α and β are dimensionless, empirically-determined parameters. BPR (1964, p. V-20) provides the reference values for the parameters $\alpha = 0.15$ and $\beta = 4$, which are still used nowadays.

Note, however, that Eq. (2.6) allows the flow to increase indefinitely, even if at a high cost. If we consider the scenario where the flow x is much greater than the road capacity, by Eq. (2.6) we can assume the vehicle will take a long time to cross that road, and therefore will be traveling at a very slow speed; in this scenario, due to Eq. (2.1), we would arrive at a very high density K, even larger than K_{jam}, which is impossible since we assumed K_{jam} is the highest possible density, since it is the density of a jam (or queue).

Davidson model To address this issue, Davidson (1966) proposed the formula

$$t = t_0 \left(1 + J \frac{q}{c - q}\right) \tag{2.8}$$

$$v = v_0 \frac{c - q}{c - (1 - J)q} \tag{2.9}$$

where J is an empirically-determined parameter, which Davidson (1966) experimentally verified to be around 0.04 and 0.20. This formula can provide a more realistic model since it does not allow a flow greater than the road capacity c. However, this also requires capacity to be enforced as a hard constraint in the form of a maximum flow to guarantee feasibility of solutions, which can slow down the solving process. Hence, the BPR formula is used most times so that road capacity does not have to be enforced as a hard constraint.

Model comparison A tool that is often used to study the flow as a function of density is the *fundamental diagram* of traffic flow (Sheffi, 1985, p. 350; TRB, 2000, p. 7-5; Boyles et al., 2019, p. 376; Kessels, 2019, p. 23), Fig. 2.2, given that these two state variables are related through Eq. (2.1). It plots the flow against density. For the Greenshields model (Eq. (2.4)), the flow is

$$q = v_0 \left(K - \frac{K^2}{K_{\text{jam}}} \right) \tag{2.10}$$

The maximum flow according to the Greenshields model is

$$q_{\text{max}} = q|_{K=K_{\text{jam}}/2} = v_0 \frac{K_{\text{jam}}}{4} \tag{2.11}$$

The flow under the BPR model (Eq. (2.7)) does not have a closed form, as it implies inverting

$$K = \frac{1}{v_0} \left(\frac{\alpha}{c'^{\beta}} q^{\beta} + q \right) \tag{2.12}$$

but we can still plot it by replacing q with successive values.

Sheffi (1985, p. 375) points out that, for $\alpha = 0.15$, c' is the flow at which the travel time is 15% higher than the free-flow travel time. Therefore, to find a value of c' that can be used to compare the BPR model with other models, it must be asserted that the flows using the BPR and another model (e.g., Greenshields model) are the same if the following condition is true:

$$t = t_0(\alpha + 1) \tag{2.13}$$

Using Eq. (2.5), the condition of Eq. (2.13) is equivalent to

$$\frac{K_{\text{jam}}}{K_{\text{jam}} - K} = 1 + \alpha \iff K = \alpha K_{\text{jam}} \tag{2.14}$$

At this point, the flow according to the Greenshields model is

$$c' = q = v_0 \left(K - \frac{K^2}{K_{\text{jam}}} \right) = v_0 K_{\text{jam}} (\alpha - \alpha^2) \tag{2.15}$$

Eq. (2.15) can be restated as c' being smaller than the maximum capacity (as per the Greenshields model) by a factor of $(\alpha - \alpha^2)/4$.

For the Davidson model (Eq. (2.8)), the flow is

$$q = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \tag{2.16}$$

$$A = J - 1 \tag{2.17}$$

$$B = c + K v_0 \tag{2.18}$$

$$C = -K v_0 c \tag{2.19}$$

As can be seen from Fig. 2.2, the BPR model allows unlimited flow, which is unrealistic, but does not require special treatment when formalizing the TAP. On the other hand, the Greenshields and Davidson models have the advantage that they do not allow unlimited flow and are therefore more realistic, but by doing so, they require special treatment to avoid unfeasible solutions, namely by adding hard constraints to bound flow. Also, according to Sheffi (1985, p. 359), using models that limit flow "can slow the algorithm considerably, with no appreciable change in accuracy", so for practical implementations, the BPR model is better. Besides, although the BPR model allows arbitrarily large flows, it still penalizes that choice heavily by applying a soft constraint over large flows, since cost grows rapidly when $q > c'$.

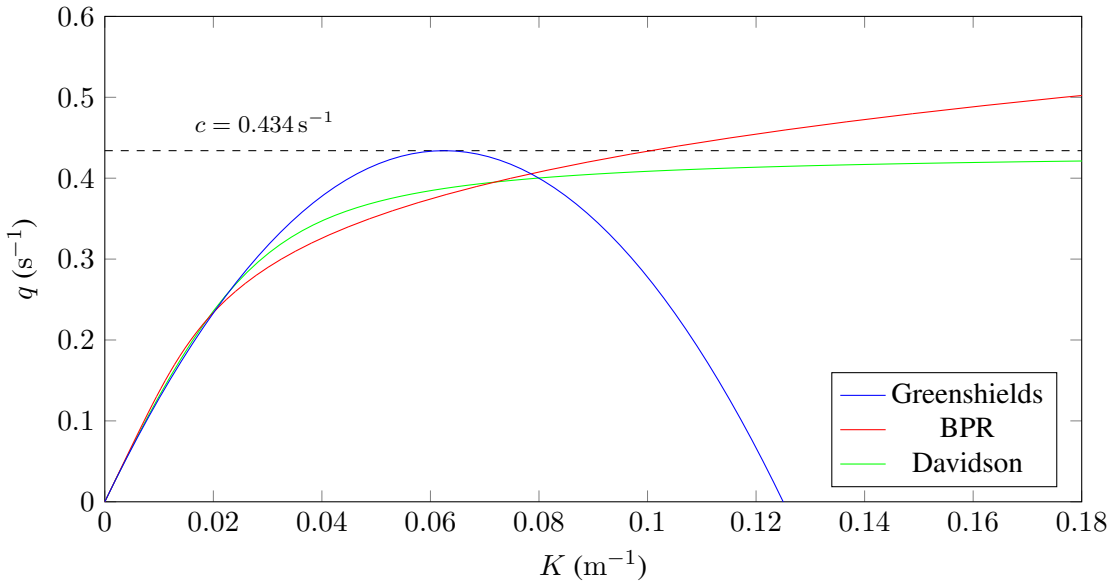


Figure 2.2: Fundamental diagram of traffic flow. The scenario is a single-lane local road with maximum speed $v_0 = 50\,\mathrm{km/h}$, minimum inter-vehicle distance $\lambda = 8\,\mathrm{m}$ ($K_{\text{jam}} = 1/\lambda = 0.125\,\mathrm{m}^{-1}$), and $J = 0.15$. For the models that require the value of the capacity c, we assume its value is the maximum flow according to the Greenshields model, calculated through Eq. (2.11) as being $c = 0.434\,\mathrm{s}^{-1}$. For the BPR model, c' is estimated from Eq. (2.15) to be $c' = c{\cdot}(\alpha{-}\alpha^2)/4 = 0.51 \cdot c = 0.221\,\mathrm{s}^{-1}$.

We can also plot the travel time against flow, Fig. 2.3, which is helpful for understanding how travel time behaves. The BPR and Davidson models already have travel time as a function of flow (Eqs. (2.6) and (2.8)), so we only need to find a closed form for the Greenshields model. From Eq. (2.10),

$$q = v_0\left(K - \frac{K^2}{K_{\text{jam}}}\right) \iff \frac{1}{K_{\text{jam}}}K^2 - K + \frac{q}{v_0} = 0 \tag{2.20}$$

$$\iff K = K_{\text{jam}} \cdot \frac{1 \pm \sqrt{1 - \dfrac{4q}{v_0 K_{\text{jam}}}}}{2} \tag{2.21}$$

If we replace this result in Eq. (2.5), we get

$$t = t_0 \cdot \frac{2}{1 \pm \sqrt{1 - \dfrac{4q}{v_0 K_{\text{jam}}}}} \tag{2.22}$$

As can be seen in Fig. 2.3, the "±" sign creates an interesting behavior. The blue full line portrays travel time when $q < c$ (free-flowing traffic) and corresponds to the "+" sign case; the blue dashed line portrays the travel time when $q > c$ (congested traffic) and corresponds to the "−" sign case.

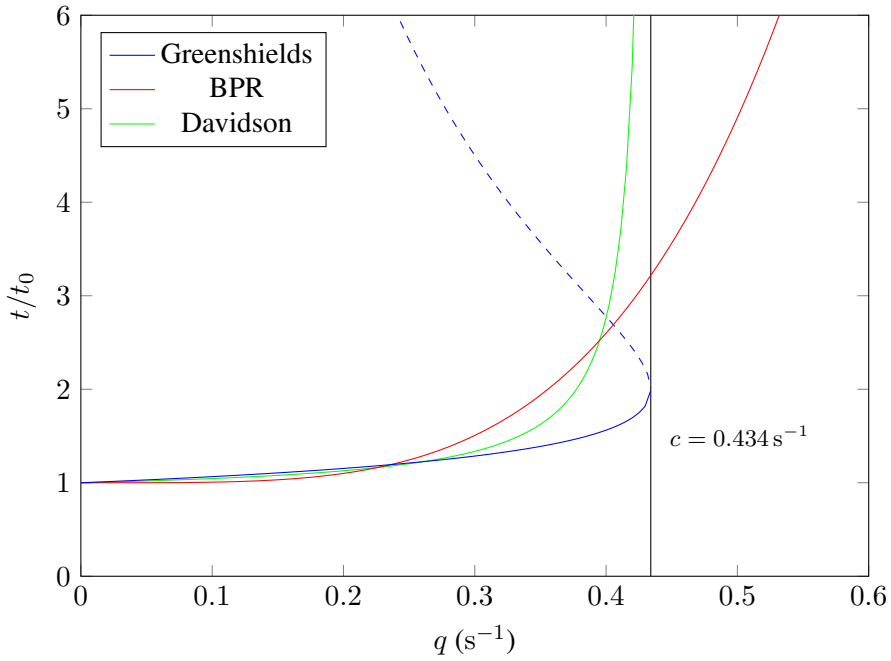


Figure 2.3: Link performance functions according to different models. The values of parameters are the same as in Fig. 2.2.

This is another reason why the Greenshields model is not common in practical implementations, since this issue must be solved either by:

- Modifying the formalization of the TAP as an optimization problem in a cumbersome way, where it has to be specified if a road is currently congested (case "−") or not (case "+") so the correct case can be picked;
- Using only one of the cases (e.g., only case "+"). This does not accurately represent the real phenomenon since it is fairly common for roads to be in a congested and non-congested state.

2.3.2 Capacity and travel time estimation

So far, it has been discussed how supply can be modeled using certain road parameters, but the issue of determining such road parameters as capacity and free-flow speed is still open. The simplest

and most reliable method to obtain a road's capacity and travel time is through local measurement, by measuring flows, delays, and travel times, and extrapolating a specific empirical model of that road that closely resembles its real behavior. This approach is, however, infeasible at the scale of an entire city, so the typical approach is to group streets by their most significant characteristics and then model each type of road differently so that road parameters may be estimated for roads for which we may only know its type.

Typical classification features according to the HCM (TRB, 2000, pp. 10.24 – 10.27) include:

- Area type: rural/outside cities, urban/inside cities, central business district (CBD)[2]/downtown;
- Road surface: asphalt/concrete, cobblestone, gravel;
- Road geometry and number of lanes;
- Density of intersections, driveways, parking spaces;
- Legal speed limits or typical FFS;
- Functional category (arterials, collectors, and streets);
- Other characteristics: separate left-turns, pedestrian activity, types of roads it connects.

For reference, the HCM (TRB, 2000, p. 8-28) mentions that experiments have reported a saturation flow rate of $1100\,\mathrm{veh/h}$ on a single-lane approach, and $2000\,\mathrm{veh/h}$ for an intersection with four evenly-loaded approaches.

2.3.3 Signalized intersections

The cycle time C in a signalized intersection can be divided into a red period r and a green period g for a given approach, such that $C = r + g$. These periods can be associated to the time intervals that the traffic lights are red and green, respectively, although the following two factors should be taken into account: (1) traffic lights often have a yellow period before turning red, and different drivers react to it differently; (2) vehicles do not start flowing instantly when the traffic light turns green, given that drivers have reaction times and starting up a vehicle is a complex and somewhat time-consuming task. As such, and for simplicity, we assume r is the effective red period, where the vehicle at the front of the queue is stopped, and g is the effective green period, in which cars are flowing.

In signaled intersections, TRB (2000, p. 15-9) suggests the following formula for its simplicity and intuitiveness to compute the through capacity of an intersection on an urban street:

$$c = N \cdot s \cdot \frac{g}{C} \tag{2.23}$$

where N is the number of through lanes, g is the green time per cycle, C the total cycle length, and s the adjusted saturation flow per through lane, estimated to be around $1700\,\mathrm{veh/h}$ for CBD (TRB, 2000, p. 10-24).

[2] CBD is an area with "higher number of parking maneuvers, pedestrians, and bus stops" (TRB, 2000, p. 10-26)

Sheffi (1985) takes a different approach, by applying queuing theory. It assumes an arrival rate[3] of λ, a departure rate of μ during the green period, and zero during the red period. The average queue length after the red period finishes is λr, so to determine the period g_0 required for the queue to dissipate, the following equation has to be solved:

$$\lambda(r + g_0) - \mu \cdot g_0 = 0 \tag{2.24}$$

where λr is the number of vehicles that entered the queue during the red period, $\lambda \cdot g_0$ is the number of vehicles that entered the queue during the green period while the queue was dissipating, and $\mu \cdot g_0$ is the number of vehicles that left the queue in time g_0. This yields

$$g_0 = r\frac{\lambda/\mu}{1 - \lambda/\mu} \tag{2.25}$$

The ratio $\rho = \lambda/\mu$ is known as the *utilization ratio*. By making geometric considerations regarding the input/output diagram of the process, Sheffi (1985, p. 352) concludes the total delay d and average delay per vehicle $\overline{d}$ are as follows:

$$d = \frac{1}{2} \cdot r \cdot \lambda(r + g_0) = \frac{\lambda r^2}{2(1 - \rho)} \tag{2.26}$$

$$\overline{d} = \frac{r^2}{2C(1 - \rho)} \tag{2.27}$$

However, this model consistently underestimates delays because it does not account for stochastic variations in the arrival rate. Therefore, Sheffi (1985) suggests adding a further component to account for random delays and traffic that waits through a full cycle,

$$\overline{d}' = \frac{R^2}{2\lambda(1 - R)} \tag{2.28}$$

where $R = \lambda/\nu$ is the effective utilization ratio, and $\nu = \mu(g/C)$ is the effective departure rate of the queue (resulting from a departure rate of μ during the green period, and zero during the red period). The resulting average delay per vehicle is thus calculated as

$$\overline{w} = \frac{r^2}{2C(1 - \rho)} + \frac{R^2}{2\lambda(1 - R)} \tag{2.29}$$

where the only required parameters are the cycle period C, the red period r, the departure rate during the green period μ, and the flow λ.

Eq. (2.29) can be simplified so that the free-flow delay $\overline{w}_0$ can be determined, and the relation between the delay and free-flow delay be shown (similarly to how the relation between travel time

[3] Note that flow is denoted by λ instead of q, as commonly used in queuing theory

t and free-flow time t_0 is shown in the BPR model):

$$\overline{w}_0 = \frac{r^2}{2C} \tag{2.30}$$

$$\overline{w} = \frac{\overline{w}_0}{1-\rho} + \frac{R^2}{2\lambda(1-R)} \tag{2.31}$$

Note that, with respect to Eq. (2.31), the first term has a vertical asymptote at $\lambda = \mu$, and the second term has a vertical asymptote at $\lambda = \nu$. Therefore, this model does not provide meaningful results for $\lambda \geq \nu$. This result is natural since a queuing theory model cannot allow a situation where the arrival rate μ is greater than the global service rate ν, given the queue length is expected to grow indefinitely, so there is no stable-state solution for this model.

2.3.4 Gap acceptance model

There are many situations where vehicles must merge onto higher-priority vehicle flows, most notably on unsignalized prioritized intersections and when changing lanes. These scenarios are particularly complex because instead of relying on external, predictable traffic management strategies like traffic lights or zipper merge, it depends heavily on flows in other edges, user behavior, and complicated rules of traffic yield implementation-wise.

The most common model for these situations is the gap acceptance model, where the main assumption is that a driver will merge onto the higher-priority flow if they are allowed a gap between consecutive vehicles in the higher-priority flow that is greater than a critical gap t_{cr}, which is the smallest gap that can be safely used to perform the merge maneuver. In some cases, users are assumed to accept any gap greater than t_{cr}, while in others, the probability of the user accepting a gap is also dependent on how much greater the gap is when compared to t_{cr}. This paradigm was first introduced by Herman and Weiss (1961).

For generality[4], it is assumed the probability of a driver accepting a gap follows a distribution with PDF $f_a(t)$ where $t < t_{cr} \implies f_a(t) = 0$. From this point onward, the deduction follows mostly the same logic as Sheffi (1985, pp. 354-357). For convenience, we consider the same scenario where a minor road meets with a major road in a T-intersection, where the minor yields to the major. It is then assumed that drivers in the minor only enter the major if the gap between two consecutive vehicles in the major is greater than or equal to a *critical gap* t_{cr} that allows a safe merge maneuver for a vehicle in the minor. Note that the mathematical findings for this scenario also apply to lane changing.

Let $f_T(t)$ be the PDF of the gaps in the major road, and let $F_T(t)$ be the corresponding CDF. The probability p of a gap being accepted is the probability $f_T(\tau)d\tau$ of it having length τ multiplied by the probability $P_a(\tau)$ of it being accepted:

$$p = \int_{t_{cr}}^{\infty} P_a(\tau) \cdot f_T(\tau)\, d\tau \tag{2.32}$$

[4]This is similar to the Herman and Weiss (1961) model where the probability $\alpha(t)$ of a driver accepting a gap of t was experimentally determined to be 0 if $t \leq t_{cr}$, and $1 - \exp\{-\lambda(t - t_{cr})\}$ otherwise.

A vehicle on the minor road will have to wait for as many N rejected gaps as necessary for gap $N+1$ to be large enough. This situation describes a geometric distribution $P(N)$ with probability p:

$$P(N) = (1-p)^N \cdot p \tag{2.33}$$

$$\mathrm{E}[N] = \frac{1-p}{p} \tag{2.34}$$

The expected waiting time $\mathrm{E}[W]$, Eq. (2.37), is the product of the expected number of rejected gaps $\mathrm{E}[N]$ and the expected length of a rejected gap $\mathrm{E}[T \mid \text{rejected}]$, Eq. (2.36), where the second factor can be estimated from

$$f_{T|\text{rejected}}(t) = \frac{(1 - P_a(t)) \cdot f_T(t)}{\displaystyle\int_0^{+\infty} (1 - P_a(\tau)) \cdot f_T(\tau)\, d\tau} \tag{2.35}$$

$$\mathrm{E}[T \mid \text{rejected}] = \int_0^{+\infty} \tau\, f_{T|\text{rejected}}(\tau)\, d\tau \tag{2.36}$$

$$\mathrm{E}[W] = \mathrm{E}[N] \cdot \mathrm{E}[T \mid \text{rejected}] \tag{2.37}$$

2.3.4.1 Unsignalized intersections

A peculiarity about the scenario of unsignalized intersections is that there is only at most one vehicle in the minor road actively trying to merge, forcing all other vehicles in the minor queue to wait until they reach the front of the queue. Therefore, as noted by Sheffi (1985, p. 356), $\mathrm{E}[W]$ is only the expected waiting time for the vehicle in front of the minor queue, so the time vehicles wait in the minor queue is still unaccounted for. Sheffi (1985) applies a correction by considering the queue in the minor street can be characterized by a Poisson arrival process and a single server with a general distribution of service time (where "processing" a vehicle is the phase where a vehicle is at the front of the queue and waiting for an adequately-sized gap in the major road traffic flow). The minor queue thus comes as an M/G/1 (arrivals are a Poisson process, service time with general distribution, one server) queue, where the total delay is given by

$$\mathrm{E}[D] = \frac{q\,\mathrm{E}[W^2]}{2(1 - q\,\mathrm{E}[W])} + \mathrm{E}[W] \tag{2.38}$$

where q is the flow in the minor road, and $\mathrm{E}[W^2]$ is the second moment of W. Appendix B contains further deductions under the assumption that arrivals in the major road follow a Poisson process.

From Eq. (2.38), two significant properties of the delay can be inferred: (1) since the flow q is in the numerator of the first term, the free-flow time is composed only of the second term, so $t_0 = \mathrm{E}[W]$; (2) because the denominator contains the factor $1 - q\,\mathrm{E}[W]$, the delay function presents a vertical asymptote at $q = 1/\mathrm{E}[W]$, which makes sense since $\mathrm{E}[W]$ is the expected time

it takes for the vehicle in front of the queue to leave the minor road, so the maximum capacity of the minor is $c = 1/\operatorname{E}[W]$ to prevent the minor queue from growing indefinitely.

This model can be extended to scenarios where minor and major roads have more than one lane. If the minor road has more than one lane, the two lanes may be separated into distinct instances of the same problem, and deal with them separately. In the hypothetical scenario where two lanes from a minor road converge on the same right-most lane in a major road, the only conflicting flow for the left-most minor lane is the flow in the major road, while the conflicting flows for the right-most minor lane are the sum of the flow in the major road and the flow in the left-most minor lane. In another scenario where vehicles on the minor road have to cross multiple major road lanes to get to their destination, it is possible to consider the summed flows in the lanes the vehicles from the minor road have to cross.

Chapter 3

Static traffic assignment

In a TAP, the traffic assignment stage is where demand interacts with supply. The classical study of traffic assignment is made along the lines of static traffic assignment (STA), where the system is assumed to be in a "steady state" equilibrium solution, the link flows and capacities are measured with respect to the entire time period (Boyles et al., 2019, p. 127), and demand does not change significantly over that period. These assertions lead to the following observations that simplify the problem:

- Congestion conditions do not change with time, so vehicle movements can be analyzed in terms of macroscopic aggregate measures such as vehicle flow, density, and average speed.
- The cost of crossing a road varies as a function of flow in that road. This cost function is responsible for encoding all the complexity of inter-vehicular interactions.

To apply STA, congestion conditions and demand are considered not to change significantly so that steady-state analyses can be applied. This requires careful selection of the period under analysis, as it needs to be at least as long as the longest trip, but not too large where this assumption becomes inaccurate (Sheffi, 1985, p. 20).

3.1 Formalization

The TAP can be formalized as an optimization problem using the notation of mathematical programming. This is the most generally-accepted formalization for static analysis and is based on Boyles et al. (2019, p. 149) and Sheffi (1985, pp. 59-69).

As discussed in Sec. 2.3, the road network (supply) is represented as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$. The OD pairs (demand) are represented by a matrix $\mathbf{D}$ with dimensions $|\mathcal{N}| \times |\mathcal{N}|$, where element d^{ij} denotes the demand from node i to j. For a specific OD pair (r, s), the set of all paths π from r to s is represented as Π^{rs}. The set of all possible paths between every OD pair is represented as $\Pi = \bigcup_{r,s \in \mathcal{N}} \Pi^{rs}$. It is often helpful to know how much flow each arc has, in contrast to the flow through each path f^{π}. Therefore, the flow x_a through arc a is defined as the sum of flows of all

paths that include arc a:

$$x_a = \sum_{\pi \in \Pi : a \in \pi} f^\pi \tag{3.1}$$

In mathematical programming, membership (in this case, the test $a \in \pi$) is better described using a binary variable. Binary variable δ_a^π is then defined as

$$\delta_a^\pi = \begin{cases} 1 & : a \in \pi \\ 0 & : \text{otherwise} \end{cases} \tag{3.2}$$

Eq. (3.1) can now be rewritten as

$$x_a = \sum_{\pi \in \Pi} f^\pi \delta_a^\pi \tag{3.3}$$

For conciseness, let $\mathbf{f}$ be the vector of flows in all paths, and $\mathbf{x}$ be the vector of flows in all arcs. The TAP can then be written in the canonical form,

$$\min \quad f(\mathbf{x}) \tag{3.4}$$

$$\text{s.t.} \quad x_a = \sum_{\pi \in \Pi} f^\pi \delta_a^\pi \qquad \forall a \in \mathcal{A} \tag{3.5}$$

$$\sum_{\pi \in \Pi^{rs}} f^\pi = d^{rs} \qquad \forall r, s \in \mathcal{N} \tag{3.6}$$

$$f^\pi \geq 0 \qquad \forall \pi \in \Pi \tag{3.7}$$

Eq. (3.4) sets the cost to be minimized as a function of the flows $\mathbf{x}$ through all arcs. Eq. (3.5) defines x_a as in Eq. (3.3). Eq. (3.6) imposes that the flows for all paths between an OD pair (r, s) meet the demand for that OD pair. Finally, Eq. (3.7) imposes that flows through paths are never negative, so the solution has physical meaning. The cost function is left unspecified since it may vary depending on whether the TAP is solved for the system optimum or for user equilibrium.

The following notation will be used throughout this section: $\mathbf{x}^n$ denotes the edge flows solution of iteration n in an iterative process; similarly, and for brevity, $z^n = z(\mathbf{x}^n)$ is the cost at iteration n (do not confuse the scalar z^n with the objective function z). The optimal solution is denoted by $\mathbf{x}^\infty$, and its corresponding optimal cost is z^∞; the use of the concept of infinity is justified under the assumption that all considered iterative processes converge to the optimal solution, so the solution after an infinite number of iterations $\lim_{n \to \infty} \mathbf{x}^n$ is the optimal solution.

3.2 All-or-nothing assignment

In the *all-or-nothing* variant of STA, cost does not vary with flow, so user behavior does not change route costs, and as such, each user will select the path that is *shortest* in terms of costs (Sheffi, 1985, p. 73; Boyles et al., 2019, p. 187). Assignment in this scenario is done by finding the shortest path for each OD and then assigning all flow of that OD to that shortest path.

The task at hand has now been reduced to finding the shortest paths between all ODs with positive flow. This can be done in a highly concurrent way by noting that the problem of finding the distance between all pairs of nodes in a graph with N nodes and A arcs is the same as solving N one-to-many shortest paths problems. These sub-problems can be solved using Dijkstra's algorithm in $O((A+N)\log N)$, yielding a total execution time of $O(N(A+N)\log N)$, or $O(N^2 \log N)$ if the graph is assumed to be sparse.

3.3 Non-uniqueness of path flows

As shown by Eq. (3.1), there is an intrinsic relation between path flows f_π and arc flows x_a. This relation is definitional in nature, meaning that arc flows are defined as a function of path flows, so for each path flow configuration $\mathbf{f}$, there is a single arc flow configuration $\mathbf{x}$. However, this relation is not symmetric: a certain arc flow configuration $\mathbf{x}$ can correspond to many different path flow configurations, as shown by Boyles et al. (2019, Sec. 6.2.2). To avoid the cumbersome process of finding a suitable path flow configuration for a given arc flow configuration, the analysis can be limited only to methods that construct the path flows or allow to retrieve them easily, instead of having to calculate them *a posteriori*. For instance, the Frank-Wolfe algorithm (FW) discussed in Sec. 3.6 can be adapted to construct the final path flows from the results of the successive all-or-nothing assignments.

3.4 Equilibrium conditions

Now that the formal framework of the TAP was presented, it is necessary to decide on the *type* of the TAP that will be solved. Literature typically mentions system optimal TA and user equilibrium TA, both of which were formulated by Wardrop (1952).

3.4.1 System optimum

The system optimum (SO), or Wardrop's second principle, is achieved when the selected values of $\mathbf{f}$ (and therefore $\mathbf{x}$) are such that the total time spent by all vehicles in the system is minimized. This scenario is expressed by the following cost function:

$$f(\mathbf{x}) = \sum_a x_a \cdot t_a(x_a) \tag{3.8}$$

3.4.2 User equilibrium

User equilibrium (UE), also known as Wardrop's first principle, or Wardrop equilibrium, arises from the assumption that users act selfishly to minimize their own travel time, regardless of the other users in the system. In such a situation, one may assume that the system has an initially arbitrary assignment of flows $\mathbf{f}$ that cause roads to have differing degrees of congestion; therefore, each user that is currently taking a certain path will try to unilaterally pick the fastest available path

taking into account current congestion conditions. The system will reach a state of equilibrium when no user can unilaterally change their current path to a faster path.

This behavior can be motivated in part by human nature, but it is also conditioned by the significant fact that users do not have the means to communicate with each other or with a centralized controller to coordinate a strategy, and therefore their only means of communication is indirect, by ascertaining the current congestion conditions, and affecting those congestion conditions through their path choices.

UE generally assumes users have perfect information and unbounded rationality, meaning that users have instantaneous access to the congestion information, consider all possible alternatives, and pick the absolute best for them (Sheffi, 1985, p. 261). In practice, users work with imperfect information (since they do not have instantaneous access to exact and complete information) and bounded rationality (the thought process is limited in time, and people tend to consider only a handful of possible paths), a scenario known as *stochastic user equilibrium*. However, because these practical requirements are hard to model, and because UE has been found to model the way users move in a traffic network fairly well, this concept of UE is still widely used.

If integer flows are used, the equilibrium condition is reached when, for each OD pair (r, s), a path p is found that meets the following condition:

$$\nexists \pi \in \Pi^{rs} : \sum_{a \in \pi} (x_a + 1) t(x_a + 1) < \sum_{a \in p} x_a t(x_a) \tag{3.9}$$

where the left-hand side is the cost of a candidate path π if one vehicle changes from path p to path π, and the right-hand side is the current cost of path p.

If flows are assumed to be real numbers, Eq. (3.9) can be restated as

$$\begin{aligned} &\nexists \pi \in \Pi^{rs} : \sum_{a \in \pi} (x_a + \varepsilon) t(x_a + \varepsilon) < \sum_{a \in p} x_a t(x_a) \\ \iff &\nexists \pi \in \Pi^{rs} : \sum_{a \in \pi} x_a t(x_a) < \sum_{a \in p} x_a t(x_a) \end{aligned} \tag{3.10}$$

where ε is an infinitesimal amount of flow that could be moved from path p to path π. Because ε is infinitesimal, it can be removed from Eq. (3.10) to yield a simpler equilibrium condition. Real flows adequately represent this problem if the number of vehicles and resulting path/arc flows are significantly larger than the granularity of integer numbers; flows can then be rounded off at the end if necessary to yield a good-quality result. While this integer variant may not be the exact solution of the problem, it is still a very good approximation.

Using the method of Lagrange multipliers, it is possible to deduce the formula of the UE objective function (Boyles et al., 2019, pp. 148-149):

$$f(\mathbf{x}) = \sum_a \int_0^{x_a} t_a(x) dx \tag{3.11}$$

The full formulation of the UE traffic assignment problem is known as Beckmann's formulation

(Beckmann et al., 1956, p. 63), and is shown below for completeness. The original formulation also accounted for elastic demand (Sheffi, 1985, p. 160), but here fixed demand is assumed, so that term was removed from the objective function.

$$\min \quad \sum_a \int_0^{x_a} t_a(x)dx \tag{3.12}$$

$$\text{s.t.} \quad x_a = \sum_{\pi \in \Pi} f^\pi \delta_a^\pi \qquad \forall a \in \mathcal{A} \tag{3.13}$$

$$\sum_{\pi \in \Pi^{rs}} f^\pi = d^{rs} \qquad \forall r, s \in \mathcal{N} \tag{3.14}$$

$$f^\pi \geq 0 \qquad \forall \pi \in \Pi \tag{3.15}$$

3.4.3 Comparison between SO and UE

The system-optimal assignment can only be reached if every user can coordinate with each other, while user equilibrium implies that users only communicate with each other through their actions. Also, in the absence of any argument as to why some users may be more important than others, SO assumes all users are equally important and have the same cost function, since the travel time of each user counts towards the cost of the optimal solution with the same weight. In UE this is not an issue, since no global cost function is required to determine the equilibrium solution, as users make selfish, unilateral decisions based on their own individual cost functions.

The UE and SO objective functions are quite similar. In fact, it is possible to convert a UE problem into a SO problem and vice-versa. Let $T_a(x)$ be the cost function to solve the SO variant, and $t_a(x)$ be the cost function to solve the UE variant. The fundamental equation that allows us to convert cost functions between the UE and SO variants is

$$\sum_a \int_0^{x_a} t_a(x)\,dx = \sum_a x_a T_a(x_a) \iff \int_0^{x_a} t_a(x)\,dx = x_a T_a(x_a) \tag{3.16}$$

Refer to Tab. 3.1 for the conversions between SO and UE cost functions.

Table 3.1: Equations to convert SO and UE cost functions. Both are deduced from Eq. (3.16).

SO		**UE**	
$T_a(x_a)$	$\Longrightarrow$	$t_a(x_a) = T_a(x_a) + x_a T_a'(x_a)$	(3.17)
$T_a(x_a) = \dfrac{1}{x_a}\displaystyle\int_0^{x_a} t_a(x)\,dx$	$\Longleftarrow$	$t_a(x_a)$	(3.18)

Conversion from SO to UE is most useful when solving using FW since it was assumed in the theoretical background of this document that the goal was UE, which simplified the formulas down the line. In most scenarios where t_a and T_a follow simple analytical expressions, it is possible to derive equally simple conversions, since the only requirements are that T_a can be derivated and t_a

can be integrated. Even in scenarios where cost functions do not have analytical forms, it may be possible to pre-compute T'_a and $\int t_a \, dx$.

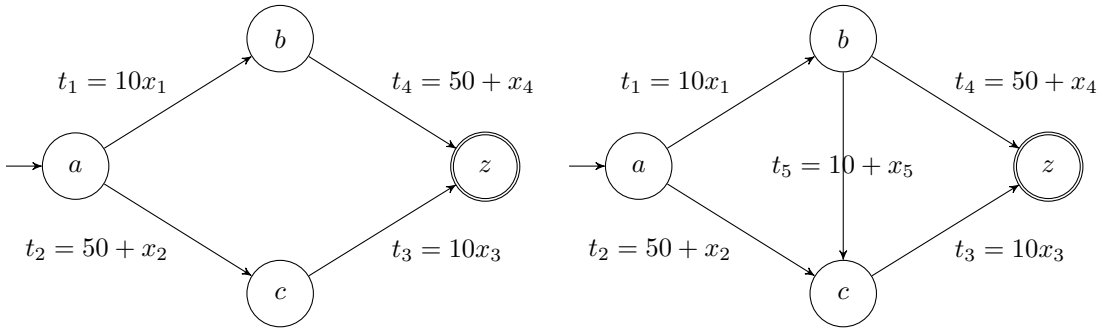
3.4.4 Price of anarchy and Braess's paradox

As already stated in Chap. 1 and Sec. 3.4.2, the UE solution is not optimal, as a result of individuals being able to make selfish decisions. The difference in cost between UE and SO is the materialization of the inefficiencies of drivers' selfishness, and is known in the literature as the *price of anarchy*; i.e., the price the system has to pay to let users choose their routes (Morandi, 2021).

Not only is UE often sub-optimal, but it also leads to some interesting results, such as the conclusion that, in some scenarios, adding a new road worsens travel times for all vehicles. This is called Braess's paradox, after the article by Braess (1968), and is widely known in literature (Sheffi, 1985, pp. 75-77; Boyles et al., 2019, pp. 136-138; Morandi, 2021). Braess (1968) demonstrates the paradox by considering the network in Fig. 3.1. It is assumed the flow from a to z is 6. Before the road is added (Fig. 3.1a), the equilibrium solution is where $t_1 + t_4 = t_2 + t_3$; it is also natural that $x_1 = x_4$ and $x_2 = x_3$, so

$$t_1 + t_4 = t_2 + t_3 \iff 10x_1 + 50 + x_1 = 50 + x_2 + 10x_2 \iff x_1 = x_2 \tag{3.19}$$

which gives $x_1 = x_2 = 3$, with a total cost of $3(10 * 3 + 50 + 3) + 3(50 + 3 + 10 * 3) = 498$, so each vehicle experiences a travel time of 83.



(a) (b)

Figure 3.1: Braess's network, demonstrating Braess's paradox. Adapted from Braess (1968).

When a new road $b \to c$ is added (Fig. 3.1b), the new equilibrium conditions are

$$\begin{cases} t_4 = t_5 + t_3 \\ t_1 + t_5 = t_2 \end{cases} \iff \begin{cases} 50 + x_4 = 10 + x_5 + 10(x_2 + x_5) \\ 10(x_4 + x_5) + 10 + x_5 = 50 + x_2 \end{cases} \tag{3.20}$$

$$\iff \begin{cases} 10x_2 - x_4 + 11x_5 = 40 \\ -x_2 + 10x_4 + 11x_5 = 40 \end{cases} \tag{3.21}$$

which has solution $x_2 = x_4 = x_5 = 2$, with a total cost of $(2+2)*(10*(2+2))+2*(50+2)+2*(10+2)+2*(50+2)+(2+2)*(10*(2+2)) = 552$, so each vehicle experiences a travel time of 92. This is a paradox since the infrastructure was improved, but travel times worsened for everyone.

3.5 As a convex optimization problem

It is important to mention that if t_a satisfies certain properties, then TA meets the criteria of a convex optimization problem and can therefore be solved using well-known methods (see Sec. 3.6).

Aside from the properties of the real-valued TAP that are required for the problem to be convex (namely, that the set of feasible path assignments $\mathbb{H}$ and the set of feasible arc assignments $\mathbb{X}$ are both compact and convex), the fundamental property is that the objective function f must be convex.

Convexity Consider the UE formulation. The first partial derivative of f is $\partial f/\partial x_a = t_a(x_a)$. If Hessian matrix $\mathbf{H}_f$ of f is taken, it yields

$$\frac{\partial^2 f}{\partial x_a \partial x_b} = \begin{cases} \dfrac{\partial t_a}{x_a} & : a = b \\ \dfrac{\partial t_a}{x_b} & : a \neq b \end{cases} \tag{3.22}$$

By applying Eqs. (2.2) and (2.3) to Eq. (3.22) it leads to

$$\frac{\partial^2 f}{\partial x_a \partial x_b} = \begin{cases} t'_a(x_a) & : a = b \\ 0 & : a \neq b \end{cases} \tag{3.23}$$

where it is known that $t'_a(x_a)$ is always positive because of Eq. (2.3). Thus $\mathbf{H}_f$ is a positive-definite matrix since it is diagonal and its entries are strictly positive, as such making f strictly convex (Boyles et al., 2019, pp. 153-155).

Uniqueness Sheffi (1985, p. 213) mentions that, aside from Eq. (2.3), the other requirement for the solution to be unique is given by Eq. (3.24)

$$\frac{\partial t_a}{\partial x_a} > \sum_{b \neq a} \frac{\partial t_a}{\partial x_b} \tag{3.24}$$

which in practice translates to the simple statement "a change in flow through a is more important in defining the cost of traversing it than all changes in other flows combined".

3.6 Frank-Wolfe algorithm

The Frank-Wolfe algorithm (FW) is an algorithm that solves convex programming problems with linear constraints. Also known as the *convex combination algorithm*, it was originally proposed by Frank and Wolfe (1956) to solve quadratic programming problems.

The algorithm provides a solution for a convex program generically formalized as

$$\min \quad z(\mathbf{x}) \tag{3.25}$$

$$\text{s.t.} \quad \sum_i h_{ij} x_i \geq b_j \quad \forall j \in \mathcal{J} \tag{3.26}$$

The Frank-Wolfe algorithm works similarly to gradient descent in that both choose a direction that locally reduces the objective function and move in that direction by a certain step. However, unlike gradient descent, FW does not necessarily (and in reality, often does not) choose the gradient as the direction.

To solve the convex optimization problem (which is manifestly hard to solve), FW splits each iteration into two auxiliary problems: one where a multidimensional linear approximation is solved and another where a unidimensional constrained convex problem is solved.

Given a current feasible solution $\mathbf{x}^n$, the algorithm has the following steps at iteration $n+1$:

1. **Direction-finding.** Find $\mathbf{x}^*$ that limits the value of λ.
2. **Step-size determination.** Find α that minimizes $z(\mathbf{x}^n + \alpha(\mathbf{x}^* - \mathbf{x}^n))$, subject to $\alpha \in [0, 1]$.
3. **Move.** $\mathbf{x}^{n+1} \leftarrow \mathbf{x}^n + \alpha(\mathbf{x}^* - \mathbf{x}^n)$.
4. **Stopping criteria.** If the iterative process meets stopping criteria, stop.

Direction-finding Generally, the best direction to move in a minimization problem is $-\boldsymbol{\nabla} z$ since it has the greatest downward slope. However, if the iteration cannot move significantly in that direction, the iteration will have little impact. Therefore, FW prefers to consider the direction that, although not having the greatest slope, may be compensated for by being sustained by a larger move (higher value of α).

In this first step, the first-degree approximation of $z(\mathbf{x}^*)$ at point $\mathbf{x}^n$ is considered, as given by

$$\tilde{z}(\mathbf{x}^*) = z(\mathbf{x}^n) + \boldsymbol{\nabla} z(\mathbf{x}^n) \cdot (\mathbf{x}^* - \mathbf{x}^n) \tag{3.27}$$

The first-degree approximation can be simplified by noting that $\mathbf{x}^n$ is a constant, and therefore $z(\mathbf{x}^n)$ and $-\boldsymbol{\nabla} z(\mathbf{x}^n) \cdot \mathbf{x}^n$ are also constants; thus, the alternative function $\tilde{z}'(\mathbf{x}^*) = \boldsymbol{\nabla} z(\mathbf{x}^n) \cdot \mathbf{x}^*$ gives the same result as the linear approximation, constants aside.

The direction-finding step can then be formalized as the problem of finding the solution $\mathbf{x}^*$ of the multidimensional linear program presented in Eqs. (3.28) and (3.29).

$$\min \quad \nabla z(\mathbf{x}^n) \cdot \mathbf{x}^* \tag{3.28}$$

$$\text{s.t.} \quad \sum_i h_{ij} x_i \geq b_j \quad \forall j \in \mathcal{J} \tag{3.29}$$

Since this is a linear programming problem, its solution will lie in the frontier of the solution set, where at least one constraint (but most often two constraints) will be binding.

Step-size determination Having found $\mathbf{x}^*$ using a linear approximation of z, the corresponding optimal step is necessary. Unlike in step **1.**, in this step the actual function z is used instead of an approximation which makes this problem harder in that aspect. On the other hand, this step only has one solution variable, the step α.

The goal is to find the solution $\mathbf{x}^{n+1}$ between $\mathbf{x}^n$ and $\mathbf{x}^*$ that minimizes z; in other words, find α that minimizes $z(\mathbf{x}^n + \alpha(\mathbf{x}^* - \mathbf{x}^n))$. The lower bound of α is 0, since it is established that the direction of movement is $\mathbf{x}^* - \mathbf{x}^n$, not in the opposite direction. Because $\mathbf{x}^*$ is at the frontier of the solution set, $\mathbf{x}^{n+1}$ cannot go any further than $\mathbf{x}^*$, so α has an upper bound of 1. This step can then be formalized as the problem of finding the solution α of the following unidimensional constrained convex problem.

$$\min \quad z(\mathbf{x}^n + \alpha(\mathbf{x}^* - \mathbf{x}^n)) \cdot \mathbf{x}^* \tag{3.30}$$

$$\text{s.t.} \quad \sum_i h_{ij} x_i \geq b_j \quad \forall j \in \mathcal{J} \tag{3.31}$$

$$0 < \alpha \leq 1 \tag{3.32}$$

This problem can be solved using any unidimensional minimization algorithm, e.g., Newton's method.

3.6.1 Frank-Wolfe algorithm applied to traffic assignment

The structure of the TAP significantly simplifies the process of solving it using the FW algorithm. Knowing the solution $(\mathbf{x}^n, \mathbf{f}^n)$ of the previous iteration, and having found an auxiliary solution $(\mathbf{x}^*, \mathbf{f}^*)$, step **2.** is trivial since the solution set is convex. This means any solution of the form $\mathbf{f} = (1 - \alpha) \cdot \mathbf{f}^n + \alpha \cdot \mathbf{f}^*$ (where $\alpha \in [0, 1]$) is feasible, and that the corresponding flows for this solution are

$$x_a^* = (1 - \alpha) \cdot x_a^n + \alpha \cdot x_a^* \tag{3.33}$$

As such, this problem can be solved by repeated computation of the original cost function f for different possible solutions $\mathbf{x}$.

Step **1.** is the opposite because of its apparent difficulty from a naïve perspective. It is assumed the problem is being solved for UE, but this method can also be adapted to solve for SO. The

derivative of Eq. (3.11) is

$$\frac{\partial z}{\partial x_a}(\mathbf{x}) = t_a(x_a) \tag{3.34}$$

Therefore,

$$\nabla z(\mathbf{x}^n) \cdot \mathbf{x}^* = \sum_a t_a^n \cdot x_a^* \tag{3.35}$$

where $t_a^n = \partial z / \partial x_a(\mathbf{x}^n) = t_a(x_a^n)$, so t_a^n are constants. In short, the auxiliary solution $\mathbf{x}^*$ is found by assuming that the change in costs is going to be small, and therefore assuming that arcs' costs do not depend on flow.

Thus, the problem becomes

$$\begin{aligned}
\min \quad & \sum_a t_a^n \cdot x_a && && (3.36)\\
\text{s.t.} \quad & x_a = \sum_{\pi \in \Pi} f^\pi \delta_a^\pi && \forall a \in \mathcal{A} && (3.37)\\
& \sum_{\pi \in \Pi^{rs}} f^\pi = d^{rs} && \forall r, s \in \mathcal{N} && (3.38)\\
& f^\pi \geq 0 && \forall \pi \in \Pi && (3.39)
\end{aligned}$$

A naïve approach would imply solving this method using one of the well-known linear programming algorithms, such as Simplex. Note that our decision variables are f_π, of which there are as many as there are paths, which are too many for this problem to be tractable as-is. However, a very interesting property of this problem can be noted. Because arc costs are considered constants, this step simplifies into an easily-solvable all-or-nothing assignment (see Sec. 3.2).

3.6.2 Stopping criteria

There are several simple stopping criteria that are applied throughout the field of mathematical optimization, such as number of iterations or difference between the costs of consecutive solutions. But, given the specificities of our scenario where FW is applied to traffic assignment, it is possible to devise new stopping criteria that make the final solution more meaningful.

Consider the formulation of the linear approximation $\tilde{z}$ of the cost function z given in Eq. (3.27). Because z is convex (shown in Sec. 3.5), any linear approximation $\tilde{z}$ of the cost function z at any point $\mathbf{x}^n$ always underestimates the true cost at $\mathbf{x}$:

$$\forall \mathbf{x}, z(\mathbf{x}) \geq \tilde{z}(\mathbf{x}) \tag{3.40}$$

For a fixed value of $\mathbf{x}^n$, the auxiliary solution $\mathbf{x}^*$ minimizes $\tilde{z}$, hence $\forall \mathbf{x}, \tilde{z}(\mathbf{x}) \geq \tilde{z}(\mathbf{x}^*)$. From these two facts it can be derived that, given a linear approximation $\tilde{z}$ of z at any point $\mathbf{x}^n$,

$$z^\infty \geq \tilde{z}(\mathbf{x}^*) \tag{3.41}$$

Thus, as noted by Sheffi (1985, p. 103), $\tilde{z}(\mathbf{x}^*)$ is a lower bound for z^∞, regardless of the choice of $\mathbf{x}^n$. Additionally, $\tilde{z}(\mathbf{x}^*)$ can be easily calculated from Eq. (3.27),

$$\begin{aligned}\tilde{z}(\mathbf{x}^*) &= z(\mathbf{x}^n) + \boldsymbol{\nabla} z(\mathbf{x}^n) \cdot (\mathbf{x}^* - \mathbf{x}^n) \\ &= z^n + \sum_a t_a(x_a^n) \cdot (x_a^* - x_a^n)\end{aligned} \tag{3.42}$$

Sheffi (1985) also notes however that this lower bound $\tilde{z}(\mathbf{x}^*)$ is not monotonically increasing, as observed in specific examples (Sheffi, 1985, p. 107). But since all values $\tilde{z}(\mathbf{x}^*)$ are lower bounds, the maximum of $\tilde{z}(\mathbf{x}^*)$ over all iterations performed so far can be taken as the strictest lower bound that is known up until a certain iteration, making this lower bound monotonically non-decreasing. Let this lower bound be z_{LB}^n.

It can be stated that z_{LB}^n also converges to z^∞, owing to two facts, the first being that z_{LB}^n is non-decreasing. The second fact is the observation that, if the current iteration $\mathbf{x}^n$ is in the neighborhood of $\mathbf{x}^\infty$ (so we can state in a more loose sense formalism-wise that $\mathbf{x}^n \to \mathbf{x}^\infty$), then $\boldsymbol{\nabla} z(\mathbf{x}^n) \to 0$ and as such $\tilde{z}(\mathbf{x}^*) \to z(\mathbf{x}^n) \to z^\infty$. This means that, at each iteration n, the optimal cost can be bounded to

$$z^\infty \in [z^n, z_{LB}^n] \tag{3.43}$$

Rose et al. (1988, p. 264) proposes a stopping criteria using Eq. (3.43), namely based on the value of the *absolute gap* $A^n = z^n - z_{LB}^n$ as the amplitude of the range of possible values of z^∞, or the more commonly used *relative gap*,

$$S^n = \frac{z^n - z_{LB}^n}{z^n} \tag{3.44}$$

Common values of S^n are in the order of 10^{-4} (Camargo, 2018).

Chapter 4

Traffic simulation models

Aside from the classical STA model shown earlier, there are several different ways to model traffic that yield different results that may be more suitable for different applications. Namely, dynamic approaches to traffic simulation, also known as dynamic traffic assignment (DTA), have become more common due to their numerous advantages in terms of more realistically modeling congestion by tracking changes at a fairly fine time resolution (Boyles et al., 2019, p. 16), so time becomes a relevant factor in changing congestion conditions.

We begin by defining the important concept of *environment*. It is the virtual representation of the portion of the real world that we aim to simulate. It includes a model of the real world (e.g., roads and service functions), as well as the current state (e.g., current time, vehicle positions, traffic light phases).

4.1 Network description

Notwithstanding the previous formulation of a traffic network for STA, a more nuanced definition of a traffic network is now presented for DTA. This definition is similar to that used in the SUMO traffic simulator (Krajzewicz et al., 2002).

A road is designated as an *edge*. All edges start and end at *junctions*, which are space-limited objects that serve as interchanges where vehicles may change between edges. The rules as to which lanes of which edges are connected are encoded by *connections*. Connections can have higher or lower priority; a connection c_1 is said to *respond* to another connection c_2 if traffic using (or about to use) c_1 has higher priority than traffic using (or about to use) c_2, and as such vehicles intending to use c_1 must yield the right of way to vehicles using c_2. Traffic lights are associated to junctions and are controlled by *traffic light programs*, where a traffic light program has many *phases*; each phase specifies the color of the traffic light being shown at that instant for each connection in the junction that the traffic light program controls.

4.2 Level of detail

The main criterion for grouping traffic simulation models is the level of detail to which they represent the real phenomena. Models can be divided into three classes: macroscopic, microscopic, and mesoscopic (Passos et al., 2011), as shown in Fig. 4.1.

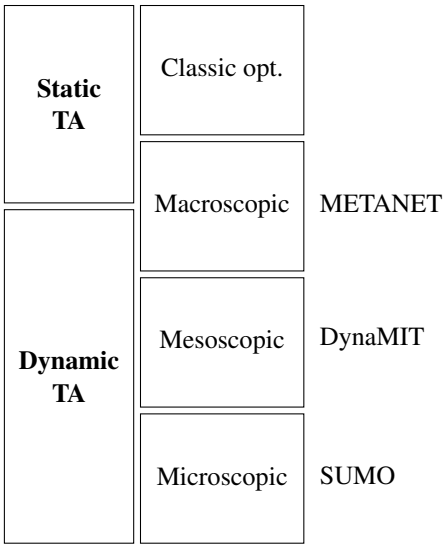


Figure 4.1: Simplified classification of traffic models by level of detail.

4.2.1 Macroscopic models

Macroscopic models are the most akin to classical STA approaches. These models represent intersections in a very simple way, and traffic flow is described as a continuum flow in analogy with continuum models for fluids, so vehicle movement is ruled by macroscopic state variables such as vehicle flow, average speed, and density (Nor Azlan and Md Rohani, 2018; Kessels, 2019, p. 53). Macroscopic models tend to be the simplest since they aim at representing the TAP as a mathematical simplified version that, although quite separated from reality, still represents it in an analytically tractable way. These models provide the greatest opportunity for applying mathematical and classical algorithmic techniques.

4.2.2 Microscopic models

Microscopic models result from applying simulation techniques to the TAP. They aim to represent each vehicle and its associated parameters in a very detailed and accurate way by simulating vehicles as physical objects with dimensions, position, and speed (Ferrara et al., 2018, p. 113). Vehicles interact with other vehicles in their immediate neighborhood by following a set of empirically-determined rules that encode the way people handle headway/spacing, breaking/accelerating patterns, reaction times, lane changing, how free-flow speed is selected and how curves are negotiated (Kessels, 2019, p. 35). Some of the most well-known microscopic model concepts are car-following, introduced by Newell (1961), and lane changing, first formalized by Gipps (1986).

Microscopic models often require time-step-based simulation (Crociani et al., 2017), which updates vehicles' parameters in instants that may not be of interest in the context of a certain study, typically making them the most computationally demanding. They also require a large number of parameters to be properly calibrated for results to be reliable (Ferrara et al., 2018, p. 113). On the other hand, microscopic models allow for rules to be easily represented and added/modified since those rules are expressed in terms of variables that drivers observe, such as leading vehicle speed.

4.2.3 Mesoscopic models

Mesoscopic models are the most recent development in traffic assignment and simulation. It is also the most broadly defined class of traffic models, since their overarching goal is to have performance similar to macroscopic models, while still representing traffic in an accurate and time-dependent way (Ferrara et al., 2018, p. 114; Kessels, 2019, p. 99), where the vehicle is the quantum unit that cannot be completely treated using aggregate measures.

It is in mesoscopic models that there is the most mathematical complexity, because it seeks to reconcile the paradigm of the vehicle as the quantum with a description of vehicle flow using aggregate terms such as in probability distributions (Kessels, 2019, p. 99). In a mesoscopic model, the precision-performance tradeoff is only improved by the development of new, often complex methodologies that improve performance without sacrificing precision significantly.

4.3 Time-driven vs event-driven

Time-driven simulation

The simplest model is one where the environment state is updated periodically at a sufficiently small time step δt. This paradigm is often called *time-driven simulation*, and finds most use in the simulation of natural or continuous systems, although it can also be repurposed as a simple approach to discrete-event systems where any individual part of the simulation may affect any other parts. It is common for such simulation models to update the entire state of the environment at every time step, which proves particularly useful when the simulated environment is being visualized as the simulation executes, since the state at every point is consistent.

However, recalculating the entire state of the system is infeasible in practically interesting situations, such as when simulating traffic in an entire city, as it implies updating the entire state of the environment even though no significant events occurred in most of the environment. Besides, even if at each time step all the events that need to be triggered since the last update at time $t - \delta t$ were considered, there is always an issue with selecting a small enough δt that provides decent granularity, but not too small as to trigger too many timestep events with close to zero updates to the system state.

Event-driven simulation

As such, approaches to discrete system modeling often move towards a simulation process where the environment state is updated only when absolutely required, and exactly at the instant that the event occurs. This paradigm is called *event-driven simulation*. For an efficient event-driven simulation model, a good definition for an *event* is a change in the state of the environment that:

1. Occurs at a specific instant in time;
2. Its occurrence is instantaneous;
3. Its occurrence changes the overall state of the environment in ways that could not be reasonably anticipated in an exact way.

Item 1 allows events to be processed sequentially if they are organized into a priority queue; item 2 makes the model simpler to reason since an event that may have a duration can be decomposed into start and finish events which are instantaneous; and item item 3 sets out a heuristic for defining events, so that they should not be trivial (in which case event-driven simulation provides no significant performance improvements), but should also not be too complex to a point where complicated algorithms to predict the future are required to determine or estimate when the next event of a given type will occur.

In the case of a traffic simulator, significant events can be, for instance:

- A traffic light changing phase;
- A vehicle entering a queue or moving to another road;
- A vehicle using a connection, therefore blocking lower-priority connections;
- A queue being popped.

4.4 Models of interest

In this section, a select group of models is analyzed to understand how other researchers designed their simulators, with the added interest of reusing some of the ideas in those models.

4.4.1 SUMO

Simulation of Urban MObility (SUMO) is an open-source, microscopic, multi-modal, time-driven traffic simulation package designed to handle large networks. It is mainly developed by the Institute of Transportation Systems at the German Aerospace Center (German Aerospace Center, 2023a). Its development started in 2000 (German Aerospace Center, 2023e), and its first public appearance was in Krajzewicz et al. (2002). The simulator is still actively maintained and developed with bug fixes and new features. Its open-source nature has originated an active community of contributors that help maintain the project and contribute to the SUMO ecosystem with included and external extensions (German Aerospace Center, 2023b).

Although SUMO is a multi-modal simulator, this section is only concerned with vehicle movement, since multi-modal simulation is outside the scope of this work.

Car-following According to Krajzewicz et al. (2002) and Maciejewski (2010), vehicle movement is modelled by SUMO using the car-following model proposed by Gipps (1986) as modified by Krauß (1998, p. 24), where at each time step the vehicle velocity is adapted to the velocity of the following vehicle. This velocity is called *safe velocity* v_{safe}, and is calculated from

$$v_{\text{safe}}(t) = v_l(t) + \frac{g(t) - v_l(t)\tau}{\frac{\overline{v}}{b(\overline{v})} + \tau} \tag{4.1}$$

where v_l is the velocity of the leading vehicle, g is the gap of the leading vehicle, τ is the driver's reaction time (usually 1 s) (Krajzewicz et al., 2002), v_f is the speed of the follower vehicle, and $\overline{v} = (v_l + v_f)/2$ is the average speed of the follower and leading vehicles (Krauß, 1998).

To bind the acceleration to the vehicle's abilities, Krajzewicz et al. (2002) calculates the *desired velocity* of a vehicle by the following formula:

$$v_{\text{des}}(t) = \min\{v_{\text{safe}}(t), v(t) + a, v_{\max}\} \tag{4.2}$$

where a is the vehicle maximum acceleration, and v_{max} is the vehicle maximum possible velocity.

Lane changing Vehicle routes in SUMO typically only contain the edges the vehicle traverses, so lane selection and changing are left to the simulator[1]. Erdmann (2014) outlines the three major types of lane changes that vehicles can perform in SUMO. When a vehicle must change lanes because the current lane does not connect to the next desired edge, it is called a *strategic lane change*, better known in the scientific community as a mandatory lane change. A vehicle may also perform a discretionary lane change, which SUMO distinguishes into two kinds. A vehicle changing lanes to move away from a lane to facilitate another vehicle to move into that lane is called *cooperative lane change*), and is particularly important in roundabouts, where strategic reasons would mandate that all vehicles use the outermost lane, but in reality, people move into the inner lanes of a roundabout to improve throughput (Erdmann, 2014). A vehicle may also change lanes to avoid a slow leader vehicle, or move into a faster-moving lane, in which case it is called a *tactic lane change* (Erdmann, 2014). Each different type of lane change comes with a different model to properly simulate real-world behavior.

[1] It is nevertheless possible to control lane changes manually using the SUMO TraCI interface (Wegener et al., 2008; German Aerospace Center, 2023d)

4.4.2 DynaMIT

DynaMIT (*Dynamic Network Assignment for the Management of Information to Travelers*) is a real-time, time-driven mesoscopic traffic assignment system that provides traffic predictions and travel guidance (Ben-Akiva et al., 1998). It was mainly developed between 1997 and 2002 at MIT Intelligent Transportation System Laboratory.

The road network is represented as a set of nodes corresponding to geographical locations, and links between nodes representing roads. DynaMIT considers three possible parts of a link: the moving part, the decelerating part, and the queuing part. A link is characterized by its capacity c (in vehicles per second), and length L.

Vehicles have an average length of l (including space separating vehicles). The position at the beginning of a link is $x = 0$, and the position at the end of the link (where vehicles leave the link) is $x = L$.

Queuing part In the queuing part, a queue emits vehicles at a rate of c vehicles per second. Therefore, a vehicle in the queue that has i vehicles in front of it has a position and speed

$$x = L - l \cdot i \tag{4.3}$$

$$v = l \cdot c \tag{4.4}$$

and the time it will take to leave the queue is

$$t = i/c \tag{4.5}$$

If a moving vehicle reaches the end of the queue at time t, its position is given by

$$q(t) = q_0 + l(ct - m) \tag{4.6}$$

where q_0 is the position of the end of the queue at time $t = 0$, and m is the number of moving vehicles between the considered vehicle and the end of the queue at time $t = 0$. This equation encodes that, by the time a certain vehicle arrives at the queue, the queue will have grown by as many vehicles were moving that are in front of the considered vehicle, and the queue will have shrunk by as many vehicles can be emitted by the queue at rate c.

Moving part In the moving part, vehicles travel with constant speed v_u. This speed depends on the congestion of the link, by the following formula:

$$v_u = v_0 \left[1 - \left(\frac{K}{K_{\text{jam}}} \right)^{\beta} \right]^{\alpha} \tag{4.7}$$

where v^0 is the free-flowing speed, K is the density, K_{jam} is the jam density, and α and β are empirically determined, dimensionless parameters. This formula resembles the Greenshields model

(Eq. (2.4)) but adds two extra parameters α and β. In this segment, the vehicle moves at a constant speed. Its position is

$$x = v_u t \tag{4.8}$$

and the time at which it will be at position x is

$$t = x/v_u \tag{4.9}$$

Deceleration part In the deceleration part, the vehicle is assumed to be initially traveling at speed v_u, and traveling at speed v_d when it arrives at the end of the queue, therefore going into the queuing part.

It is assumed that the vehicle decelerates linearly with respect to the distance to the queue. It starts decelerating at position $x = L - L_s$, and the position at the end of the queue is $x = L$. In the absence of a queue ($q(t) = L$), the speed x' of the vehicle abides by the following first-degree differential equation:

$$x'(x) = \lambda(x - L) + v_d \tag{4.10}$$

where λ is defined for convenience as

$$\lambda = \frac{v_d - v_u}{L_s} \tag{4.11}$$

Ben-Akiva et al. (2002) apparently set $L = L_S$ to mean the vehicle starts decelerating at the beginning of the link, and arrives at the end of the link at speed v_d. This yields the following solution for Eq. (4.10):

$$x = e^{\lambda t}\left(x_0 + \frac{v_u}{\lambda}\right) - \frac{v_u}{\lambda} \tag{4.12}$$

The author then assumes $v_d = 0$ and considers the existence of a queue. The vehicle now has to decelerate in a road segment of length $q(t)$ instead of L_S, so the solution becomes

$$x = e^{\lambda(t)t}\left(x_0 + \frac{v_u}{\lambda(t)}\right) - \frac{v_u}{\lambda(t)} \tag{4.13}$$

$$\lambda(t) = \frac{-v_u}{q(t)} \tag{4.14}$$

This solution results in the vehicle asymptotically approaching the position of the end of the queue (Fig. 4.2), so some criterion has to be decided to determine at which point the vehicle leaves the deceleration part and joins the queue. Ben-Akiva et al. (2002) suggests the criteria to be that the vehicle arrives at the queue at time t^*, defined as

$$t^* = \frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right)}{v_u + cl\ln\left(e^{-v_u/cl} + \varepsilon\right)} \tag{4.15}$$

where ε is an approximation parameter.

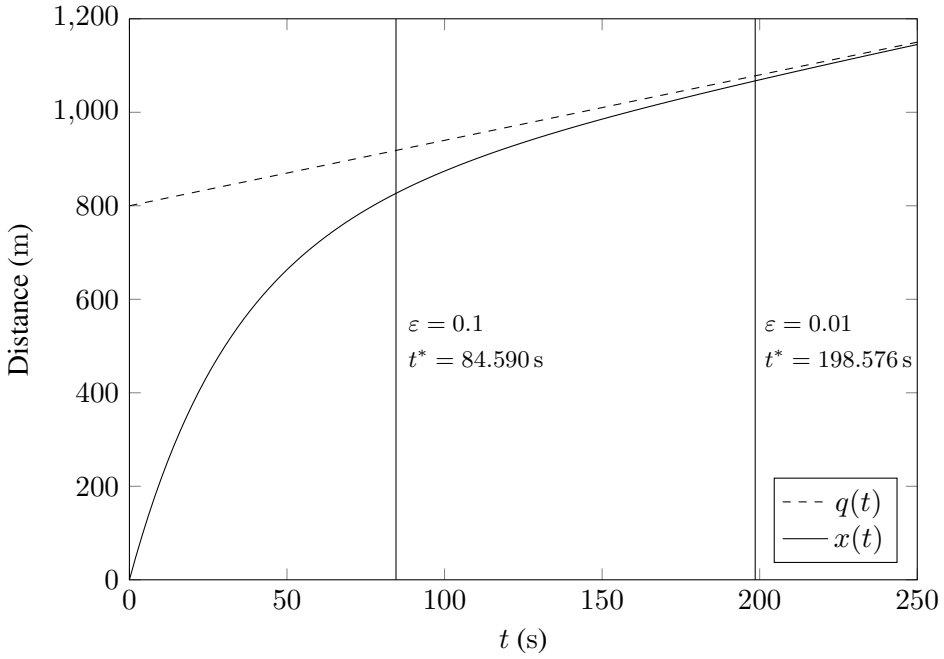


Figure 4.2: Position at time t in the presence of a queue, as used by DynaMIT. Values of t^* for $\varepsilon = 0.1$ and $\varepsilon = 0.01$ are marked with vertical lines. Adapted from Ben-Akiva et al. (2002).

While the full rationale of this approximation is not made available by Ben-Akiva et al. (2002), appendix C can be used to argue that the goal is to bound $x(t*) - q(t*)$ to a certain small fraction of $x_0 - q(t*)$, with ratio $r = e^{-v_u/cl} + \varepsilon$.

This argument is further strengthened by the assumption made by the authors that $v_u/cl \gg 1$, which is reasonable if we recall that v_u is the speed a vehicle enters the link and that $c \cdot l$ is the speed of the queue, which will be relatively smaller. With this simplification, the above-mentioned ratio becomes merely $r = \varepsilon$, and Ben-Akiva et al. (2002) obtain the following equations:

$$t^* = \frac{(ml - q_0)\ln\varepsilon}{v_u + cl\ln\varepsilon} \tag{4.16}$$

$$x(t) = \begin{cases} e^{\lambda(t)t}\left(x_0 + \dfrac{v_u}{\lambda(t)}\right) - \dfrac{v_u}{\lambda(t)} & : t < t^* \\ q_0 + l(ct - m) & : t \geq t^* \end{cases} \tag{4.17}$$

The vehicle reaches the end of the link at time

$$t = \frac{\dfrac{L - q_0}{l} + m}{c} \tag{4.18}$$

If $t < t^*$, the vehicle reaches the end of the link before reaching the end of the queue, meaning the queue is already dissipated.

Chapter 5

Assignment strategies

In this chapter, three assignment strategies are presented whose main goal is to change user behavior to bridge the gap between UE and SO. Particular emphasis is placed on reinforcement learning methods, namely on the concepts of Q-learning (Sec. 5.3.1) and difference reward (Sec. 5.3.2), which are the main concepts addressed in this work.

5.1 Tolls

The simplest way to modify user behavior is to apply tolls, which are artificially-added edge costs. Tolls do not affect travel time, but affect user behavior. Because tolls are not imposed to be only positive, the possibility of tolls being incentives (which are basically negative tolls) is also admissible. Congestion tolling is a form of Pigouvian taxes, named after Pigou (1920), which are levied on market activities that generate negative externalities[1], so that an undesirable market outcome is compensated by forcing the market to internalize external costs.

Here we present some simple algebraic manipulations with respect to how a UE can be transformed into a SO by adding tolls. The bijective functions are first restated for each case, where the UE cost function is modified to use a generalized cost $c_a(x)$ instead of a time $t_a(x)$:

$$f(\mathbf{x}) = \sum_a \int_0^{x_a} c_a(x)dx \qquad \text{(UE)} \tag{5.1}$$

$$g(\mathbf{x}) = \sum_a t_a(x_a)x_a \qquad \text{(SO)} \tag{5.2}$$

The portion that an arc a contributes to the cost functions is given by Eqs. (5.3) and (5.4),

$$f_a(x_a) = \int_0^{x_a} c_a(x)dx \qquad \text{(UE)} \tag{5.3}$$

$$g_a(x_a) = t_a(x_a)x_a \qquad \text{(SO)} \tag{5.4}$$

[1] An externality is "a cost or benefit arising from any activity which does not accrue to the person or organization carrying out the activity" (Black et al., 2017).

The toll T_a over arc a is then formally defined as the artificial cost charged to users so that a certain global goal is approached/reached (e.g., make user behavior converge with the SO). Therefore, $c_a(x) = T_a + t_a(x)$.

At first sight, the easiest way to make users converge to the SO is to set $f_a(x_a) = g_a(x_a)$. If this equality is differentiated with respect to x_a, it yields

$$c_a(x_a) = T_a + t_a(x_a) \tag{5.5}$$

$$T_a = x_a \cdot t'_a(x_a) \tag{5.6}$$

which not only confirms tolls can be imposed that force the UE to be equal to the SO, as also stated by Boyles et al. (2019, p. 166), but also arrives at the same conclusion as Beckmann et al. (1956, p. 87):

> Since average cost increases with flow, the cost of a vehicle mile, as averaged over all road users, is raised by an extra vehicle. This increase of the average cost level $[t'_a(x_a)]$ times the traffic flow $[x_a]$ equals the excess of social incremental cost over private incremental cost. If the road user were to bear his full share of cost, he would pay a tax or toll equal to this excess. This would make him realize the cost he causes to others and thus provide an incentive to keep social cost down by making the proper choices.

where private cost is the cost incurred by an individual vehicle, and social cost is the SO cost function. Notably, this statement conveys the same principle behind difference reward (Sec. 5.3.2).

Note that the value of x_a being used to calculate tolls $T_a(x_a)$ is not that of the original UE problem, because the tolls are calculated using the actual flow going through the arc, not the flow that would go through it if the original UE formulation was being used. But because the intended final flows configuration is the same as SO, it is only necessary to solve the SO and then calculate the tolls using the flows for the corresponding edges.

Regarding recent work, Ramos et al. (2020) applies reinforcement learning to a simple macroscopic traffic model using the BPR (1964) formula, associated with a tolling scheme based on Eq. (5.6), where agents learn by deciding between the K shortest paths. Recent work also focuses on more realistic alternatives to SO tolling, also known as *first-best* tolling, due to the impracticality of tolling all roads. These tolling schemes are divided into *second-best* tolling, where only a subset of links is subject to toll charge (Yang and Zhang, 2003); and *third-best* tolling, where other goals are set without consideration for the overall impact of tolls on other roads, of which *no-queue* tolling is an example, where only links with frequent queue formation are tolled (de Palma et al., 2005).

Greediness-based tolls Eqs. (2.3) and (5.6) mean that $T_a \geq 0$, and that $T_a = 0$ only if arc a has no flow; thus, as long as a has flow through it, $T_a > 0$. This means that every user will pay tolls, even if they are very small. Let us define T^π as the total toll value a user will pay if they

follow policy π[2]. T^{SO} is the minimum toll that a user pays for a certain OD if 100% altruistic, if we consider only rewards of the type $R = -t_a(x_a) - \alpha T_a$ (where $0 \leq \alpha \leq 1$).

From a mathematical viewpoint, tolls are a useful tool to influence how people choose routes. However, the societal interpretation of tolls is *punitive*, therefore the starting point should be that a 100% altruistic user (in the sense they value the SO over a greedy strategy) should pay no tolls, and that users should be charged increasing amounts of tolls in proportion to how greedily they made their route decisions. A simple change to this toll system that would make it fairer would be to charge $\max\{0, T^{\pi} - T^{\text{SO}}\}$ instead of just T^{π}.

5.2 Auctions and negotiation

According to Irani and Leung (1996), most work on scheduling with conflicts was based on the *Dining Philosophers Problem*[3], with their work being the first where the concept of agents using resources with conflicts was explicitly formulated for traffic, where the limited resources are intersections.

The basis for current methods on reservations and interactions between vehicle and intersection agents was presented in Dresner and Stone (2004). This concept was further expanded by Vasirani (2009) and Vasirani and Ossowski (2012) through a distributed, market-inspired mechanism based on combinatorial auctions, where intersection agents participate in the market as suppliers and vehicle agents as buyers. Schepperle and Böhm (2007) propose the valuation-aware variant *Initial Time-Slot Auction*, and in a related paper Schepperle et al. (2008) propose *Time-Slot Exchange*, an optional mechanism through which vehicle agents can bilaterally negotiate slot exchanges after obtaining an initial slot they are not satisfied with. More recently, Rahimi-Farahani et al. (2019) proposed a *forced-node* route guidance system, where a selected set of intersection and roundabout agents makes binding decisions on which roads vehicles should use, so as to avoid congestion, and Elimadi et al. (2021) proposed an *Agent Node-Centric* approach where vehicle agents book times for crossing intersections while intersection agents adjust booking times according to demand.

5.3 Reinforcement learning

Reinforcement learning (RL) is one of the three main forms of machine learning (along with supervised and unsupervised learning). In a RL problem, there is an *environment* that can either represent the real world (if simulated), or it can be the real world. This environment is populated

[2]Lending from reinforcement learning. Agents always choose a solution that minimizes their own cost function. Policy UE is 100% greedy and only takes time into account, with reward $R = -t_a(x_a)$; policy SO is one that will originate a system optimum if all agents use it, and can be considered for instance a policy that takes tolls into account. Its reward is $R = -t_a(x_a) - T_a$.

[3]In this problem, five philosophers sit at a round table to eat spaghetti. Between each philosopher, there is one fork on the table. The philosophers are either thinking or eating; when thinking, they are not using any fork; to eat, a philosopher needs the two forks on his/her side. The problem is to design an algorithm such that no philosopher starves. It was originally formulated by Dijkstra in 1965 (Dijkstra, 1987) and adapted by Hoare (1985).

with one or more agents that can make decisions with respect to the actions they take in each state. Agents learn properties about the environment by interacting with it, which makes this paradigm abstract enough to be applied to a vast set of problems. It is further generalized that, by transitioning from one state to another, an agent receives a *reward* for doing it. Thus, an agent should learn enough about the environment to maximize its total rewards, similar to how animals can be taught tricks by positive reinforcement training in the form of treats.

Exploration and exploitation An agent that only picks actions that it knows very well is prone to be stuck in a local minimum due to its greediness, since it does not have the required *curiosity* to explore the environment to possibly find a better policy. On the other hand, an agent that is purely curious will explore many alternative paths, but will never use the precious information it collected so far, so it lacks in *pragmatism*. This is what is called the exploration-exploitation tradeoff, since the agent must balance how curious it is–to escape local minima–with how pragmatic it is–how frequently it only uses the information it already has, and picks the best option (Sutton and Barto, 2020).

Model-based and model-free approaches A particular distinction can be made between *model-based* and *model-free* algorithms. An agent following a model-based algorithm generally aims at understanding the environment by building a model of the environment, so that it may predict the reward of future actions. On the other hand, an agent following a model-free algorithm cannot estimate or predict future rewards based on the problem structure, so it must perform the actions to learn what the rewards are.

This does not mean that model-free algorithms do not use a model in a loose sense. Model-free algorithms often have memory of past decisions and rewards, and can derive a model from it. The main difference is that model-based algorithms try to understand the environment in a more rational way and apply some sort of *planning*, while model-free algorithms are explicitly trial-and-error methods (Sutton and Barto, 2020, p. 7).

As a simplification, consider that an agent is placed in a certain environment. It is often possible for the environment to provide all possible instantaneous rewards for all possible states, although usually very costly. The difference between model-based and model-free algorithms is that, in model-based algorithms, many possible reward estimates are considered, while model-free algorithms prefer to build a model from memory by testing single samples of (state, action) pairs.

5.3.1 Q-learning

Q-learning is a model-free RL algorithm proposed by Watkins (1989). Provided with a function that maps a (state, action) pair to a reward, it finds an optimal policy that maximizes total reward. Q-learning draws many parallels from the generic paradigm of Dynamic Programming (DP) in the sense that it maintains a data structure that is successively modified to compute optimal policies (Sutton and Barto, 2020). It is however different from DP in that it relaxes some constraints regarding optimal substructure and reward functions. This is also why DP algorithms find the

optimal policy in usually finite time spans, while Q-learning algorithms are said to iteratively converge towards an optimal policy.

Q-learning algorithms work by maintaining a structure, or a function $Q : S \times A \to \mathbb{R}$, that is built by the Q-learner from memory of past decisions and is used to decide the next course of action. This function $Q(a, s)$ is used to estimate the quality of the decision for an action $a \in A$ given that the agent is currently at state $s \in S$. The agent's learning process consists of progressively updating the Q function according to the rewards that the agent obtains from the actions it takes.

One of the fundamental principles of Q-learning is that the Q function can express not only instantaneous rewards but also future rewards. This is done by updating the Q with the following formula:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \lambda \cdot \max_a Q(s_{t+1}, a)) \tag{5.7}$$

where α is the learning rate, r_t is the reward for moving from state s_t to s_{t+1}, $\max_a Q(s_{t+1}, a)$ is an estimate of the maximum possible future reward that can be obtained from transitioning to s_{t+1}, and λ is the *discount factor*, which represents how important future rewards are.

In generic terms, the discount factor can also be interpreted as a learning rate, although it leads to different interpretations. α is part of the exponential convergence process so it does not necessarily describe the way the agent acts (or if you will, it does not describe its *personality*), as it is merely a parameter that is common in learning algorithms. However, λ describes the way the agent values future rewards, so it can be said to be a parameter of the agent's personality. λ often varies between 0 (where the agent is said to be *myopic*, since it only takes into account instant rewards) and 1 (where the agent is said to strongly strive for a long-term high reward); the first impression is that higher values of λ lead to a smarter learner, and therefore to a faster learning process, but it most often only results in divergent action values. Instead, much like the learning rate α, in a series of experiments these two parameters should be set to small values, and then progressively increased to improve the learning process while avoiding divergence. Q-learning algorithms are guaranteed to arrive at an optimal policy in a finite Markov decision process, so the values of these two parameters are only relevant in determining how fast learners converge to an optimal policy.

The generic formulation of the Q-learning algorithm leaves the exploration vs. exploitation problem as an open question. A particular type of Q-learning algorithms that addresses this issue is called ε-greedy, because they exploit the action with the largest expected reward with a usually large probability $1 - \varepsilon$, but sometimes explore the environment with probability ε to try to find better alternatives (and as such, escape local minimums) by selecting a random action. In such a scenario, the agent is usually trained over multiple episodes where it maintains memory of previous episodes, and the value of ε is progressively reduced so that the agent initially prefers exploration, but tends to prefer exploitation as it learns more about the environment and becomes more confident about its own reward estimates that are provided by Q.

5.3.2 Difference reward

In theory, it is possible to use the global cost function as the private reward function for all agents, in which case agents will strive to optimize the global state. This however is not very realistic, since users cannot often or easily determine the global cost at any point.

Difference reward is a technique introduced by Tumer and Wolpert (2004) to model a reward function that is both sensitive to agent decisions as well as taking into account the global cost function. A difference reward is the reward that the agent receives as a result of it being a part of the system; so difference reward basically translates how much the global utility increased by adding one more agent. It thus translates the difference in utilities if we consider that agent to be present or missing from the scenario. Thus, the agent will indirectly try to minimize their negative impact on global cost.

More formally, a difference reward, or difference utility DU, is defined by Tumer and Wolpert (2004, p. 26) for agent η as a utility of the form

$$DU(z) = G(z) - \Gamma(f(z)) \tag{5.8}$$

where G is the global utility function, and $\Gamma(f)$ does not depend on the decisions of agent η. Albeit a good generic framework, the obvious question is, then, what is $\Gamma(f)$ supposed to represent? Tumer and Agogino (2007) suggest a well-defined value for Γ that makes Eq. (5.8) easier to understand, with DU expressed by:

$$D_i \equiv G(z) - G(z - z_i + c_i) \tag{5.9}$$

where G is the global utility function, z is a global solution, z_i is the action of agent i, and c_i is a fixed constant. Under this notation system, $z - z_i + c_i$ is the global state if the action of agent i is replaced by a constant c_i; this constant is typically a neutral element, so that $G(z)$ reflects the global utility including the actions of agent i, and $G(z - z_i + c_i)$ represents the global utility if the states z_i of agent i were replaced by constants c_i; in most scenarios, c_i is set to 0 to mean that agent i was completely removed from the system.

This reward function was created so agents could rapidly converge toward a global optimum. The intuitive reason behind its effectiveness is that, with this reward function, the agent is aware of its actions' influence over other agents in the system (Grunitzki et al., 2014) and will aim to maximize its positive impact in the global utility function; or otherwise, if rewards are negative, the agent will seek to minimize its negative impact in the system. This reasoning draws many parallels with the concept of tolls mentioned in Sec. 5.1.

According to Tumer and Wolpert (2004, p. 25), the quality of a reward function g_i for agent i on a system with a global cost function G depends on its *factoredness* and *learnability*.

Factoredness is associated with the alignment between the reward function and the global utility; i.e., if the designed reward function g frequently increases when G also increases, then the

reward function has a high degree of factoredness. Agogino and Tumer (2005) further provides a formula for the degree of factoredness $\mathcal{F}_{g_i}$ at a state z, which is an adaptation of the covariance between g and G:

$$\mathcal{F}_{g_i} = \frac{\sum_{z'} u[(g_i(z) - g_i(z'))(G(z) - G(z')]}{\sum_{z'} 1} \tag{5.10}$$

where z and z' only differ in the states of agent i (so, the possible values of z' are all the alternatives that agent i could have selected), u is the unit step function, and $\sum_{z'} 1$ is the number of possible values of z'. This means that $\mathcal{F}_{g_i}$ is the percentage of cases in which g and G agree on both increasing or decreasing. This property is important, because a high factoredness causes the agent behavior to be aligned with the global goal of minimizing G; when $\mathcal{F}_{g_i} = 1$, we have perfect factoredness, where the agent reward is perfectly aligned with the global goal: $\forall z, z', g_i(z) \geq g_i(z') \iff G_i(z) \geq G_i(z')$.

Learnability is associated with the phenomenon of t. The point learnability $L_{g_i}(z')$ of reward function g_i between states z and z' (where z and z' are arbitrary) is defined as the ratio of the change in g_i due to a change in the states of agent i over the change in g_i due to a change in the states of other agents:

$$L_{g_i}(z') = \frac{|g_i(z) - g_i(z - z_i + z'_i)|}{|g_i(z) - g_i(z' - z'_i + z_i)|} \tag{5.11}$$

The learnability $L_{g_i}(z')$ of reward function g_i at state z is the average of all point learnabilities:

$$L_{g_i} = \frac{\sum_{z'} L_{g_i}(z')}{\sum_{z'} 1} \tag{5.12}$$

Eq. (5.11) is akin to a signal-to-noise ratio: the numerator is the desired signal, corresponding to the measurable influence of agent i's decisions, that needs to be kept as high as possible; and the denominator is the noise from changing other vehicles' states, which should be kept as low as possible since the goal of g_i is to measure the quality of agent i's decisions, not other vehicles' decisions. Hence why Grunitzki et al. (2014) remark that high learnability corresponds to the agent getting a "cleaner" signal. Learnability is also related to agent i being able to understand, in a situation where G increases, how much of that increase in G is owed to that agent's decisions.

These two concepts are illustrated by Tumer and Wolpert (2004) through a "team game" where private utility functions (i.e., g_i) are set to the global cost G; this is equivalent to a game where players are completely aligned with the goal of scoring the most points against the other team(s), but team members have difficulties discerning how their individual performance and decisions impact the game outcome. In this situation, factoredness is very high because all players can clearly see the outcome of the game and strive to improve it, but learnability is very low because players cannot easily discern how a change in their individual behaviors may lead to a better score.

Part II

Methodology

Chapter 6

Data

In this chapter, the data that were used in this project are presented, as well as the methodology used to obtain them. All the data used in this work was collected and processed in a semi-automatic way, Fig. 6.1. The data used in this project is predominantly in XML format, and was loaded into the simulator using C++ library RapidXML (Bellini and Inconis, 2021).

6.1 Traffic network data

The traffic network under analysis is the traffic network of the city of Porto. It is geographically bounded by the borders of the municipality of Porto, plus complementary roads and major highways that cross municipality borders. This data was obtained from OpenStreetMap (2023), using JOSM (Scholz and Stöcker, 2023) to download a local working version, and Overpass API (Olbricht, 2023) to automatically download data in the pipeline.

All available OSM data is downloaded for the bounding box with coordinates $(41.05°, -8.70°)$ and $(41.20°, -8.50°)$ (latitude, longitude respectively), yielding a total of 212.2 MB raw OSM data including some versioning metadata (`porto-unbounded.osm`). This data was then filtered so that a OSM file was obtained containing all data inside the municipality of Porto plus motorways in the vicinity (`porto.osm`), totaling at 100.8 MB. Finally, the OSM file is converted to the SUMO format and stored as `porto.net.xml`, with 18.1 MB. The network file includes all roads traversable by motorized vehicles, except for `living_street` and `service`.

Numerous manual fixes were necessary for the simulation results to be reliable. The majority of fixes was made directly to the public OSM dataset, including road classification, direction, number of lanes, as well as turning lanes, arrows, and restrictions. Many traffic lights were added by noting areas of high congestion in the traffic simulations (which were often a sign that traffic lights were not present in the dataset) and cross-checking those situations with the traffic lights dataset from the municipality of Porto (CM Porto, 2023) and local knowledge. The fixes that could not be made in OSM were made in the resulting SUMO files, and the diff files corresponding to those fixes were obtained (`diff.*.xml`), so that those diff files could be later applied to SUMO network files obtained from new, fixed OSM files.

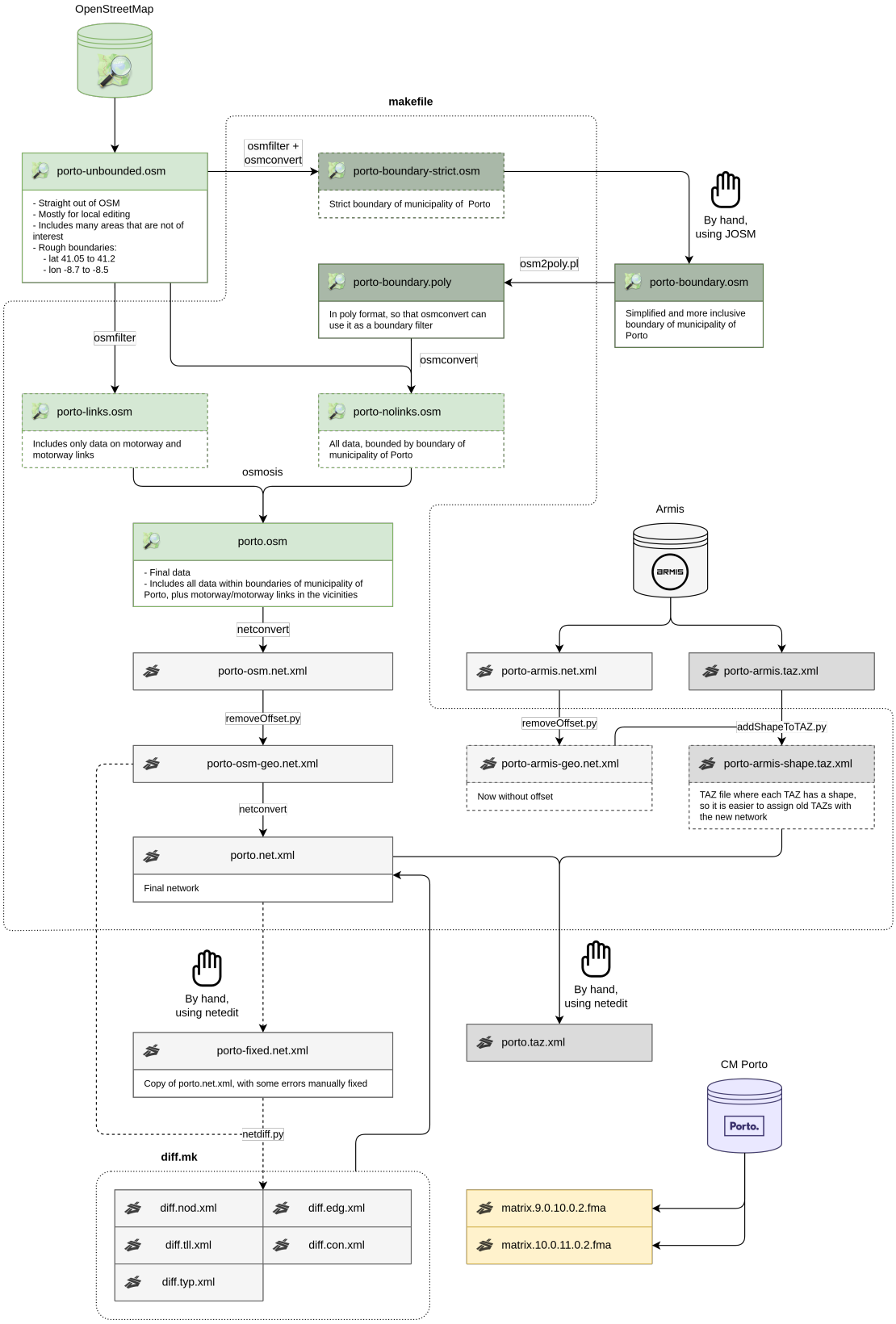


Figure 6.1: Data retrieval and processing pipeline.

The traffic network as of 2/Jul/2023 (see Fig. 6.2) contained:

- 4690 junctions;
- 8749 edges (average of 1.865 edges per junction);
- 11 544 lanes (average of 1.319 lanes per edge);
- 16 823 connections (average of 1.457 connections per lane);
- 404 traffic light programs and 54 roundabouts.

Note that, as per the SUMO network format, a traffic light program usually controls one junction but may simultaneously control many junctions that are geographically near each other.

Objects in the network file `porto.net.xml` can be made to correspond to the original OSM objects as a result of their IDs being largely derived from OSM IDs, so this network file can be considered to use unique and globally-recognizable IDs.

Figure 6.2: Traffic network of the municipality of Porto with only relevant road classes. Rendered with JOSM (Scholz and Stöcker, 2023).

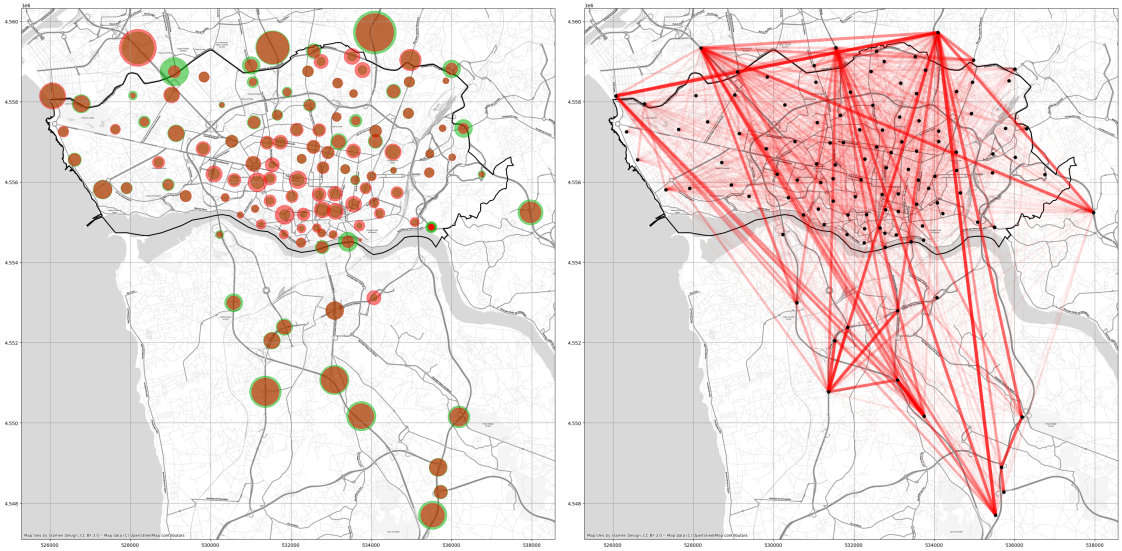
6.2 Demand and TAZ data

The TAZ and demand data were kindly provided by the municipality of Porto to the DynamiCITY project. The data resulted from a study made for the municipality of Porto by Armis Group, a company specialized in Intelligent Transport Systems (ITS) based in Porto. This dataset includes:

- A SUMO network file, hereby named `porto-armis.net.xml`, that was used in the Armis study.

- A SUMO TAZ file, hereby named `porto-armis.taz.xml`, where TAZ edges are identified with respect to network file `porto-armis.net.xml`.
- Two files of demand data corresponding to demand estimates between 09:00-10:00 and 10:00-11:00, hereby named `matrix.9.0.10.0.2.armis.fma` and `matrix.10.0.11.0.2.armis.fma` respectively. The demand data files are in VISUM (PTV, 1992) O-format (PTV, 2001, p. 5-2) to represent OD matrices, with estimated countings of vehicles that travel between each pair of TAZs over the 1 h period they refer to. The TAZs are identified with respect to TAZ file `porto-armis.taz.xml`.

The TAZ file identifies 119 TAZs in the city of Porto and its vicinity, and also identifies the edges of `porto-armis.net.xml` that are part of each TAZ; in total, there are 1086 edges from `porto-armis.net.xml` that are part of TAZs, an average of 9.126 edges per TAZ. From visual inspection, it was concluded that each TAZ corresponds to a specific source/sink point in the network since each TAZ typically includes all the edges around a single junction.



(a) TAZs mapped by the number of departures (green) and arrivals (red). (b) Overview of major vehicle flows between TAZs.

Figure 6.3: Exploratory visualization of demand data from 09:00-10:00.

Files `matrix.9.0.10.0.2.armis.fma` and `matrix.10.0.11.0.2.armis.fma` contain 102 731 and 96 431 OD pairs respectively. Attending to the time periods the matrices refer to, they aim at representing the mid- and late-morning rush times, respectively.

Since the roads and IDs used in `porto-armis.net.xml` are different from those in `porto.net.xml`, it was necessary to manually identify the edges in `porto.net.xml` that corresponded to edges in TAZs in `porto-armis.net.xml`, and create a new SUMO TAZ file `porto.taz.xml` that uses the edge IDs in `porto.net.xml`.

Chapter 7

Static simulator

It was determined that the first step would be to develop a simple static model, essentially based on the classical formulation of TAP as an optimization problem. The criteria that this static model must meet are that (1) it must be a classical model, in that vehicle movements are represented by flows, and time is not a relevant variable; (2) it must be a convex problem with a unique solution, so it can be solved using FW; and (3) proper impedance functions must be developed so that the cost of a road depends on its speed limit, number of lanes, and vehicle flow, and that signalized and unsignalized intersections impose capacity and time penalties upon users.

7.1 Intersections

Edges connecting two roads in a junction can also use the BPR model, but the definition of free-flow time t_0 has to be adapted since this connection edge is merely the connection between two roads, so it does not have a geometry, and as such it cannot be said that t_0 is the time needed to cross it. Instead, this edge was interpreted as the physical representation of the lanes of the start road that lead into another road. Therefore, the delay in this edge should adequately represent the time it takes for a vehicle to perform the movement it describes, plus delays associated with red traffic light signals and yield rules.

7.2 Signalized intersections

For connections in signalized intersections, Eqs. (2.23) and (2.30) were used to estimate the capacity and free-flow delay of junction-internal arcs:

$$c = \sum_{\kappa \in K_a} s_\kappa \frac{g_\kappa}{C} \tag{7.1}$$

$$t_0 = \frac{1}{|K_a|} \sum_{\kappa \in K_a} \frac{r^2}{2C} \tag{7.2}$$

where K_a is the set of connections leaving arc a. The practical meaning of these equations is that the capacity of a junction-internal arc is the sum of the capacities of each individual connection κ that is grouped into a, and that the free-flow delay is the average delay of all connections κ that are grouped into a.

It was decided not to use Eq. (2.31) because of its asymptotic behavior at $\lambda = \mu$ and $\lambda = \nu$, which is contrary to the fundamental principle of simplicity that guided the design. As such, it was opted to use the BPR model similarly to what is done for other edges.

7.3 Unsignalized intersections

For simplicity of language, consider that, if connection κ' conflicts with connection κ, and κ' has higher priority than κ (i.e., if vehicles are using connections κ and κ' simultaneously, those using κ' will pass before those in κ, and those vehicles using κ will only traverse the junction if there are no vehicles using κ'), then we say that κ' *rules* κ.

In unsignalized intersections, the deductions made about Eq. (2.38) and the conclusions in appendix B were used to obtain the following basic relations:

$$\mathrm{E}[W] = \frac{1}{\lambda}(e^{\lambda t_{cr}} - 1) - t_{cr} \tag{7.3}$$

$$c = \frac{1}{\mathrm{E}[W]} \tag{7.4}$$

$$t_0 = \frac{1}{c} \tag{7.5}$$

The issue is that we must now formulate these equations into a form that is suitable to calculate the junction-internal arcs' capacities and free-flow delays.

The solution is, for each junction-internal arc a, to consider its connections κ. For each connection κ, all other connections κ' that rule κ are determined, and the share of flow of other junction-internal arcs that are expected to go through those κ' connections is found. The shares of flow can be estimated by considering the total number k' of connections in a junction-internal arc a', and then claim the average flow through an individual connection of that same arc a' is $1/k$ of the total flow through a'. Then, for each arc a, we store for each connection κ, a list of pairs (a', k), where a' is a junction-internal arc distinct from a that contains at least one connection κ' that rules κ, and the corresponding share k of flow through a' that conflicts with κ. This method allows us to calculate the capacities of all connections κ that are grouped in a junction-internal arc a, given that we have the current flows through all connections that rule κ. We now have to find the total capacity of arc a from these individual capacities.

If time delays were being used, it would now be necessary to group them together into a unified delay, so the total capacity could then be calculated. The problem is that there is no intuitive way to group delays from different connections of the same arc together. The advantage of using capacities in this context is that capacities are additive, unlike delays of parallel connections. This means it is only necessary to add up capacities to obtain a total capacity.

Chapter 8

Dynamic simulator

This chapter describes DynamiNATOR, a mesoscopic, event-driven dynamic traffic simulator. This simulator implements congestion effects, stopping at traffic lights, right-of-way rules, queues, and queue spill-back. To contextualize the developed simulator, Fig. 8.1 shows an overview of the data types that DynamiNATOR consumes and produces, and how that data is obtained and visualized, respectively. It receives a network description, composed of files `*.net.xml` and `*.taz.xml` describing the network and the TAZs, both in SUMO format (German Aerospace Center, 2023c,f). Demand is provided in a `*.fma` file, in VISUM O-format (PTV, 2001, p. 5-2).

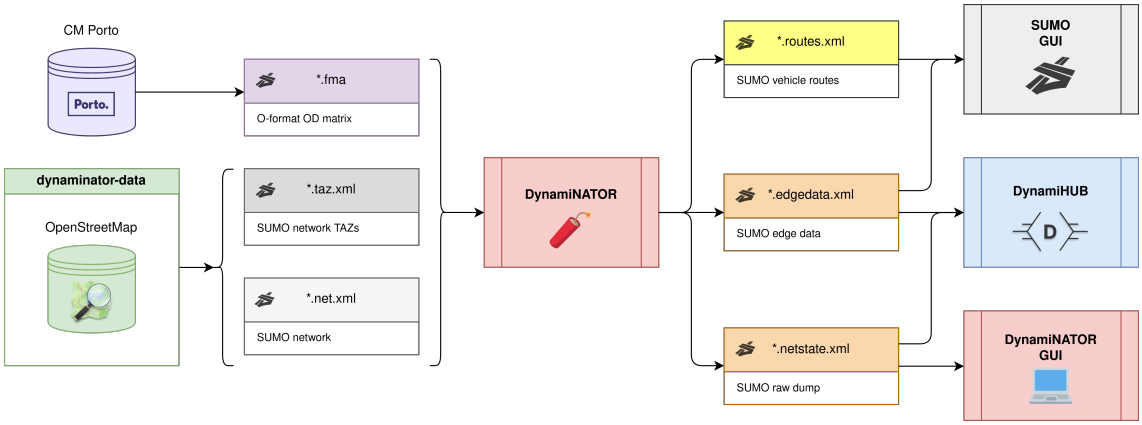


Figure 8.1: Data flow diagram of DynamiNATOR. The outputs can be visualized using SUMO GUI (Krajzewicz et al., 2002), DynamiHUB (Ribeiro and Sousa, 2023) or DynamiNATOR GUI.

This chapter is essentially focused on the modeling aspects of DynamiNATOR, and starts by describing the model parameters in Sec. 8.1. A general description of the environment is presented in Sec. 8.2, followed by the description of the considered events in Sec. 8.3 that is closer to the computational realization of this model. Sec. 8.4 and Sec. 8.5 describe the way vehicles are introduced (or *spawned*) into the network, and how the vehicle interacts with the environment through its corresponding policy. Finally, Sec. 8.6 shows the ways DynamiNATOR output can be visualized, and Sec. 8.7 is a brief reflection on how parallelism could be leveraged in this model.

8.1 Parameters

In this section, the parameters of the model are described, and their default values are shown.

Vehicle gap Defined as the sum of the vehicle length and the distance between the front of the vehicle and the back of the next vehicle in a traffic jam. The average vehicle gap l_v is thus the average value of the gap between all vehicles. This is a convenient way to condense average vehicle length and distance between vehicles in a traffic jam, so as to reflect the effective space that vehicles take up in a queue. The default value of the average vehicle gap was determined from the estimates in appendix D.1 to be $l_v = 6.52\,\mathrm{m}$.

Queue dissipation speed Queue dissipation speed v_q is the average speed of all vehicles in a dissipating queue. Formally, it is defined as the speed at which a platoon of vehicles where consecutive vehicles are separated by the lowest possible gap l_v would have to travel through a junction to achieve the maximum capacity of a junction in the real world. The default value of the queue dissipation speed was determined from the estimates in appendix D.2 to be $v_q = 3.44\,\mathrm{m/s}$.

Queue period Queue period T_q is the average time a vehicle takes to leave a queue. It can be calculated by dividing the average vehicle gap by the queue dissipation speed: $T_q = l_v / v_q$. Using the default values already mentioned, this yields a queue period of $T_q = 1.895\,\mathrm{s}$.

Critical gap Critical gap t_{cr} is the minimum headway between two consecutive vehicles that a less-priority vehicle can accept when joining a higher-priority flow. Ashalatha and Chandra (2011) point to typical values of the critical gap between 2 s to 3 s, so it was decided to use $t_{cr} = 2.5\,\mathrm{s}$.

8.2 Environment

In this section, the description of the environment is laid out. It goes into detail on the rules of vehicle motion and junction operation.

The atomic unit of road in this model is the *lane*, and the atomic unit of traffic flow is the *vehicle*. All variables and parameters are described in SI units. A position in the environment is described as a lane-offset pair, where the offset is measured from the beginning of said lane; the offset may never be less than zero, or greater than the length of the lane.

A vehicle is described by its position, speed, and state. Its state is one of either:

- `MOVING`: the vehicle is moving under uncongested flow;
- `STOPPED`: the vehicle is stopped at a queue or is part of a queue that is dissipating;
- `LEFT`: auxiliary state, indicating the vehicle left the environment; allows collecting statistics at the end of the simulation about vehicles that left the simulation mid-way through.

For simplicity, it was assumed with respect to traffic lights that there is no yellow traffic light phase, so a connection can be in one of only two states: green (move) or red (stop).

8.2.1 Uncongested flow

The most basic scenario is uncongested flow. Under uncongested flow, vehicles flow at a constant speed that depends on the number of vehicles in the lane. Since vehicle density K can be readily calculated from the number of vehicles in the lane and the length of the lane, Greenshields (1935) model was adopted for its simplicity. Eq. (2.4) is reproduced here for convenience:

$$v = v_0 \left(1 - \frac{K}{K_{\text{jam}}}\right)$$

The speed that a vehicle adopts when traversing a lane is constant and determined when the vehicle enters the lane. This means that any vehicles entering a lane after that vehicle have no impact on its speed.

This model comes with some obvious issues. First off, K may occasionally rise above K_{jam}, so it was decided the uncongested flow speed should always be at least the queue dissipation speed v_q. Secondly, since vehicles may be leaving a potentially very long edge at a higher rate than vehicles are arriving, the uncongested flow speed will tend to increase, and as such, a vehicle j entering a lane after vehicle i will have a greater speed. Over great distances, vehicle j may "overtake" vehicle i owed to its higher speed, although our model does not contemplate overtaking. We deemed this issue irrelevant, so vehicles in uncongested flow in a lane are stored in a set without any specific order.

Another issue with this uncongested flow is that instantaneous acceleration and deceleration are not realistic, even if this matter does not have any significant impact on travel times. An exploration into a more realistic model for movement under an uncongested flow regime is available in appendix E, although it was not implemented after considering the balance between its complexity and the associated benefits.

8.2.2 Queues

A traffic queue can occur if the traffic light controlling a connection is red, or if a vehicle must yield to another vehicle according to right-of-way rules. A queue is represented as a FIFO[1], and may only form at the end of a lane.

The offset of vehicle i that is in a queue is given by

$$x_i = L - n_i^q \cdot l_v \tag{8.1}$$

where L is the edge length, l_v is the average gap, and n_i^q is the ordinal position of the vehicle in the queue, where $n_i^q = 0$ would indicate that vehicle i is at the front of the queue.

[1] To avoid confusion, the term *queue* is used to mean a traffic queue. We refer to the queue data structure as FIFO.

Similarly, the length q and the position x_q of the end of a queue of size Q are given by

$$q = Q \cdot l_v \tag{8.2}$$

$$x_q = L - q \tag{8.3}$$

If the obstacle that created the queue disappears (e.g., if traffic lights allow passing or if a gap in a higher-priority connection occurs), the queue may start dissipating at constant speed v_q. Consequently, if the queue is dissipating, all vehicles in the queue are moving at speed v_q, otherwise they are stopped.

The *capacity* of a queue is the number of vehicles that can fit in that queue, and it can be calculated by dividing the lane length by the average gap $C_q = \lfloor L/l_v \rfloor$. A queue is said to be *full* iff $Q \geq C_q$. Unfortunately, this definition of queue fullness creates an edge case where a very short lane can never have a queue, so to mitigate this issue the definition of queue capacity was modified to use the ceiling function so as to guarantee that all edges have a capacity of at least one vehicle, Eq. (8.4).

$$C_q = \left\lceil \frac{L}{l_v} \right\rceil \tag{8.4}$$

In the developed model, vehicles are considered to instantly brake to enter a queue; similarly, vehicles in a queue are considered to instantly accelerate up to v_q when the queue starts dissipating. Queue spill-back was also implemented so that if a vehicle tries to transition to a lane with a full queue, the queue starts spilling back to the current lane.

Because it may be important to know the position of a vehicle inside the queue, a special FIFO was implemented that allows finding the order of an element inside the queue. This structure was named an *order-statistic queue*; refer to appendix F for more information on this data structure.

8.2.3 Prioritized intersections

If a vehicle in a low-priority connection approaches a prioritized intersection, it must yield if there are any vehicles in a higher-priority connection. In practice, drivers decide to accept a gap in higher-priority traffic by looking at oncoming traffic and estimating how much time an approaching vehicle will take until it reaches the junction, and if the gap is deemed long enough, the driver accepts the gap and starts the road-changing maneuver. This process can be simplified into deciding whether the gap is greater than the critical gap t_{cr} that the average driver perceives as acceptable between higher-priority vehicles.

But abiding by principle 3 laid out for event-driven simulation (Sec. 4.3), implementing such logic would be quite complex since it could require looking onto possibly numerous incoming lanes and the roads before those, as well as anticipating agent decisions. Note that this also corresponds to some of the tasks that drivers have to perform in the real world: anticipating possible danger and reacting accordingly; hence the importance of drivers being predictable on the road.

Thus, a different method was developed to enforce a penalty on vehicles using low-priority connections while avoiding such a complex process. Under this method, each connection c keeps

a timestamp t_c of the last time it was used. A vehicle trying to use a low-priority connection must wait at least until a time interval of T_q has passed since the most recent usage of a connection with higher priority; in other words, if a vehicle is trying to use a connection c, and R_c is the set of connections that c responds to, the vehicle must wait until at least

$$t = T_q + \max_{c' \in R_c} t_{c'} \tag{8.5}$$

When compared with the real-world process of analyzing oncoming traffic, this method gives the same expected penalty; but it operates differently by looking at the past, which is simpler than predicting the future.

8.2.4 Lane changing

Since the atomic unit of road in this model is the lane, it is inevitable that lane changing must be considered. When a vehicle enters a lane, it is allowed to change to any other lane of the same edge, Fig. 8.2. A vehicle is never prohibited from changing lanes, but if the destination lane is full, the vehicle is enqueued at the previous junction.

If the destination lane is not full, the vehicle changes lanes and should incur a penalty that depends on the vehicle density at the destination lane. But since this penalty would not significantly impact travel time significantly (besides, penalties are already incurred at traffic lights, prioritized intersections, and high-density uncongested lanes), it was decided not to implement this lane-changing penalty.

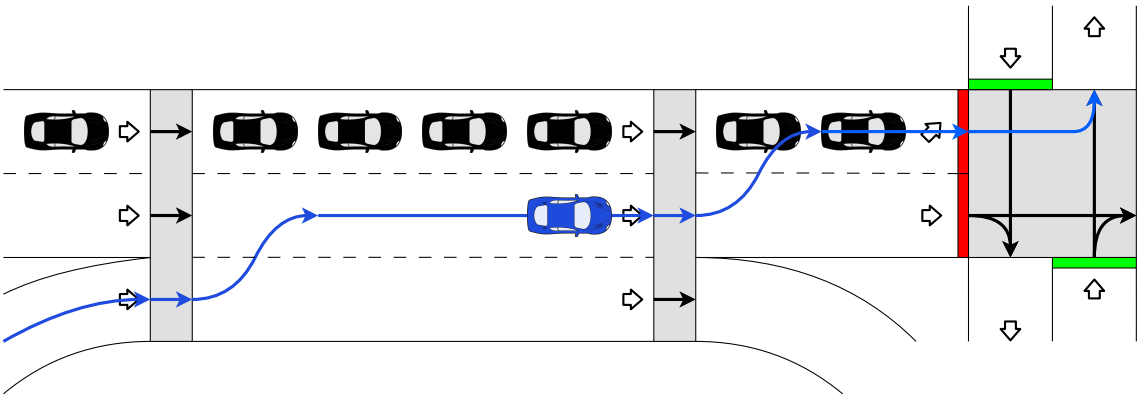


Figure 8.2: Queue spill-back when destination lane is full. Grey areas are junctions. The intended path for the blue car is marked with blue arrows. Although the connection before the traffic lights leads to a non-full lane, after that the blue vehicle will move to the left lane, which is full; so the blue vehicle is enqueued at the previous lane.

8.3 Events

When implementing the events needed for the simulator to function, the main goal was that the number of events that have to be processed is minimal, and that no task is performed twice. Therefore, the essential principle that guided event design is that any object either (1) has the responsibility of taking care of itself; or (2) it is delegating responsibility over its workings to another object.

This principle is made clearer by the way vehicle-queue interaction was designed. When a vehicle is moving under uncongested flow it is responsible for its own movement, and when leaving a lane it is responsible for deciding where to go next. On the other hand, when a vehicle enters a queue, the queue becomes responsible for emitting the vehicle when the queue is dissipating, as well as for pulling vehicles from spill-back queues (i.e., popping spill-back queues). The main events of this simulator are shown in Fig. 8.3, and are described in more detail as follows.

EventSpawnVehicle Spawns vehicle into the environment in the chosen origin lane. If the origin lane is full, the new vehicle is added to a separate FIFO of uninstantiated vehicles. The time these vehicles spend uninstantiated does not count towards total travel time.

EventMoveVehicle Updates the offset of the vehicle based on its previous offset and speed. If the vehicle is in a queue, it updates the vehicle offset based on its position in the queue, which can be obtained because traffic queues are implemented in this model using an order-statistics FIFO.

This event is generally only called standalone by loggers to obtain an updated position at a specific instant so that the whole updated state of the environment may be stored in a file.

EventUpdateVehicle Updates the overall state of the vehicle. This event should only be triggered when the vehicle arrives at the end of a lane or queue. If the vehicle intends to leave the simulation, it is removed from the environment. Otherwise, it is assumed the vehicle intends to leave the current lane, in which case it informs what connection it will use to leave the lane.

If the vehicle cannot leave the lane (if 1. the controlling traffic light is red, or 2. the destination lane is full), the vehicle is enqueued and no new events are created, since in both scenarios the vehicle delegates control to its queue, and the queue is delegating control either to the traffic light in case 1 or to the downstream queue that is spilling back in case 2.

Otherwise, all that is left to do is to check if the vehicle must yield to any vehicle using a connection with higher priority. If that is the case, then the vehicle is enqueued in the current edge, and a new EventPopQueue is scheduled for the time that the current vehicle is expected to be able to use the connection. If not, then the vehicle transitions to uncongested flow in the lane it picked, and a new EventUpdateVehicle is scheduled for the time the vehicle is expected to arrive at the end of the lane (or at the end of the queue in that lane).

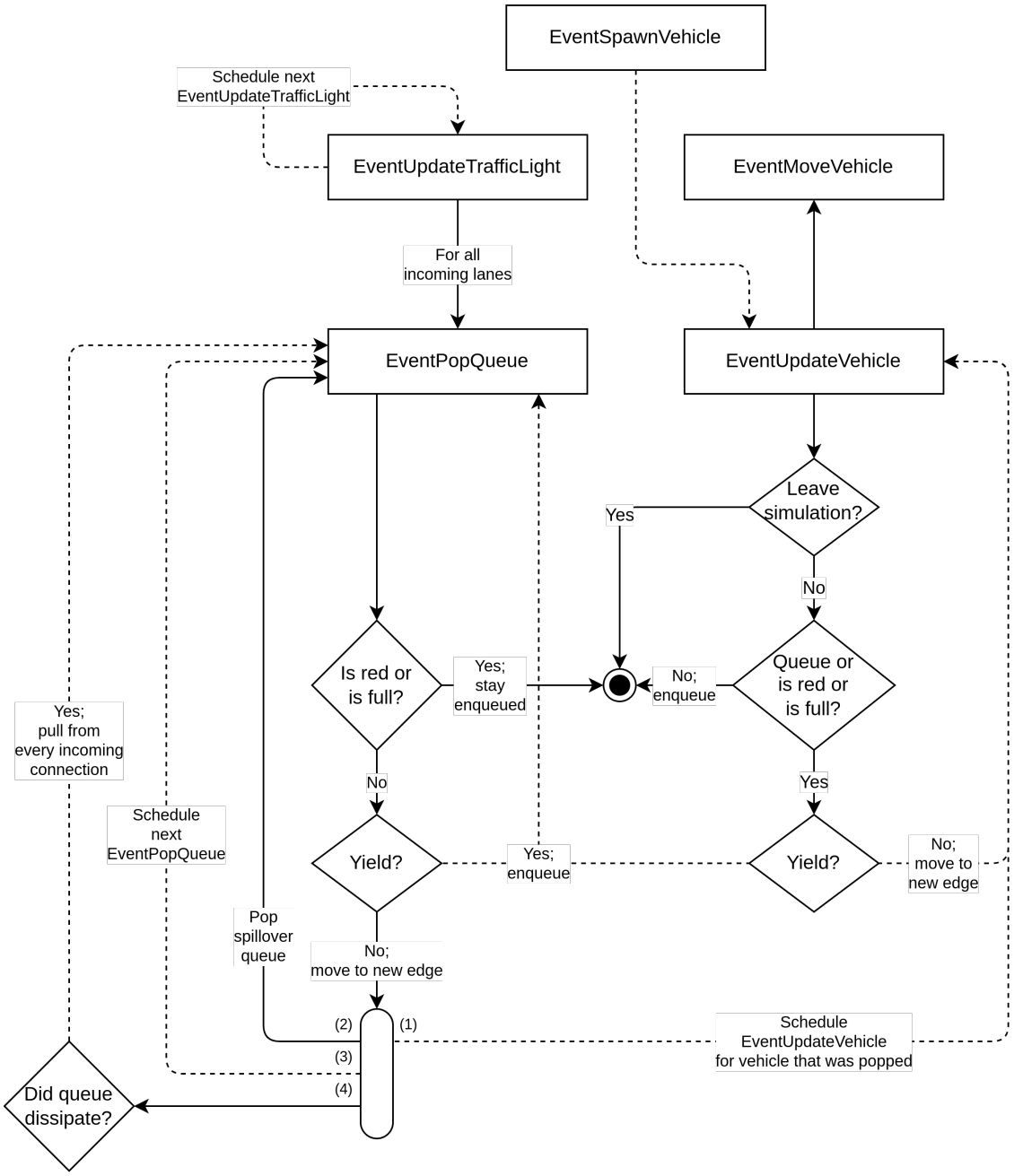


Figure 8.3: Event diagram for the dynamic simulator. Solid line indicates a synchronous (immediate) action was taken, dashed line indicates a new event was scheduled in the future.

EventPopQueue Aside from the check to leave the simulation, the initial part of processing an EventPopQueue is mostly similar to EventUpdateVehicle, except that when all prerequisites are cleared, some more steps need to be taken:

1. Schedule an EventUpdateVehicle for the popped vehicle, similarly to when processing EventUpdateVehicle.

2. spill-back queues are popped; before any spill-back queue is popped, it is first checked if

there are any uninstantiated vehicles in that lane, and if so, those vehicles take precedence; otherwise, the vehicle at the top of a spill-back queue that will use the highest-priority connection is selected, popped and moved to the current queue.

3. The next EventPopQueue is scheduled to occur T_q seconds after the current EventPopQueue.

4. If the queue has dissipated, then pull from all incoming connections so they can sort it out among each other which one takes precedence.

EventUpdateTrafficLight This event changes the phase of a traffic light. It first changes the current phase, then it executes several EventPopQueue events, one for each lane with connections that are controlled by that traffic light. Finally, it schedules the next EventUpdateTrafficLight according to its program.

Because traffic light programs are relatively simple, an EventUpdateTrafficLight can easily schedule the next phase update of that traffic light, without requiring all EventUpdateTrafficLight events to be generated at the start of the simulation, where they would clutter the event queue throughout the entire simulation process.

8.4 Vehicle spawning

Vehicle spawning (or emission) is done by EventSpawnVehicle, but there are many possible ways that EventSpawnVehicle events can be scheduled.

Vehicle spawn scheduling was implemented so as to create all EventSpawnVehicle events *a priori*, before the simulation starts. In the present work, demand data was available in an OD matrix format over a $1\ \mathrm{h}$ time period, so those hourly flows must be converted to discrete EventSpawnVehicle events according to a defined strategy.

Assuming f is the flow between an OD pair, the two strategies that were implemented are:

1. **UniformDemandLoader**: vehicles are uniformly spawned at a regular time interval of $\Delta t = 1/f$, and the first vehicle is spawned at a random instant in range $[0, \Delta t)$ to avoid spawning a large amount of vehicles at $t = 0$.

2. **PoissonDemandLoader**: vehicle arrivals are assumed to follow a Poisson process, so vehicles are spawned at intervals following an exponential distribution with parameter $\lambda = 1/f$. Similarly to UniformDemandLoader, the time instant of the first spawned vehicle is also sampled from the same distribution.

The default loader is UniformDemandLoader, for its simplicity and because it is not relevant to force vehicles to spawn according to a Poisson process since vehicle arrivals in edges that are not sources/sinks will naturally tend towards a Poisson process as a consequence of the many chaotic effects of obstacles in the environment.

8.5 Environment-policy interaction

In this section, the environment-policy interface is described. Policies are not discussed in-depth in this section, since the environment is only concerned with simulating reality, not with how agents decide which connections vehicles should pick.

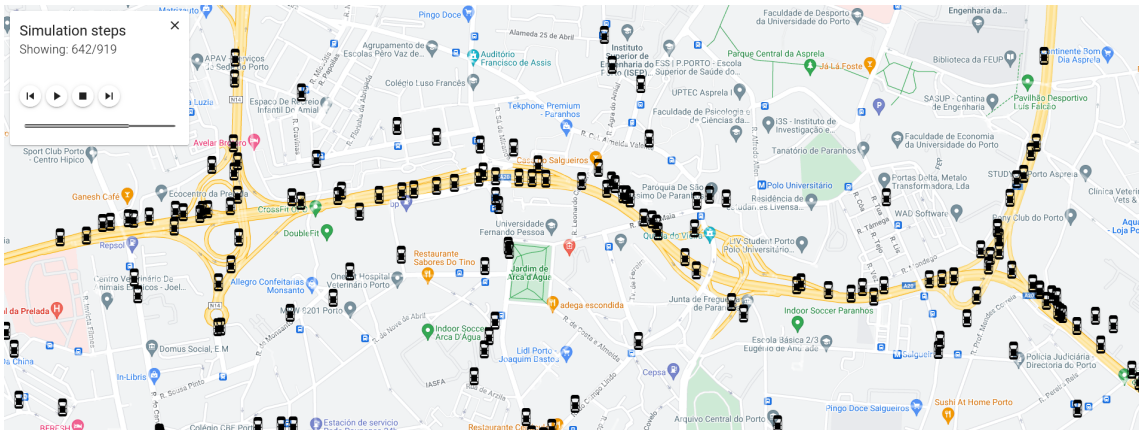
When the environment needs to know where the vehicle will go next, it prompts the policy to provide an *action*, which is an object containing two pieces of information: the connection the vehicle intends to use (while making sure such connection originates from the lane where the vehicle is currently at); and the lane on the destination edge that the vehicle intends to move to after using the connection. This last part allows vehicles to change lanes when traversing an edge. Under this model, the environment must know where the vehicle will go next when it arrives at the end of a lane (or at the end of the traffic queue in that lane).

8.6 Deployment and visualization

DynamiNATOR was containerized using Docker, to make it suitable for off-premise deployment as a service, in alignment with the secondary objective of being available in the form of a SimAAS. DynamiHUB (Ribeiro and Sousa, 2023) is a web-based visualization tool that integrates all sub-projects developed under the DynamiCITY project. As such, DynamiNATOR can be launched as a simulator in a cluster of simulators controlled by DynamiHUB to execute and visualize simulation runs, as shown in Fig. 8.4a.

DynamiNATOR produces outputs of three types, all of which are in SUMO format for better interoperability with other tools that already support those formats. *Edgedata* files contain edge data that can vary with time, and can be visualized with DynamiHUB and SUMO GUI. *Route* files contain the list of edges that vehicles traverse, and can be provided to SUMO for further processing or to visualize vehicle routes in a rough way; this is not recommended, since route files were produced using the DynamiNATOR model but vehicle movements are then being computed and visualized using the SUMO model, so inconsistencies may occur. Finally, *netstate* files, also known as *raw dumps*, contain a list of timesteps, each of which is a snapshot of the network state at a given time instant, containing the position and instantaneous velocity of each vehicle in the network. These are the largest files, typically reaching 1 GB to 2 GB per hour of simulation time.

Although DynamiHUB already allowed visualizing netstate files, its performance for large scenarios was unsatisfactory. Besides, SUMO GUI does not allow to visualize netstate files, presumably because netstate files are only considered inside the SUMO community as outputs for post-processing. This prompted the development of a simple DynamiNATOR GUI application, Fig. 8.5, with the advantage of efficiently accessing local netstate files. The GUI was developed using the C++ programming language as a semi-independent application from the main simulator. It uses the C++ multimedia library SFML (Gomila, 2018), as well as the polygon triangulation library Earcut (Mapbox, 2022) for non-convex polygon drawing.

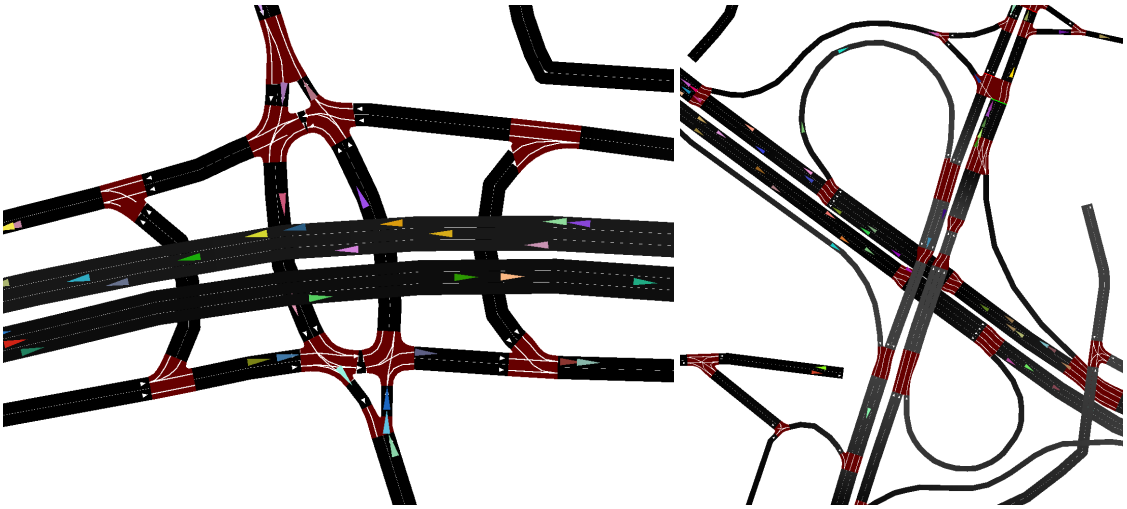


(a) Rendered with DynamiHUB (Ribeiro and Sousa, 2023).

(b) Rendered with DynamiNATOR GUI.

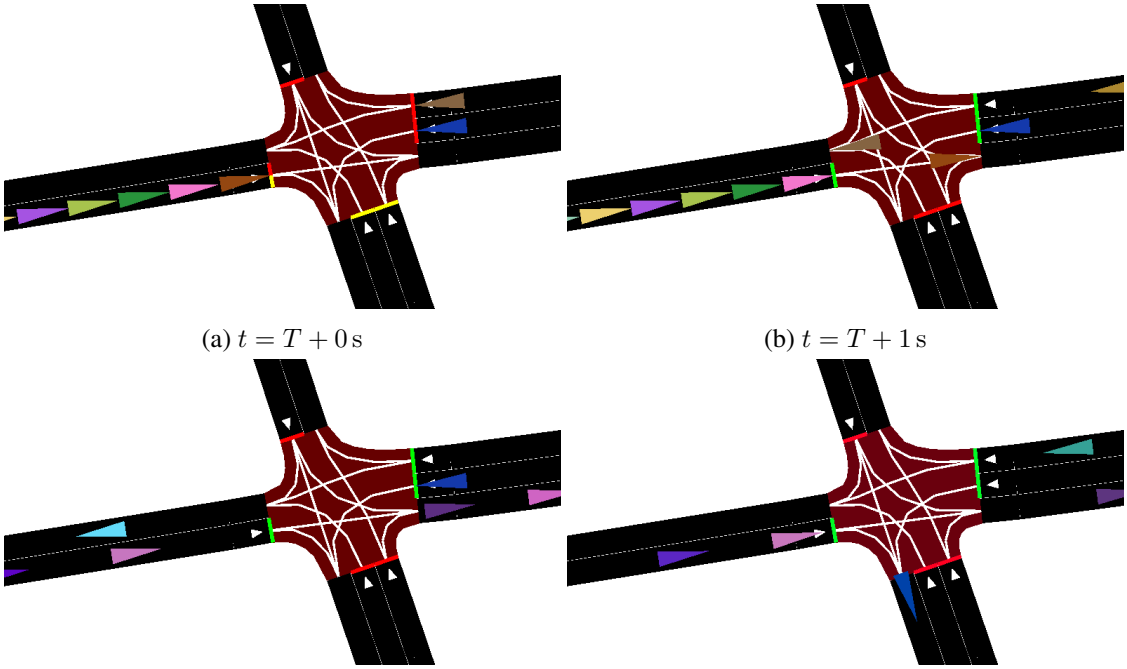
Figure 8.4: Screen captures of a netstate file rendered for same region (VCI between N14 and A3).

Figure 8.5: Screen capture of the traffic network of Porto, rendered with DynamiNATOR GUI.



(a) VCI / Amial (b) VCI / Antas

Figure 8.6: Detailed screen captures of two intersections, rendered with DynamiNATOR GUI.



(a) $t = T + 0\,\mathrm{s}$ (b) $t = T + 1\,\mathrm{s}$

(c) $t = T + 21\,\mathrm{s}$ (d) $t = T + 23\,\mathrm{s}$

Figure 8.7: Four frames of the same intersection at different time instants in DynamiNATOR.

The DynamiNATOR GUI interface and color scheme are based on SUMO GUI for standardization reasons. It accepts arbitrary netstate files, can optionally render elements using z-coordinate information if available in the network file, and roads at higher layers are drawn in lighter colors for more intuitive visualization of roads crossing above or below, Fig. 8.6. As shown in Fig. 8.7, DynamiNATOR can effectively model vehicle queuing at the left-inbound road, the queue dissipating after the traffic light turns green, and the dark-blue vehicle in the left-turning dedicated lane of the right-inbound road yielding to oncoming traffic until it accepts a gap at $t = T + 23\,\mathrm{s}$.

8.7 Parallelism

DynamiNATOR is currently single-threaded since it is simpler to have a single event queue and a single event consumer. Nevertheless, this model has room for improvement through parallelism since most events are highly local, but it is anticipated that such a change would imply vast amounts of mutexes and locking to guarantee no serious race conditions are possible.

Chapter 9

Assignment strategies

In this chapter, the list of implemented assignment strategies, or policies, is presented. The first policy that was implemented was RandomPolicy, which randomly picks the next connection for a vehicle. It was implemented as a simple test to check if the simulator was working correctly. The second policy was PathPolicy, which has the vehicle follow a predetermined path. In conjunction with ShortestPathFactory, a more realistic testing scenario could be implemented where each vehicle follows a predetermined shortest path through the traffic network, assuming no traffic congestion. The most relevant policy, associated with the goal of applying reinforcement learning in assignment strategies, is the policy using Q-learning, described in the following section.

9.1 Q-learning

Under the developed model, state and action are defined as follows. A state $s(l)$ consists of being at the very end of a lane l. An action $a(c, l'')$ consists of the physical action of using connection $c(l, l')$ to move from lane l to lane l', followed by changing lanes in the same edge e' from lane l' to lane l'', and in the meantime traversing the full length of edge e'. For a state-action pair to be valid, the action connection must start at the same lane state.

The ε-greedy Q-learning algorithm was implemented as described in Sec. 5.3.1. It was decided that if s_f is a final state, then $\forall a, Q(s_f, a) = 0$. As such, if we have a state-action pair (s_{f-1}, a_{f-1}) that leads to s_f, the reward is the cost that the vehicle incurred to go from state s_{f-1} to s_f through action a_{f-1} (which is the cost of traversing e_f plus the cost of changing lanes). This reasoning hints at the general conclusion that an initial estimate for $Q(s, a)$ can be made by the following process. Consider $a = a(c(s, l^*), s^*)$, such that s^* is the result of executing action a from state s. We additionally define that l_i^* is the initial position at lane l^*. We then have for the initial estimate of Q,

$$Q_0(s, a) = |c| + |l_i^* \to s^*| + |s^* \rightsquigarrow s_f| \tag{9.1}$$

where $|c|$ is the cost of traversing connection c; $|l_i^* \to s^*|$ is the cost of changing lanes from the start of lane l^* until reaching state s^*, while traversing the edge e that both lanes belong to; and finally, $s^* \rightsquigarrow s_f$ is the shortest path from s^* to s_f, and as such $|s^* \rightsquigarrow s_f|$ is the cost of said path.

In order to reduce memory usage and speed up the learning process, it was decided to use the same Q-matrix for all vehicles that have the same goal TAZ.

9.1.1 Parameter estimation

Learning rate For the learning rate α, it is important to note that a too high α may lead to instabilities in the learning process, while a too small α will hamper significant progress while training. Therefore, we want to pick the highest possible α that will not cause significant instabilities.

To provide an estimate of α, we will assume a very simple scenario of a four-way signalized intersection where the traffic lights have a cycle of $C = 60\,\mathrm{s}$, each road has a green period of $g = 30\,\mathrm{s}$, and the intersection capacity is $\mu = 1440\,\mathrm{h}^{-1} = 0.4\,\mathrm{s}^{-1}$. The aim is to prevent delay from deviating more than $1\,\mathrm{s}$ due to changing traffic light phases.

Assuming an effective utilization ratio of $R = 0.8$, we get $\rho = 0.4$, $\lambda = 0.16\,\mathrm{s}^{-1}$, and as per Eq. (2.29) we get $\overline{w} = 28.5\,\mathrm{s}$. Assuming a platoon of vehicles arrives at the intersection when the traffic lights turn green, at most $\mu g = 12$ vehicles can cross the intersection without any signal-induced delay. These vehicles modify the Q-matrix by reporting a reward $r = 0\,\mathrm{s}$, so the final effect is $Q(s_t, a_t) \leftarrow (1-\alpha)^{12}\overline{w}$. Replacing $\overline{w} = 28.5\,\mathrm{s}$ and $Q(s_t, a_t) = 27.5\,\mathrm{s}$, we get $\alpha \approx 0.002972$. Therefore, we arrive at the theoretical conclusion that α should not be picked to be much greater than 0.003 to avoid excessive fluctuations in the values of the Q-matrix.

The practical scenario is quite different: it was concluded that values of α as high as 0.5 did not cause excessive fluctuations and that any value of α lower than 0.1 significantly slowed down the learning process.

Discount factor The discount factor determines the importance of future rewards. For this problem we know that there are final states, because a vehicle strives to maximize its rewards but the ultimate goal is to reach its destination. Additionally, the second main goal is to optimize the total rewards, and it was experimentally determined that a high discount factor did not cause instabilities. Therefore it was chosen that $\lambda = 1$ for the entire learning process.

9.1.2 Metrics

In this section, the metrics that were used to evaluate the convergence of the learning process and the performance of the resulting policies are presented.

Relative drift Drift D measures the expected difference between previous $Q^*(s, a)$ and new $Q(s, a)$ values in the Q-matrix when it is updated. Absolute drift D_A measures the expected absolute difference between previous and new values, and is used to normalize drift. As such, the

algorithm to determine relative drift is

$$\Delta Q = Q(s,a) - Q^*(s,a) \tag{9.2}$$

$$D \leftarrow (1-\alpha_D)\cdot D + \alpha_D \cdot \Delta Q \tag{9.3}$$

$$D_A \leftarrow (1-\alpha_D)\cdot D_A + \alpha_D \cdot |\Delta Q| \tag{9.4}$$

$$d = \frac{D}{D_A} \tag{9.5}$$

This metric can vary between -1 and $+1$ and is most useful to determine if initial Q-matrix estimations are overestimates (closer to -1) or underestimates (closer to $+1$). It can also be used to check for convergence, although $d \approx 0$ only hints that, if there are any drifts, they are reasonably balanced overall, so it might be the case that some values of the Q-matrix are increasing while others are decreasing and as such cancel out.

Number of vehicles The number of vehicles in the simulation was the main metric that was used to evaluate divergence. When the simulation process was converging, the number of vehicles in the network would typically remain approximately constant or slowly diminish as a result of the policy improving over time, so vehicles would take less time to travel, and fewer vehicles would be in the network at any given time.

However, when grid-locking occurred, the number of vehicles in the network would typically increase rapidly due to queue spill-back. From manual inspection, it was concluded that the vast majority of grid-locking scenarios were due to traffic network errors (most commonly, the traffic network allowed unnatural movements such as turn angles higher than $90°$ in very short edges, which easily led to grid-locking).

Travel time of leaving vehicles Under the assumption that the policy converged to a static solution, the policy performance can be evaluated by considering the average travel time of vehicles that left the simulation in the last Δt seconds of the simulation. In all the experiments that were conducted, the value $\Delta t = 600\,\mathrm{s} = 10\,\mathrm{min}$ was used.

9.2 Reward function

Two reward functions were developed to train the Q-learners.

Greedy The greedy, or user-equilibrium reward function, is the simplest of all. The reward given to an agent for picking action $a(c(l,l'),l'')$ from state $s(l)$ is the negative value of the time t it took for the vehicle to travel from the end of lane l to the end of lane l'', including the cost of using connection c, changing lanes from l' to l'', and traversing the length of the edge e of which both l' and l'' are a part of, as shown in Eq. (9.6).

$$r = -t \tag{9.6}$$

Difference The difference reward function is based on the concept of difference reward presented in Sec. 5.3.2. Difference reward reflects the negative impact the agent has on the overall system performance, so it includes two components.

The first component is the time it took for the vehicle to take action a from state s, similar to the greedy reward function. This is considered a negative impact on overall system performance because the agent does not have an option not to travel, so while having to travel, it should still strive to pick the shortest path.

The second component is the additional costs incurred by other vehicles in the network that are caused by the choice of action a; in other words, the negative effects of the agent on other vehicles. This component is the hardest to quantify, since determining the exact value of the negative effects of an agent's decisions can only be done by performing the entire simulation without said vehicle; needless to say, this is not computationally feasible in a useful time frame. The alternative, as was adopted for this model, is to estimate the negative impact by making some simplifications. The negative effects of a vehicle on others is one of either: slowing down other vehicles in the uncongested flow section, since the vehicle's presence increases vehicle density in the lane; or causing a longer queue, so vehicles in the queue take more time to leave the queue.

As for the first effect, it was estimated by determining the vehicle density with and without the vehicle i at lane l'' where the vehicle is at after taking action, and then the speed of vehicles with and without vehicle i is calculated. The time it would take to travel the length of the lane is then determined for the case with and without vehicle i, and the difference in travel times is considered to be the negative impact of vehicle i on each individual vehicle that was at lane l'' at the time vehicle i leaves it.

For the second effect, all the vehicles in the queue at lane l'' as of the time vehicle i leaves l'' are considered to incur a time delay of T_q, because all those vehicles were forced to leave the queue T_q seconds after what they would if vehicle i was not in that queue.

Thus, the process to calculate the difference reward is as follows:

$$K_{\text{w}} = \frac{N}{L} \tag{9.7}$$

$$K_{\text{wo}} = \frac{N-1}{L} \tag{9.8}$$

$$v_{\text{w}} = \max\left\{v_q, v_0\left(1 - \frac{K_{\text{w}}}{K_{\text{jam}}}\right)\right\} \tag{9.9}$$

$$v_{\text{wo}} = \max\left\{v_q, v_0\left(1 - \frac{K_{\text{wo}}}{K_{\text{jam}}}\right)\right\} \tag{9.10}$$

$$t_{\text{w}} = L/v_{\text{w}} \tag{9.11}$$

$$t_{\text{wo}} = L/v_{\text{wo}} \tag{9.12}$$

$$\Delta t_{\text{moving}} = t_{\text{w}} - t_{\text{wo}} \tag{9.13}$$

$$\Delta t_{\text{queue}} = T_q \tag{9.14}$$

$$r = -t - w\left[(N-1)\Delta t_{\text{moving}} + (Q-1)\Delta t_{\text{queue}}\right] \tag{9.15}$$

where L is the length of the lane, N is the number of vehicles in the lane, Q is the number of vehicles in the queue, and w is the *greediness factor*, a weighing factor that defines how concerned an agent is with the negative effect it has on other agents; $w = 0$ is equivalent to a purely greedy agent, while $w = 1$ refers to an agent that is very concerned with its impact on the system and its actions will contribute toward a system optimum.

Chapter 10

Results & Discussion

All experiments were executed on a laptop computer with the following specifications:

- RAM: 15.3 GB DDR4
- CPU: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz × 8
- Operating system: Ubuntu 20.04.6 LTS (64-bit)

The first set of experiments seeks to evaluate performance gains compared with SUMO (Sec. 10.1), followed by a second set of experiments on the effect of different reward functions on the number of vehicles in simulation and average travel time of leaving vehicles (Sec. 10.2).

10.1 Performance comparison

Three experiments were conducted to compare SUMO and DynamiNATOR in terms of performance. Each experiment contains three trials, and was conducted as follows:

1. The scenario was simulated with DynamiNATOR, using the Q-learning algorithm with greedy reward. Route files are obtained from DynamiNATOR, and execution time is recorded.
2. Route files obtained from DynamiNATOR are supplied to SUMO, which runs those routes in a similar scenario. The execution time is recorded.

Tab. 10.1 presents the results of these experiments, showing that DynamiNATOR was significantly faster than SUMO on all experiments, with speedup ratios between 22.52 and 32.51. It can be concluded that DynamiNATOR meets the goal of high performance, particularly when compared with SUMO.

Table 10.1: Execution times of simulators SUMO and DynamiNATOR for similar scenarios

Scale	Simulation time (h)	SUMO (s)	DynamiNATOR (s)	Speedup
0.2	1	128.045 ± 2.921	4.853 ± 0.083	26.39
0.1	2	140.130 ± 2.683	4.310 ± 0.023	32.51
0.05	4	114.307 ± 2.065	5.076 ± 0.007	22.52

10.2 Assignment strategies

To evaluate different assignment strategies, a common scenario was established (Tab. 10.2). This scenario includes a warm-up process where the demand scale is progressively increased over the simulation, allowing agents to gain some insight into the traffic network before loading it with a higher number of vehicles, to prevent gridlocks. Warm-up occurs until $t = 100\,\mathrm{h}$, after which point the scale is fixed at 0.45. Higher values of the demand scale caused simulations to diverge due to excessive congestion and simulator limitations.

Fig. 10.1 shows the number of vehicles and average travel time of vehicles leaving the simulation. Data points were sampled at in-simulation periods of $p = 600\,\mathrm{s}$, and the average travel time of vehicles leaving the simulation at each sample was calculated from the travel time of vehicles that left the simulation since the last sample, during time period $(t - p, t]$.

Table 10.2: Scenario to evaluate different assignment strategies.

Scale	Start (h)	Finish (h)
0.010	0	1
0.100	1	10
0.200	10	30
0.300	30	43
0.350	43	55
0.375	55	75
0.400	75	90
0.425	90	100
0.450	100	200

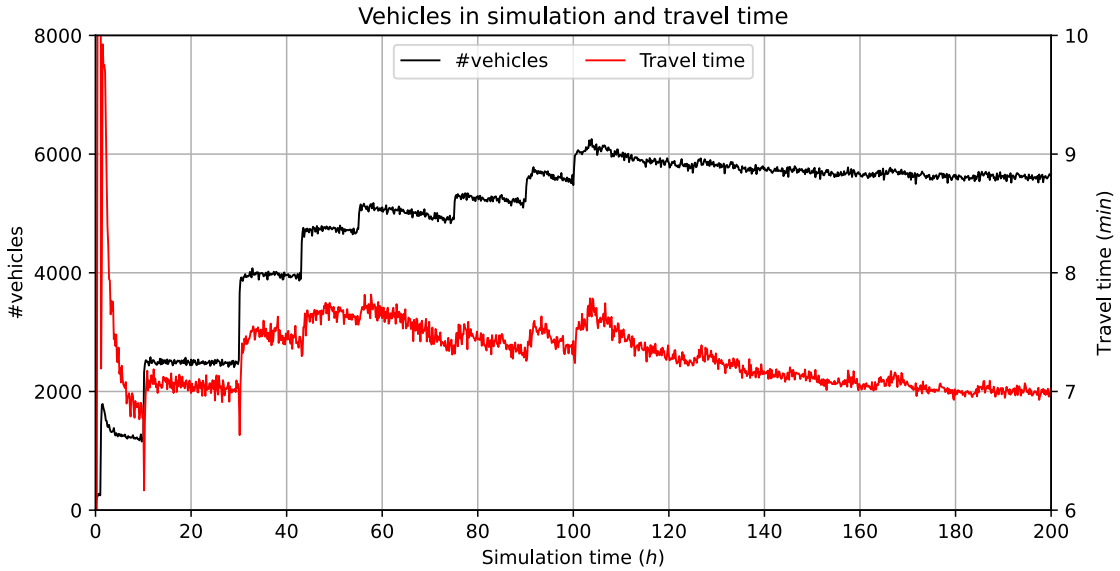


Figure 10.1: Number of vehicles and average travel time of vehicles leaving the simulation.

As seen in Fig. 10.1, the number of vehicles in the simulation initially shows a step increase as a natural consequence of the warm-up program designed for this experiment. Once the program reaches maximum scale at $t = 100\,\mathrm{h}$, the number of vehicles slowly decreases from 6253 to 5651, a 10% decrease. Another pattern that can be observed is that, as the number of vehicles in the simulation increases, the slope of the number of vehicles is increasingly negative. This is because vehicles with the same goal share the same Q-matrix, so the more vehicles there are in the network, the more updates to the Q-matrix there are, and vehicles learn faster.

Average travel time is initially large, reaching $15.23\,\mathrm{min}$, which can be linked to an initial stage where a few Q-value estimates differ significantly from observed rewards, so the agents go through a very rapid adaptation process. After that point, average travel time presents a more stable behavior. Travel time increases immediately after the demand scale increases, since agents are using knowledge of a less congested network to navigate a more congested network, and are taking longer paths but also adapting to the new congestion conditions. After the increase in travel time caused by increasing scale, travel time slowly decreases as agents learn better actions to navigate the congested network more efficiently. When the last stage starts, average travel time rises up to $7.785\,\mathrm{min}$, and then slowly decreases until it reaches the final value of $7.031\,\mathrm{min}$.

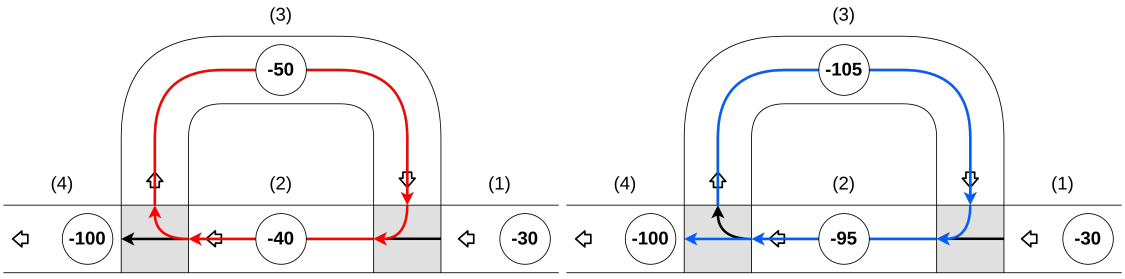
Both of these results clearly show that the agents are learning, albeit at a slow pace; note that the simulation spans over a period of $200\,\mathrm{h}$. This can be explained by the high level of detail that was chosen for the network, which means vehicles have to take a much larger number of actions to reach their final destinations, so expected future reward values also take longer to propagate to previous actions' Q-values.

However, it was noticeable that some vehicles occasionally became stuck driving in loops using the same small set of edges. This is assumed to be caused by the selected low value of ε that hinders the capability of vehicles from leaving local optimums, together with faulty initial estimates of Q-values that underestimate penalties. This causes a scenario similar to Fig. 10.2a where the downstream Q-value in edge 4 has already been updated to a more realistic estimate of -100, so the vehicle is stuck in the red loop in edges 2 and 3, slowly aggravating the costs of edges 2 and 3 at each time it goes around the loop until, finally, the updated cost of edge 3 becomes -105, which is worse than the downstream Q-value of -100, at which point the vehicle selects the blue route in Fig. 10.2b and leaves the loop. Agents eventually leave loops, but their looping behavior is nevertheless unrealistic.

Five experiments were then conducted with Q-learners using the difference reward function, for distinct values of the greediness factor w, acoording to Tab. 10.3.

Table 10.3: Experiments with different greediness factors

Experiment	w
1	0.0
2	0.1
3	0.5
4	0.75
5	0.9



(a) When the vehicle first enters this section of the network, it selects the loop.

(b) After updating Q-values, the vehicle escapes the loop.

Figure 10.2: Diagrams demonstrating how a vehicle may become stuck in a loop due to Q-value underestimation.

The higher the greediness factor corresponding to the reciprocal of greediness (so, the more concerned agents are with the overall system performance), the more learning difficulties agents had at the beginning of the simulation (up to $t = 100\,\mathrm{h}$), Fig. 10.3. At the later stage $t > 100\,\mathrm{h}$, the travel time presents a similar trend in all experiments. The final average travel times in Tab. 10.4 were calculated by averaging the travel times of vehicles leaving the simulation in the period $[198, 200]$. There is a decreasing trend of the final average travel time as w increases in agreement with the literature (Grunitzki et al., 2014). However, the relative differences of 0.4% are small when compared to the mean values, and these differences smaller than the values around 1% in literature (Peeta and Mahmassani, 1995) under similar conditions. The SO - UE gap is most noticeable in high-load situations, with the considered scenario presenting low loads so as to avoid divergent behavior of the model under high loads.

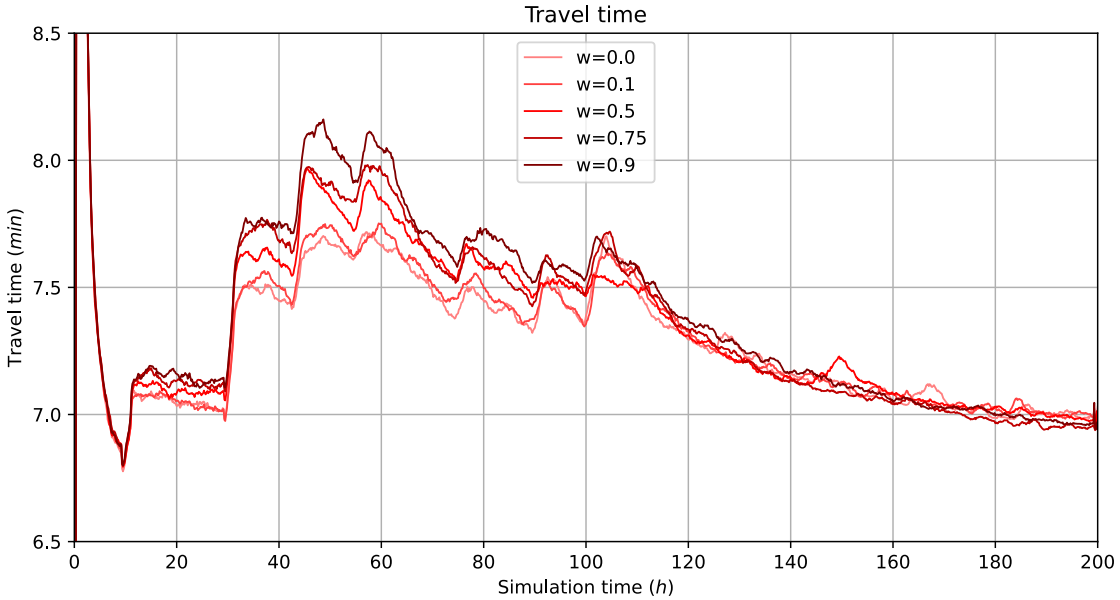


Figure 10.3: Travel time for agents using difference demand with distinct values of greediness factor w.

Table 10.4: Final travel times in experiments with different values of greediness factor w.

w	Final average travel time (s)
0.00	419.680
0.10	419.319
0.50	418.512
0.75	417.344
0.90	417.836

Chapter 11

Conclusions

The goals of this work were to develop a performant traffic simulator, and to use the simulator to investigate the impact of more socially-aware traffic assignment strategies. Namely, the scientific gaps this work addressed are the lack of means to include new metrics in the decision process that lead to fairer and more socially-aware assignments (which was addressed by using different reward functions), the lack of study of complex assignment strategies in large-scale traffic networks (addressed by studying the traffic network of an entire city), as well as the lack of a specific traffic model for the data (city of Porto) and type of analysis that was performed (mesoscopic but performant). These goals were accomplished through the implementation of a performant mesoscopic simulator, and its application to a large-scale scenario with different traffic assignment strategies.

With respect to the performed experiments, it was verified that DynamiNATOR was significantly faster than SUMO. It was also verified that the Q-learning agents were able to learn in a large-scale, complex, and noisy scenario of a large city, and that the employment of difference reward functions with different greediness factors yielded better global cost results, even if the differences in global costs are small enough to be debatable whether they are significant or not.

This work makes contributions at the following levels:

- **Scientific:** By studying and applying new traffic assignment strategies from a theoretical standpoint, by building on existing scientific literature.
- **Technological:** Several deliverables resulted from this dissertation work, including a traffic network processing pipeline, a mesoscopic large-scale traffic model and simulator, and implementations of different assignment strategies. Refer to appendix A for the source code.
- **Application:** The mesoscopic simulator will be available via the web as a virtual laboratory integrated with DynamiHUB (Ribeiro and Sousa, 2023) so that stakeholders may use it for pure or applied research and to guide public policy regarding infrastructure.

Some proposals for improvement of this work were already laid out, including making the model more realistic by modifying the movement equations under uncongested flow. This simulation model only supports light passenger vehicles, so a natural extension of this model would be

towards multimodal transportation with the inclusion of a diverse mix of vehicles, including most notably trucks, buses, and bikes, as well as pedestrian traffic to simulate the impedance they exert on vehicles when crossing roads. An additional improvement could be made by allowing vehicles to teleport or disappear from the environment after having waited for a certain time interval, so as to automatically resolve gridlocks in a similar way to SUMO and as such create high-congestion scenarios without the model diverging.

Regarding the RL aspects, the adopted approach based on ε-greedy Q-learning can be extended to use other algorithms, namely SARSA (Rummery and Niranjan, 1994) that can be effortlessly implemented in DynamiNATOR for a potential improvement in learning speed. To address the issue of vehicles looping through the same edges repeatedly, the Q-learning algorithm can be modified to incorporate the concept of *tabus* from Tabu Search (Glover, 1989), namely through the use of intermediate-term tabus (Glover, 1990) where edges that the vehicle already visited receive a penalty that grows exponentially with the number of times the vehicle visited that edge. The extensible way in which the code was developed also allows other distributed multi-agent approaches, such as auctions and negotiation, to be applied in future studies.

The methodological approach proposed in this dissertation and the deliverables that resulted from its implementation constitute a relevant framework for traffic assignment in the form of a virtual laboratory, allowing the study of large-scale networks and analysis of city-wide metrics directly associated with the Sustainable Development Goals proposed by the United Nations.

References

ACEA (2022). Vehicles in use Europe 2022. Yearly report, European Automobile Manufacturers Association (ACEA).

Agogino, A. and Tumer, K. (2005). Multi-agent reward analysis for learning in noisy domains. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS'05, pages 81–88, Utrecht, The Netherlands. ACM.

Ashalatha, R. and Chandra, S. (2011). Critical gap through clearing behavior of drivers at unsignalised intersections. *KSCE Journal of Civil Engineering*, 15(8):1427–1434.

Beckmann, M., McGuire, C. B., and Winsten, C. B. (1956). *Studies in the Economics of Transportation*. Yale University Press, Hew Haven, CT.

Bellini, F. and Inconis, R. (2021). RapidXML v1.15.

Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H. N., and Mishalani, R. (1998). DynaMIT: A simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop*, Delft, The Netherlands.

Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H. N., and Mishalani, R. (2002). Real time simulation of traffic demand-supply interactions within DynaMIT. In Gendreau, M. and Marcotte, P., editors, *Transportation and Network Analysis: Current Trends*, volume 63 of *Applied Optimization*, chapter 2, pages 19–36. Springer, Dordrecht, The Netherlands.

Black, J., Hashimzade, N., and Myles, G. (2017). *A Dictionary of Economics*. Oxford University Press, New York, NY, 5th edition.

Boyles, S. D., Lownes, N. E., and Unnikrishnan, A. (2019). *Transportation Network Analysis*, volume 1. Self-published, v0.8 edition.

Braess, D. (1968). Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268.

Brannigan, C., Biedka, M., Hitchcock, G., and Fiorello, D. (2017). Study on urban mobility – Assessing and improving the accessibility of urban areas. Final report and policy proposals, European Comission, Directorate-General for Mobility and Transport, Brussels, Belgium.

Bureau of Public Roads (1964). *Traffic Assignment Manual for Application with a Large, High Speed Computer*. U.S. Department of Commerce.

Camargo, P. (2018). Traffic Assignment Procedure. Python Package Documentation v0.8.3, AequilibraE.

CM Porto (2023). Sistema de Gestão de Tráfego - Semáforos.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3rd edition.

Crociani, L., Lämmel, G., Park, H. J., and Vizzari, G. (2017). Cellular Automaton Based Simulation of Large Pedestrian Facilities - A Case Study on the Staten Island Ferry Terminals. *arXiv: Computer Science > Multiagent Systems*.

Davidson, K. B. (1966). A Flow Travel Time Relationship for use in Transportation Planning. *Australian Road Research Board Proceedings*, 3(1):183–194.

de Palma, A., Kilani, M., and Lindsey, R. (2005). Comparison of Second-Best and Third-Best Tolling Schemes on a Road Network. *Transportation Research Record*, 1932(1):89–96.

Dijkstra, E. W. (1987). Twenty-eight years. Manuscript EWD1000, University of Texas at Austin, Austin, TX.

Dresner, K. and Stone, P. (2004). Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. *New York*.

Elimadi, M., Abbas-Turki, A., and Koukam, A. (2021). A Novel Approach for Dynamic Traffic Assignment Based on Multi-Agent Node Reservation: Comparative Study on Two Competing Roads. *Procedia Computer Science*, 191:320–327.

Erdmann, J. (2014). Lane-Changing Model in SUMO. In *Proceedings of the SUMO2014 Modeling Mobility with Open Data*, volume 24 of *Reports of the DLR-Institute of Transportation Systems*, pages 77–88, Berlin, Germany. German Aerospace Center.

European Comission (2011). Impact Assessment - Accompanying document to the Roadmap to a Single European Transport Area – Towards a competitive and resource efficient transport system. Commission Staff Working Paper SEC(2011) 358 final, European Comission, Brussels, Belgium.

Eurostat (2023). Population on 1 January by age and sex.

Ferrara, A., Sacone, S., and Siri, S. (2018). *Freeway Traffic Modelling and Control*. Advances in Industrial Control. Springer International Publishing, Cham, Switzerland.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

German Aerospace Center (2023a). SUMO. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

German Aerospace Center (2023b). SUMO - Contributed. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

German Aerospace Center (2023c). SUMO - PlainXML. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

German Aerospace Center (2023d). SUMO - TraCI. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

German Aerospace Center (2023e). SUMO at a Glance. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

German Aerospace Center (2023f). SUMO Road Networks. User documentation, Institute of Transportation Systems, German Aerospace Center, Hamburg, Germany.

Gipps, P. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414.

Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu Search: A Tutorial. *Interfaces*, 20(4):74–94.

Gomila, L. (2018). SFML v2.5.1.

Greenshields, B. D. (1935). A Study of Traffic Capacity. *Highway Research Board*, 14:448–477.

Grunitzki, R., Ramos, G. d. O., and Bazzan, A. L. C. (2014). Individual versus Difference Rewards on Reinforcement Learning for Route Choice. In *2014 Brazilian Conference on Intelligent Systems*, pages 253–258, São Paulo, Brazil. IEEE.

Hardin, G. (1968). The Tragedy of the Commons: The population problem has no technical solution; it requires a fundamental extension in morality. *Science*, 162(3859):1243–1248.

Herman, R. and Weiss, G. (1961). Comments on the Highway-Crossing Problem. *Operations Research*, 9(6):828–840.

Hirose, Y. (2023). Cpp-httplib v0.12.3.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.

Irani, S. and Leung, V. (1996). Scheduling with Conflicts, and Applications to Traffic Signal Control. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'96, pages 85–94, Atlanta, GA. Society for Industrial and Applied Mathematics.

Kessels, F. (2019). *Traffic Flow Modelling: Introduction to Traffic Flow Theory Through a Genealogy of Models*. EURO Advanced Tutorials on Operational Research. Springer, Cham, Switzerland.

Klein, I., Levy, N., and Ben-Elia, E. (2018). An agent-based model of the emergence of cooperation and a fair and stable system optimum using ATIS on a simple road network. *Transportation Research Part C: Emerging Technologies*, 86:183–201.

Knight, F. H. (1924). Some Fallacies in the Interpretation of Social Cost. *The Quarterly Journal of Economics*, 38(4):582–606.

Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. (2002). SUMO (Simulation of Urban MObility) – an open-source traffic simulation. In Al-Akaidi, M., editor, *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM2002)*, pages 183–187, Sharjah, UAE. SCS Europe.

Krauß, S. (1998). *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. PhD thesis, Faculty of Mathematics and Natural Sciences – University of Cologne, Cologne, Germany.

Liu, R. (2010). Traffic Simulation with DRACULA. In Barceló, J. and Hillier, F. S., editors, *Fundamentals of Traffic Simulation*, volume 145 of *International Series in Operations Research & Management Science*, pages 295–322. Springer, New York, NY.

Lujak, M., Giordani, S., and Ossowski, S. (2015). Route guidance: Bridging system and user optimization in traffic assignment. *Neurocomputing*, 151:449–460.

Luttinen, R. T. (1996). *Statistical Analysis of Vehicle Time Headways*. PhD thesis, Helsinki University of Technology, Otaniemi, Finland.

Maciejewski, M. (2010). A comparison of microscopic traffic flow simulation systems for an urban area. *Transport Problems*, 5(4):27–38.

Mapbox (2022). Earcut v2.2.4. Mapbox.

Martínez, L. M., Viegas, J. M., and Silva, E. A. (2009). A traffic analysis zone definition: A new methodology and algorithm. *Transportation*, 36(5):581–599.

Microsoft and Vexcel Imaging (2023). Bing aerial imagery.

Miller, H. J. and Shaw, S.-L. (2001). *Geographic Information Systems for Transportation: Principles and Applications*. Spatial Information Systems. Oxford University Press, New York.

Milosavljevic, D. D. M. (2022). Color.

Monteiro, N., Rossetti, R., Campos, P., and Kokkinogenis, Z. (2014). A Framework for a Multimodal Transportation Network: An Agent-based Model Approach. *Transportation Research Procedia*, 4:213–227.

Morandi, V. (2021). Bridging the user equilibrium and the system optimum in static traffic assignment: How the cooperation among drivers can solve the congestion problem in city networks. *arXiv: Computer Science > Computer Science and Game Theory*.

Morandi, V. (2023). Bridging the user equilibrium and the system optimum in static traffic assignment: A review. *4OR - A Quarterly Journal of Operations Research*, postprint.

Nash, J. (1951). Non-Cooperative Games. *The Annals of Mathematics*, 54(2):286.

Newell, G. F. (1961). Nonlinear Effects in the Dynamics of Car Following. *Operations Research*, 9(2):209–229.

Nor Azlan, N. N. and Md Rohani, M. (2018). Overview Of Application Of Traffic Simulation Model. *MATEC Web of Conferences*, 150:03006.

Olbricht, R. (2023). Overpass API. OpenStreetMap Foundation.

OpenStreetMap contributors (2023). OpenStreetMap.

Passos, L. S., Rossetti, R. J. F., and Kokkinogenis, Z. (2011). Towards the next-generation traffic simulation tools: A first appraisal. In *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*, pages 1–6, Chaves, Portugal. IEEE.

Peeta, S. and Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1):81–113.

Phillips, E. (2017). GMath. YclepticStudios.

Pigou, A. C. (1920). *The Economics of Welfare*. MacMillan, London, UK.

PTV (1992). VISUM. Planung Transport Verkehr GmbH.

PTV (2001). *VISUM User Manual Version 7.50*. Karlsruhe, Germany.

Rahimi-Farahani, H., Rassafi, A. A., and Mirbaha, B. (2019). Forced-node route guidance system: Incorporating both user equilibrium and system optimal benefits. *IET Intelligent Transport Systems*, 13(12):1851–1859.

Ramos, G. d. O., Da Silva, B. C., Rădulescu, R., Bazzan, A. L. C., and Nowé, A. (2020). Toll-based reinforcement learning for efficient equilibria in route choice. *The Knowledge Engineering Review*, 35:e8.

Ribeiro, R. and Sousa, J. A. (2023). DynamiHUB. DynamiCITY.

Rose, G., Daskin, M. S., and Koppelman, F. S. (1988). An examination of convergence error in equilibrium traffic assignment models. *Transportation Research Part B: Methodological*, 22(4):261–274.

Rossetti, R. J. F. and Liu, R. (2005a). An Agent-Based Approach to Assess Drivers' Interaction with Pre-Trip Information Systems. *Journal of Intelligent Transportation Systems*, 9(1):1–10.

Rossetti, R. J. F. and Liu, R. (2005b). A Dynamic Network Simulation Model Based on Multi-Agent Systems. In Klügl, F., Bazzan, A., and Ossowski, S., editors, *Applications of Agent Technology in Traffic and Transportation*, pages 181–192. Birkhäuser-Verlag, Basel.

Rossetti, R. J. F., Liu, R., Cybis, H. B. B., and Bampi, S. (2002). A Multi-Agent Demand Model. In *Proceedings of the 13th Mini-Euro Conference and The 9th Meeting of the Euro Working Group Transportation*, pages 193–198, Italy. Polytechnic University of Bari.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-Learning Using Connectionist Systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department - Cambridge University, Cambridge, UK.

Schepperle, H. and Böhm, K. (2007). Agent-Based Traffic Control Using Auctions. In Carbonell, J. G., Siekmann, J., Klusch, M., Hindriks, K., Papazoglou, M. P., and Sterling, L., editors, *Cooperative Information Agents XI*, volume 4676 of *Lecture Notes in Artificial Intelligence*, pages 119–133, Delft, The Netherlands. Springer.

Schepperle, H., Böhm, K., and Forster, S. (2008). Traffic Management Based on Negotiations between Vehicles – A Feasibility Demonstration Using Agents. In van der Aalst, W., Mylopoulos, J., Sadeh, N. M., Shaw, M. J., Szyperski, C., Collins, J., Faratin, P., Parsons, S., Rodriguez-Aguilar, J. A., Sadeh, N. M., Shehory, O., and Sklar, E., editors, *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, volume 13 of *Lecture Notes in Business Information Processing*, pages 90–104, Berlin. Springer.

Scholz, I. and Stöcker, D. (2023). JOSM v18746.

Sheffi, Y. (1985). *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Englewood Cliffs, NJ.

Sloboden, J., Lewis, J., Alexiadis, V., Chiu, Y.-C., and Nava, E. (2012). Traffic Analysis Toolbox Volume XIV: Guidebook on the Utilization of Dynamic Traffic Assignment in Modeling. Final Report FHWA-HOP-13-015, Federal Highway Administration, Office of Operations, Washington, DC.

Speranza, M. G. (2018). Trends in transportation and logistics. *European Journal of Operational Research*, 264(3):830–836.

Sutton, R. S. and Barto, A. G. (2020). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2 edition.

Transportation Research Board (2000). *Highway Capacity Manual*. Transportation Research Board, National Research Council, Washington, DC.

Tumer, K. and Agogino, A. (2007). Distributed Agent-Based Air Traffic Flow Management. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS'07, pages 342–349, Honolulu, HI. ACM.

Tumer, K., Welch, Z. T., and Agogino, A. (2008). Aligning social welfare and agent preferences to alleviate traffic congestion. In Padgham, L., Parkes, D., Müller, J., and Parsons, S., editors, *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2 of *AAMAS'08*, pages 655–662, Estoril, Portugal. International Foundation for Autonomous Agents and Multiagent Systems.

Tumer, K. and Wolpert, D. (2004). A Survey of Collectives. In Tumer, K. and Wolpert, D., editors, *Collectives and the Design of Complex Systems*, pages 1–42. Springer, New York, NY.

Ulrich, J. (2022). HTTP Status Codes and Reason Phrases for C, C++ and Qt.

United Nations (2015). Transforming our World: The 2030 Agenda for Sustainable Development. Technical Report A/RES/70/1, United Nations, New York, NY.

United Nations (2018). World Urbanization Prospects: The 2018 Revision, Online Edition.

United Nations (2019). World Urbanization Prospects 2018: Highlights. Technical Report ST/ESA/SER.A/421, United Nations, Department of Economic and Social Affairs, Population Division, New York, NY.

Vasirani, M. (2009). *Vehicle-Centric Coordination for Urban Road Traffic Management: A Market-Based Multiagent Approach*. PhD thesis, Rey Juan Carlos University, Madrid.

Vasirani, M. and Ossowski, S. (2012). A Market-Inspired Approach for Intersection Management in Urban Road Traffic Networks. *Journal of Artificial Intelligence Research*, 43:621–659.

Vitsentiy, V. (2015). CTPL.

Wardrop, J. G. (1952). Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., and Hubaux, J.-P. (2008). TraCI: An interface for coupling road traffic and network simulators. In *Proceedings of the 11th Communications and Networking Simulation Symposium*, pages 155–163, Ottawa, Canada. ACM.

Yang, H. and Zhang, X. (2003). Optimal Toll Design in Second-Best Link-Based Congestion Pricing. *Transportation Research Record*, 1857(1):85–92.

Yang, Q. and Koutsopoulos, H. N. (1996). A Microscopic Traffic Simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129.

Appendix A

Source code

The source code corresponding to the simulator developed in this work is available on GitHub, at `https://github.com/dmfrodrigues/dynaminator`.

The original working name of the simulator was "DynamiNATOR". Owed to the lack of any better alternative, this impactful name progressively came to be accepted not only as its working name but also as its definite name.

The simulator is implemented in C++17. Besides containing the actual simulator as a library, the repository also contains the necessary utilities to containerize and make the simulator available via the web in the form of an API server.

This project uses the following external libraries:

- Color: yet another c++ library that implements color (Milosavljevic, 2022).
- cpp-httplib: a C++11 single-file header-only cross platform HTTP/HTTPS library (Hirose, 2023).
- CTPL: modern and efficient C++ Thread Pool Library (Vitsentiy, 2015).
- Earcut: C++ port of earcut.js, a fast, header-only polygon triangulation library (Mapbox, 2022).
- GMath: lightweight C++ math library aimed at geometry math problems (Phillips, 2017).
- HTTP Status Codes and Reason Phrases for C, C++ and Qt (Ulrich, 2022).
- RapidXML: *in-situ* XML parser written in modern C++ (Bellini and Inconis, 2021).
- SFML: simple, fast, cross-platform and object-oriented multimedia API (Gomila, 2018).

Appendix B

Gap acceptance model

This appendix shows some algebraic expansions for estimating average waiting time per vehicle coming from the minor road, in the situation where a minor road meets a major road and all vehicles in the minor road yield. This scenario is better described in Sec. 2.3.4 and 2.3.4.1. The expression for $\mathrm{E}[W]$ is also useful for modeling delay associated to lane changing.

Consider the mathematical formulation proposed by Sheffi (1985). The author assumes a gap is accepted whenever it is greater than t_{cr}, so $P_a(t)$ simply becomes

$$P_a(t) = \begin{cases} 0, & t < t_{cr} \\ 1, & t \geq t_{cr} \end{cases} \tag{B.1}$$

so the formulas become

$$p = 1 - F_T(t_{cr}) \tag{B.2}$$

$$\mathrm{E}[N] = \frac{F_T(t_{cr})}{1 - F_T(t_{cr})} \tag{B.3}$$

$$f_{T|\text{rejected}}(t) = \frac{f_T(t)}{F_T(t_{cr})} \tag{B.4}$$

$$\mathrm{E}[T \mid \text{rejected}] = \frac{\displaystyle\int_0^{t_{cr}} \tau \cdot f_T(\tau)\, d\tau}{F_T(t_{cr})} \tag{B.5}$$

$$\mathrm{E}[W] = \frac{\displaystyle\int_0^{t_{cr}} \tau \cdot f_T(\tau)\, d\tau}{1 - F_T(t_{cr})} \tag{B.6}$$

where $f_{T|\text{rejected}}(t)$ is only defined for $t \in [0, t_{cr})$.

To calculate the actual waiting times, all we need is the distribution of the gap lengths. If we assume traffic in the major can be modeled as a Poisson process, then $f_T(t)$ follows an exponential

distribution of parameter λ which is the flow on the major, and we get the following:

$$f_T(t) = \lambda e^{-\lambda t} \tag{B.7}$$

$$F_T(t) = 1 - e^{-\lambda t} \tag{B.8}$$

$$p = 1 - F_T(t_{cr}) = e^{-\lambda t_{cr}} \tag{B.9}$$

$$\int_0^t \tau\, f_T(\tau)\, d\tau = \int_0^t \tau\, \lambda\, e^{-\lambda\tau}\, d\tau = \frac{1}{\lambda}(1 - (1+\lambda t)e^{-\lambda t}) \tag{B.10}$$

$$f_{T|\text{rejected}}(t) = \frac{f_T(t)}{F_T(t_{cr})} = \frac{\lambda}{1 - e^{-\lambda t_{cr}}} \cdot e^{-\lambda t} \tag{B.11}$$

$$\mathrm{E}[T \mid \text{rejected}] = \frac{\displaystyle\int_0^{t_{cr}} \tau\, f_T(\tau)\, d\tau}{F_T(t_{cr})} = \frac{1}{\lambda} - t_{cr}\frac{1}{e^{\lambda t_{cr}} - 1} \tag{B.12}$$

and thus Eq. (B.6) becomes

$$\mathrm{E}[W] = \frac{\displaystyle\int_0^{t_{cr}} \tau\, f_T(\tau)\, d\tau}{1 - F_T(t_{cr})} = \frac{1}{\lambda}(e^{\lambda t_{cr}} - 1) - t_{cr} \tag{B.13}$$

We now want to calculate

$$\mathrm{E}[D] = \frac{q\,\mathrm{E}[W^2]}{2(1 - q\,\mathrm{E}[W])} + \mathrm{E}[W] \tag{B.14}$$

We already have $\mathrm{E}[W]$, so we want to calculate $\mathrm{E}[W^2]$.

From Sheffi (1985, p. 357), we know that

$$\mathrm{E}[W^2] = \mathrm{E}[N^2]\,\mathrm{E}[T \mid \text{rejected}]^2 + \mathrm{E}[N]\,\mathrm{Var}(T \mid \text{rejected}) \tag{B.15}$$

Calculating $\mathrm{E}[N^2]$ Since we know that, for any variable X,

$$\mathrm{E}[X^2] = \mathrm{Var}(X) + \mathrm{E}[X]^2 \tag{B.16}$$

and knowing that N follows a geometric distribution, $\mathrm{Var}(N) = (1-p)/p^2$ so

$$\mathrm{E}[N^2] = \frac{1-p}{p^2} + \frac{(1-p)^2}{p^2} = \frac{(1-p)(2-p)}{p^2} \tag{B.17}$$

Calculating $\mathrm{E}[T \mid \text{rejected}]$ Already determined in Eq. (B.12).

Calculating $\mathrm{Var}(T \mid \text{rejected})$ This is the most algebra-intensive factor we need to calculate. For brevity of our deduction process, consider the following definitions:

$$\mu = \mathrm{E}[T \mid \text{rejected}] \tag{B.18}$$
$$K = e^{\lambda t_{cr}} \tag{B.19}$$
$$k = K - 1 = e^{\lambda t_{cr}} - 1 \tag{B.20}$$
$$\beta = 1/\lambda \tag{B.21}$$
$$\Delta = \beta - \mu \tag{B.22}$$

We can now deduce the variance.

$$\begin{aligned}
\mathrm{Var}(T \mid \text{rejected}) =& \int_0^{t_{cr}} (t-\mu)^2 \, f_{T|\text{rejected}}(t)\, dt = \int_0^{t_{cr}} (t-\mu)^2 \, \frac{\lambda}{1-e^{-\lambda t_{cr}}} e^{-\lambda t}\, dt \\
=& \frac{(\mu^2\lambda^2 - 2\mu\lambda + 2)e^{\lambda t_{cr}} - (t_{cr}^2 - 2\mu t_{cr} + \mu^2)\lambda^2 + (2\mu - 2t_{cr})\lambda - 2}{\lambda^2(e^{\lambda t_{cr}} - 1)} \\
=& \frac{(\mu^2\lambda^2 - 2\mu\lambda + 2)K - (t_{cr}^2 - 2\mu t_{cr} + \mu^2)\lambda^2 + (2\mu - 2t_{cr})\lambda - 2}{\lambda^2(K - 1)} \\
=& \frac{\mu^2\lambda^2 K - 2\mu\lambda K + 2K - t_{cr}^2\lambda^2 + 2\mu t_{cr}\lambda^2 - \mu^2\lambda^2 + 2\mu\lambda - 2t_{cr}\lambda - 2}{\lambda^2(K - 1)} \\
=& \frac{(K-1)[(1-\mu\lambda)^2 + 1] + 1 - (t_{cr}\lambda + 1)^2 + 2\mu t_{cr}\lambda^2}{\lambda^2(K-1)} \\
=& \frac{(K-1)[(1-\mu\lambda)^2 + 1]}{\lambda^2(K-1)} + \frac{1 - (t_{cr}\lambda + 1)^2}{\lambda^2(K-1)} + \frac{2\mu t_{cr}\lambda^2}{\lambda^2(K-1)} \\
=& \frac{(1-\mu\lambda)^2 + 1}{\lambda^2} + \frac{2\mu t_{cr}}{K-1} + \frac{1 - (t_{cr}\lambda + 1)^2}{\lambda^2(K-1)} \\
=& \left(\frac{1}{\lambda} - \mu\right)^2 + \left(\frac{1}{\lambda}\right)^2 + \frac{2\mu t_{cr}}{K-1} + \frac{\left(\frac{1}{\lambda}\right)^2 - (t_{cr} + \frac{1}{\lambda})^2}{K-1} \\
=& (\beta - \mu)^2 + \beta^2 + \frac{2\mu t_{cr}}{K-1} + \frac{\beta^2 - (t_{cr} + \beta)^2}{K-1} \\
=& \Delta^2 + \beta^2 + \frac{\beta^2 - (t_{cr} + \beta)^2 + 2\mu t_{cr}}{K-1} \\
=& \Delta^2 + \beta^2 + \frac{\beta^2 - t_{cr}^2 - 2t_{cr}\beta - \beta^2 + 2\mu t_{cr}}{K-1} \\
=& \Delta^2 + \beta^2 - \frac{t_{cr}^2 + 2t_{cr}\beta - 2\mu t_{cr}}{K-1} \\
=& \Delta^2 + \beta^2 - t_{cr}\frac{t_{cr} + 2(\beta - \mu)}{K-1} \\
=& \Delta^2 + \beta^2 - \frac{1}{K-1} t_{cr}(t_{cr} + 2\Delta) \\
=& \Delta^2 + \beta^2 - \frac{1}{k} t_{cr}(t_{cr} + 2\Delta)
\end{aligned} \tag{B.23}$$

Final result Using the abbreviations defined in the previous section,

$$p = e^{-\lambda t_{cr}} = 1/e^{\lambda t_{cr}} = K^{-1} \tag{B.24}$$

$$\mathrm{E}[N] = \frac{1-p}{p} = \frac{1-K^{-1}}{K^{-1}} = K - 1 = k \tag{B.25}$$

$$\mathrm{E}[N^2] = \frac{(1-p)(2-p)}{p^2} = (K-1)(2K-1) = k(2k+1) \tag{B.26}$$

$$\mathrm{E}[T \mid \text{rejected}] = \frac{1}{\lambda} - t_{cr}\frac{1}{e^{\lambda t_{cr}} - 1} = \beta - \frac{t_{cr}}{k} \tag{B.27}$$

$$\mathrm{Var}(T \mid \text{rejected}) = \Delta^2 + \beta^2 - \frac{1}{k}t_{cr}(t_{cr} + 2\Delta) \tag{B.28}$$

We can therefore obtain the analytical form of $\mathrm{E}[W^2]$:

$$\begin{aligned}
\mathrm{E}[W^2] &= \mathrm{E}[N^2]\,\mathrm{E}[T \mid \text{rejected}]^2 + \mathrm{E}[N]\,\mathrm{Var}(T \mid \text{rejected}) \\
&= k(2k+1)\left(\beta - \frac{t_{cr}}{k}\right)^2 + k\left[\Delta^2 + \beta^2 - \frac{1}{k}t_{cr}(t_{cr} + 2\Delta)\right] \\
&= k(2k+1)\left(\beta - \frac{t_{cr}}{k}\right)^2 + k(\Delta^2 + \beta^2) - t_{cr}(t_{cr} + 2\Delta) \\
&= k\left[(2k+1)\left(\beta - \frac{t_{cr}}{k}\right)^2 + \Delta^2 + \beta^2\right] - t_{cr}(t_{cr} + 2\Delta)
\end{aligned} \tag{B.29}$$

Appendix C

DynaMIT supply simulation

This appendix shows some algebraic expansions for the estimate of t^* suggested by Ben-Akiva et al. (2002). Please refer to Sec. 4.4.2 for some context. The goal is to explore a possible reason why this estimate was chosen, as such a rationale is not made available by the authors.

$$
\begin{aligned}
q(t^*) &= q_0 + l(ct^* - m) \\
&= q_0 + lct^* - lm \\
&= q_0 + lc\frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right)}{v_u + cl\ln\left(e^{-v_u/cl} + \varepsilon\right)} - lm \\
&= \frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right)}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} - (ml - q_0) \\
&= \frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right)}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} - \frac{(ml - q_0)(v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right))}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} \\
&= \frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right) - (ml - q_0)(v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right))}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} \\
&= \frac{(ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right) - (ml - q_0)(v_u/cl) - (ml - q_0)\ln\left(e^{-v_u/cl} + \varepsilon\right)}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} \\
&= \frac{-(ml - q_0)(v_u/cl)}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)} \\
&= \frac{(q_0 - ml)(v_u/cl)}{v_u/cl + \ln\left(e^{-v_u/cl} + \varepsilon\right)}
\end{aligned}
\tag{C.1}
$$

$$\begin{aligned}
\lambda(t^*) &= \frac{-v_u}{q(t^*)} \\
&= \frac{-v_u}{\frac{(q_0-ml)(v_u/cl)}{v_u/cl+\ln(e^{-v_u/cl}+\varepsilon)}} \\
&= \frac{-v_u(v_u/cl+\ln(e^{-v_u/cl}+\varepsilon))}{(q_0-ml)(v_u/cl)} \\
&= \frac{-v_u}{q_0-ml}\frac{v_u/cl+\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl} \\
&= \frac{-v_u}{q_0-lm}\left(1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}\right)
\end{aligned} \tag{C.2}$$

$$\begin{aligned}
\lambda(t^*)t^* &= \frac{-v_u}{q_0-lm}\left(1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}\right)\frac{(ml-q_0)\ln(e^{-v_u/cl}+\varepsilon)}{v_u+cl\ln(e^{-v_u/cl}+\varepsilon)} \\
&= \frac{v_u}{ml-q_0}\left(1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}\right)\frac{(ml-q_0)\ln(e^{-v_u/cl}+\varepsilon)}{v_u+cl\ln(e^{-v_u/cl}+\varepsilon)} \\
&= v_u\left(1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}\right)\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u+cl\ln(e^{-v_u/cl}+\varepsilon)} \\
&= \left(1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}\right)\frac{\ln(e^{-v_u/cl}+\varepsilon)}{1+\frac{\ln(e^{-v_u/cl}+\varepsilon)}{v_u/cl}} \\
&= \ln(e^{-v_u/cl}+\varepsilon)
\end{aligned} \tag{C.3}$$

$$\begin{aligned}
x(t^*) &= e^{\lambda(t^*)t^*}\left(x_0+\frac{v_u}{\lambda(t^*)}\right)-\frac{v_u}{\lambda(t^*)} \\
&= e^{\ln(e^{-v_u/cl}+\varepsilon)}(x_0-q(t^*))+q(t^*) \\
&= (e^{-v_u/cl}+\varepsilon)(x_0-q(t^*))+q(t^*) \\
&= (e^{-v_u/cl}+\varepsilon)x_0-(e^{-v_u/cl}+\varepsilon)q(t^*)+q(t^*) \\
&= (e^{-v_u/cl}+\varepsilon)x_0-(e^{-v_u/cl}-1+\varepsilon)q(t^*)
\end{aligned} \tag{C.4}$$

$$\begin{aligned}
x(t^*)-q(t^*) &= (e^{-v_u/cl}+\varepsilon)x_0-(e^{-v_u/cl}-1+\varepsilon)q(t^*)-q(t^*) \\
&= (e^{-v_u/cl}+\varepsilon)x_0-(e^{-v_u/cl}+\varepsilon)q(t^*) \\
&= (e^{-v_u/cl}+\varepsilon)(x_0-q(t^*))
\end{aligned} \tag{C.5}$$

Therefore, a certain value of ε causes $x(t^*)-q(t^*)$ (distance to queue at $t=t^*$) to be a small value when compared to $x_0-q(t^*)$ (distance between initial position and queue at $t=t^*$), where $x_0-q(t^*)$ can be considered to represent the order of magnitude of the values we are considering, and as such that the ratio $e^{-v_u/cl}+\varepsilon$ should be small enough that $x(t^*)-q(t^*)$ and $x_0-q(t^*)$ are separated by a few degrees of magnitude.

Appendix D

Parameter estimation

This appendix complements Sec. 8.1 by determining estimates for the parameters required by the dynamic model described in Chap. 8.

D.1 Vehicle gap

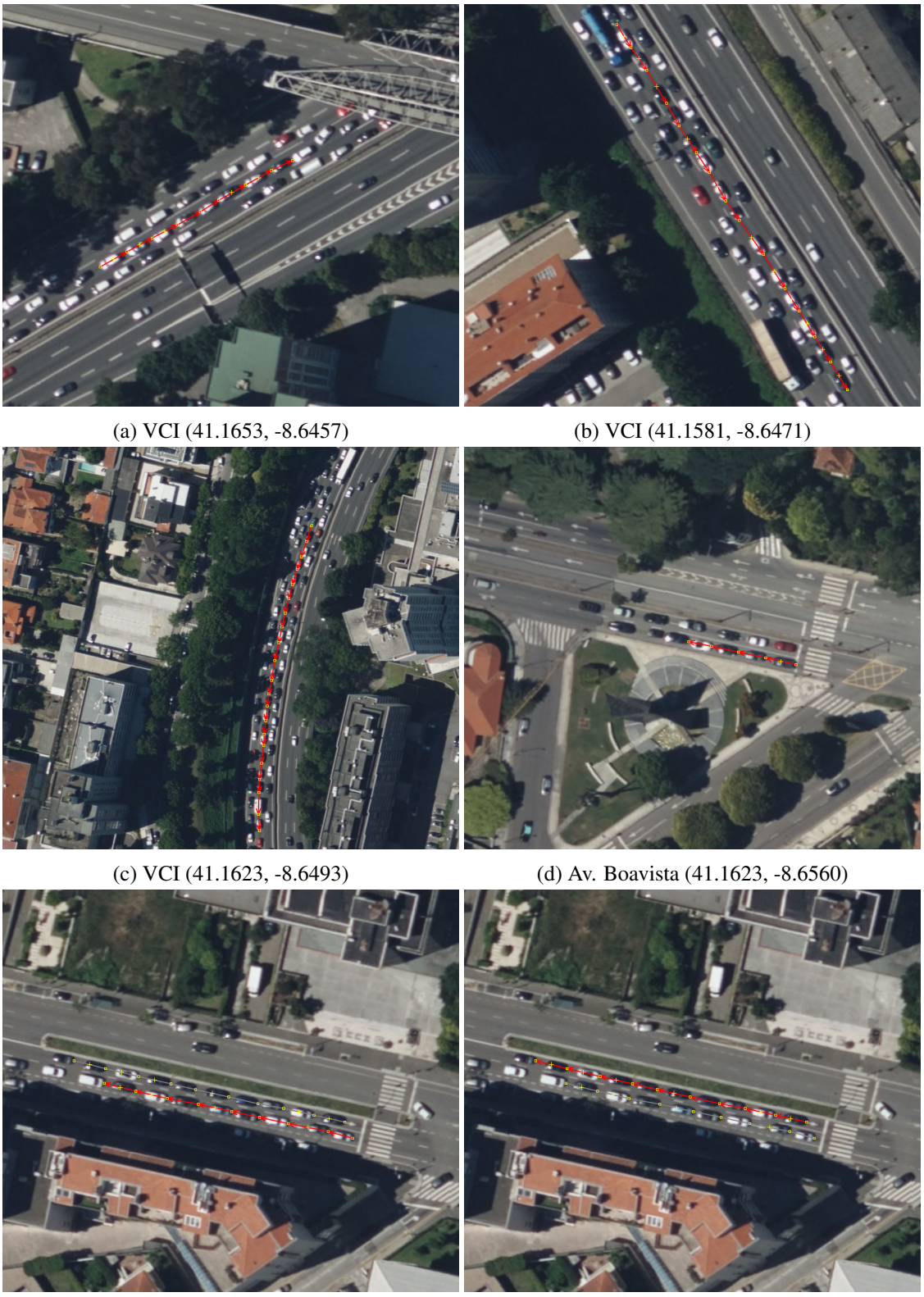
In this section, the average vehicle gap l_v is estimated. See Sec. 8.1 for the definition of average vehicle gap.

It is being assumed that all vehicles are light passenger or cargo vehicles (including, but not limited to, minicars, sedans, minivans and smaller SUVs, but notably excluding larger American-type SUVs and pick-up trucks) since this is the most common and representative vehicle mix in Europe.

To determine the average vehicle gap, JOSM (Scholz and Stöcker, 2023) was used to load Bing aerial imagery (Microsoft and Vexcel Imaging, 2023); we then visually identified six zones with traffic jams containing mostly light vehicles and manually scanned vehicle coordinates using as reference the front of each vehicle, Fig. D.1. The length and number of vehicles in each measurement can be seen in Tab. D.1. By taking the total length of all measurements $410.93\,\mathrm{m}$ and dividing it by the total number of vehicles 63, we obtain the average vehicle length plus gap of $6.52\,\mathrm{m}$.

Table D.1: Gap length measurements' results.

Measurement	Total length (m)	Number of vehicles	Length per vehicle (m)
1	47.21	8	5.90
2	92.61	14	6.62
3	133.42	20	6.67
4	23.54	4	5.89
5	54.57	8	6.82
6	59.58	9	6.62
Total	**410.93**	**63**	**6.52**



(a) VCI (41.1653, -8.6457)

(b) VCI (41.1581, -8.6471)

(c) VCI (41.1623, -8.6493)

(d) Av. Boavista (41.1623, -8.6560)

(e) Av. Boavista (41.1601, -8.6427)

(f) Av. Boavista (41.1602, -8.6427)

Figure D.1: Vehicle length and gap measurements in the city of Porto.

D.2 Queue dissipation speed

In this section, the queue dissipation speed v_q is estimated. See Sec. 8.1 for the definition of queue dissipation speed.

Using the default value for base saturation flow rate of $1900\,\mathrm{h}^{-1}$ provided in the TRB (2000, Exhibit 10-12) without any adjustments as the default value of capacity for a junction as a queue is dissipating, we can infer that the queue dissipation speed is $v_q = v_l \cdot 1900\,\mathrm{h}^{-1}$, which using the value of l_v calculated in appendix D.1, gives $v_q = 3.44\,\mathrm{m/s} = 12.38\,\mathrm{km/h}$. This value also happens to be consistent with the range of recommended speeds for a manual drive passenger car in 1st gear, $8\,\mathrm{km/h}$ to $15\,\mathrm{km/h}$.

It should be noted that this value is naturally an overestimate of the queue dissipation speed, since additional adjustments should be made to account for the initial delay associated with reaction times and initial vehicle acceleration.

Appendix E

Movement equations under uncongested flow

This appendix lays out an alternative, more realistic model of movement under uncongested flow with respect to the model from Sec. 8.2.1.

The conditions that the model must meet are:

1. A vehicle travels as fast as legally allowed on a road.
2. When a vehicle brakes, it always breaks at a constant acceleration a_b. Similarly, when a vehicle accelerates, it always accelerates at a constant acceleration a_a.
3. When a vehicle approaches a stopped traffic queue, it must brake to enter the queue at speed $v = 0\,\mathrm{m/s}$. Similarly, when a vehicle approaches a dissipating traffic queue, it must enter it at speed $v = v_q$.

E.1 On driver behavior in the face of speed limits

It is customary that the speed limit mandated by a traffic sign applies to the stretch of road starting at the point where the sign is located and ending at the next junction, so it is the drivers' responsibility to take the necessary precautions to make sure they can decelerate in a safe manner so as to enter a speed-limited stretch of road at no more than the posted speed limit. By law, drivers are not required to know the posted speed limits for all upcoming stretches, but they are required to remember all indications given by signs located in stretches they already crossed, and are required to take into account upcoming traffic signs within their field of vision, and take all necessary precautions within reason so as not to break any laws.

In practice, a cautious driver starts taking precautions when:

- They see the speed limit sign, to which a driver must respond by gradually reducing their speed to reach the location of the sign at no more than the posted speed limit; or

- They notice a change in road design that hints at a change in road classification, which may imply different speed limits (such as when entering or leaving urban areas, where different general speed limits apply); or
- By looking at complementary signs (such as end of motorway sign), to which a driver must respond by exercising caution and preferably reducing speed so that they can easily adjust to a lower speed limit.

Drivers can readily apply this sort of empirical knowledge, and when confronted with situations that are out of the ordinary, drivers are expected to resort to common sense to deduce how they should act when faced with complex situations where there may be unusual or conflicting traffic sign layouts. Consider the common scenario of a motorway offramp with an auxiliary lane, where the first sign indicates the posted speed limit on the auxiliary lane is $60\,\mathrm{km/h}$ and the second sign indicates the posted speed limit on the offramp is $40\,\mathrm{km/h}$, Fig. E.1. But now consider that the auxiliary lane is too short such that a driver traveling at $60\,\mathrm{km/h}$ when they pass the first sign would not be able to safely slow down to $40\,\mathrm{km/h}$ before reaching the second sign. A reasonable driver is expected to drive slower than $60\,\mathrm{km/h}$ through the first sign, because their common sense gives them the necessary foresight to make that decision.

Encoding this sort of logic in a computer program is a challenging matter, because humans are much better than computers at responding to unforeseen circumstances. We could then devise a complex method to find the maximum speed that is allowed at each point so that condition 1 is met in a strict sense. We define speed limits in upcoming stretches of road as *obstacles* to the maximum speed of a vehicle. We consider an obstacle o_1 to be *more strict* than another obstacle o_2 iff all possible states that meet o_2 also meet o_1, but there is at least one state that meets o_1 but not o_2. For the sake of clarity, o_1 may be the $60\,\mathrm{km/h}$ limit on the auxiliary lane, and o_2 may be the $40\,\mathrm{km/h}$ limit on the offramp; as can be intuitively deduced, o_2 is more strict than o_1 because a driver must have the foresight to take measures in advance to obey o_2 instead of just obeying o_1 and then having to perform emergency braking.

The goal of the computational task is to find the obstacle o that the driver should react to first, given the current time is t:

1. Find the first obstacle o_1 and set $o \leftarrow o_1$; assume the obstacle occurs at time t_1, and that after abiding to that obstacle, the driver starts braking as intensely as practically possible and safe, with deceleration a_b; consider that the vehicle comes to a full stop at time t'_1.
2. Give the driver some foresight, by having it consider all obstacles between t_1 and t'_1, and if there is any obstacle o_i that is more strict than o_1 after t_1, then assign $o \leftarrow o_i$, and repeat this step until o does not change between iterations.

To visualize this algorithm, consider the scenarios presented in Fig. E.2, inspired on Fig. E.1 but with differing lengths of the auxiliary lane. To simplify the diagram, we consider vehicles brake in such a way that speed reduces linearly with distance. In each scenario there are three stretches of road, with speed limits 90, 60 and $40\,\mathrm{km/h}$, but the length of the auxiliary lane is

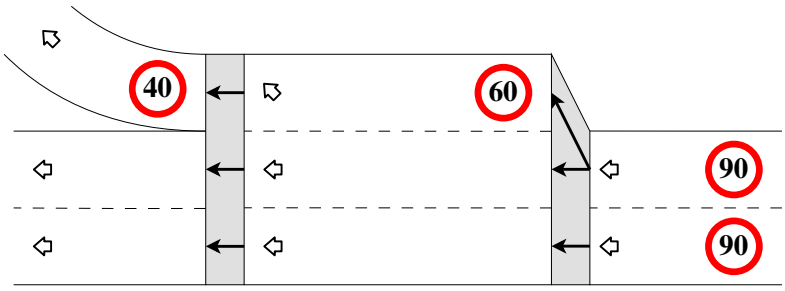


Figure E.1: Scenario of highway and offramp with progressively lower speed limits.

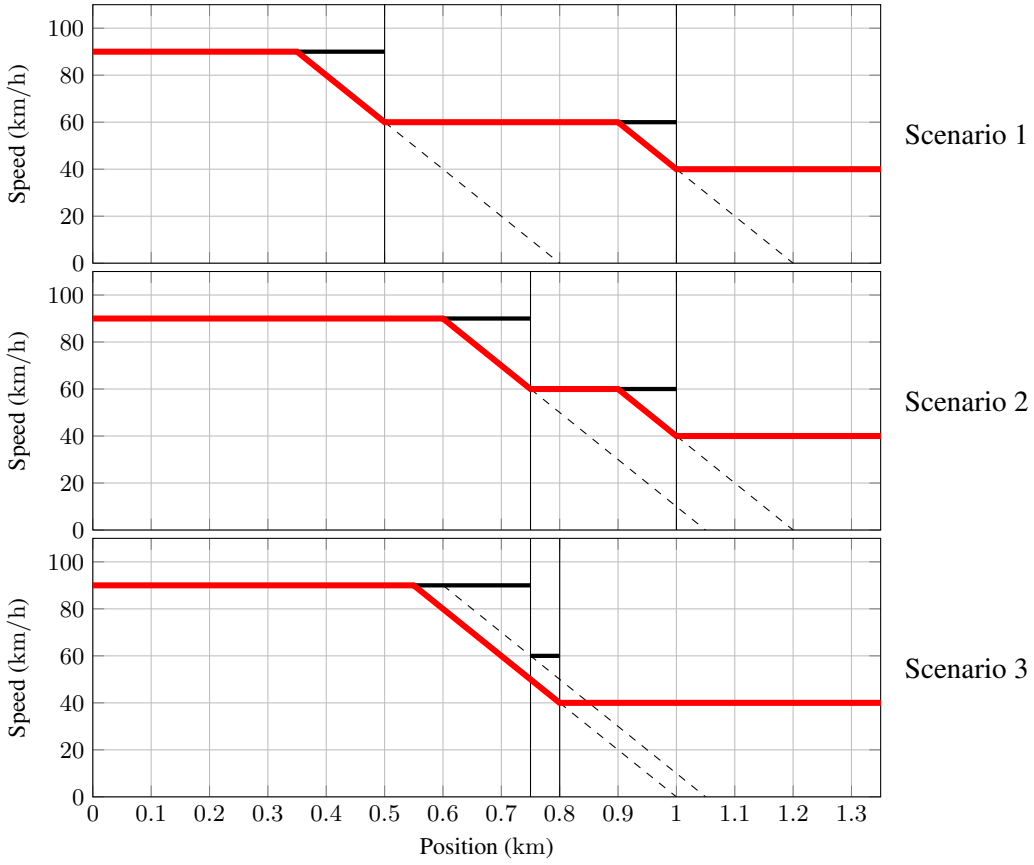


Figure E.2: Three scenarios where different speed limit layouts affect vehicle speed profiles. The red curve is the speed profile of a vehicle that always obeys speed limits while traveling as fast as possible. The horizontal black lines are the speed limits at each stretch of road. For instance, in scenario 1 there are three stretches, with lengths 0.5, 0.5 and $0.4\,\mathrm{km}$ and speed limits 90, 60 and $40\,\mathrm{km/h}$ respectively

different in each scenario. Obstacle o_1 is the first vertical black line, and obstacle o_2 is the second vertical black line. In the first scenario, the two obstacles are far enough apart that they do not need to be analysed together. In the second scenario, the two obstacles are close enough that they need to be analysed together (since starting to brake at o_1 would not guarantee that o_2 is met if we only knew the time at which o_2 occurs), but still none of the obstacles is more strict than the other since o_1 is not skipped. In the third scenario, the two obstacles are very close, and if the driver

started braking only to satisfy o_1 they would not be able to satisfy o_2, so this concept of foresight forces the driver to ignore o_1 and instead satisfy o_2 only, because o_2 is more strict than o_1.

The described process supports the idea that implementing such an algorithm would be a complex task when compared with the benefits therein. There are also three simple arguments that make such an algorithm unnecessary:

- It can be argued that many people only start taking measures to match their speed with the legal limit after passing the sign, and as such, there is a brief period of time when people are breaking legal limits. But because such an attitude is not considered to show deliberate criminal intent, this is rarely a problem for drivers, so this principle may very well be adopted in the model.

- Highway features are carefully designed, and signs are equally carefully placed so as to maximize the level of service as well as safety. Therefore, it is a common safety feature for speed limits to be posted before the driver arrives at the stretch of road that actually needs those speed limits. This is also one of the reasons the behavior in the previous item is common, since drivers are aware that highways and signs are engineered such that there is a certain safety margin, and as such, drivers already know that, even if they see a speed limit sign, there is some margin for drivers to only take measures after the sign.

- On the same subject of highway engineering: highways and speed limits are engineered such that the user experience is consistent and makes sense, so it is highly unlikely that a highway would be designed such that a driver would not be allowed enough time or space to safely brake while still mostly traveling at the maximum legally allowed speeds. Emergency braking and similar events in simulated environments are most often owed to poor network data quality than to poor engineering of the real infrastructure.

It was therefore considered that this issue did not have enough importance for a foresight mechanism to be considered in this hypothetical model. As such, driver behavior with respect to speed limits is modeled by having the driver slow down to match the speed limit of the next edge, ignoring any speed limits of stretches of road further down the trip other than the next edge. The speed limit of the next edge can be calculated taking into account the possibility of a long queue such that a vehicle needs to enter the following edge at a low speed so as to join the queue at an appropriate speed.

E.2 The movement model

A vehicle is assumed to enter an edge at the recommended speed for that edge, taking into account a possible queue at that edge. The behavior vehicles should follow is that they should accelerate at a rate of a_a while they can, maintain constant speed when the vehicle reaches the speed limit, and then decelerate at a rate of a_b to match the speed of the next edge (or speed of the end of the queue).

In a scenario where the end of the queue is moving at a speed of v_q (which may be positive or zero), the procedure is as follows:

1. Assume the vehicle is initially traveling at speed v_i, accelerates at a rate a_a during time period t_a until it reaches the speed limit v_m, maintains that speed during time period t_m, and then decelerates from speed v_m to speed v_f at a rate a_b during time period t_b.

$$\begin{cases} q + v_q(t_a + t_m + t_b) &= \dfrac{v_i + v_m}{2} t_a + v_m t_m + \dfrac{v_m + v_f}{2} t_b \\ v_m &= v_i + a_a t_a \\ v_f &= v_m + a_b t_b \end{cases} \tag{E.1}$$

$$\iff \begin{cases} t_a &= \dfrac{v_m - v_i}{a_a} \\ t_b &= \dfrac{v_f - v_m}{a_b} \\ t_m &= \dfrac{t_a(v_i + v_m - 2v_q) + t_b(v_m + v_f - 2v_q) - 2q}{2v_q - 2v_m} \end{cases} \tag{E.2}$$

2. In the previous item it was assumed the vehicle had enough time to reach the speed limit. But it might not have time, in which case t_m will be negative. In this case, the speed increases from v_i to v_m at a rate of a_a, and then decreases from v_m to v_f at a rate of a_b (note that now v_m is not the speed limit, but the maximum achievable speed, which is lower than the speed limit).

$$\begin{cases} q + v_q(t_a + t_b) &= \frac{v_i + v_m}{2} t_a + \frac{v_m + v_f}{2} t_b \\ v_m &= v_i + a_a t_a \\ v_f &= v_m + a_b t_b \end{cases} \tag{E.3}$$

$$\iff \begin{cases} q + v_q(t_a + t_b) &= \dfrac{v_i + v_m}{2} t_a + \dfrac{v_m + v_f}{2} t_b \\ t_a &= \dfrac{v_m - v_i}{a_a} \\ t_b &= \dfrac{v_f - v_m}{a_b} \end{cases} \tag{E.4}$$

$$\implies q + v_q \left(\frac{v_m - v_i}{a_a} + \frac{v_f - v_m}{a_b} \right) = \frac{v_i + v_m}{2} \frac{v_m - v_i}{a_a} + \frac{v_m + v_f}{2} \frac{v_f - v_m}{a_b} \tag{E.5}$$

$$\iff q + \frac{v_q v_m}{a_a} - \frac{v_q v_i}{a_a} + \frac{v_q v_f}{a_b} - \frac{v_q v_m}{a_b} = -\frac{v_i^2}{2a_a} + \frac{v_m^2}{2a_a} - \frac{v_m^2}{2a_b} + \frac{v_f^2}{2a_b} \tag{E.6}$$

$$\iff \frac{v_m^2}{2a_a} - \frac{v_m^2}{2a_b} - \frac{v_q v_m}{a_a} + \frac{v_q v_m}{a_b} - \frac{v_i^2}{2a_a} + \frac{v_f^2}{2a_b} + \frac{v_q v_i}{a_a} - \frac{v_q v_f}{a_b} - q = 0 \tag{E.7}$$

$$\iff v_m^2 \left[\frac{1}{2a_a} - \frac{1}{2a_b} \right] + v_m \left[-\frac{v_q}{a_a} + \frac{v_q}{a_b} \right] + \tag{E.8}$$

$$\left[-\frac{v_i^2}{2a_a} + \frac{v_f^2}{2a_b} + \frac{v_q v_i}{a_a} - \frac{v_q v_f}{a_b} - q \right] = 0 \tag{E.9}$$

The final system is

$$\begin{cases} a & = \frac{1}{2}\left(\frac{1}{a_a} - \frac{1}{a_b}\right) \\ b & = -v_q\left(\frac{1}{a_a} - \frac{1}{a_b}\right) \\ c & = \frac{v_i}{a_a}\left(v_q - \frac{v_i}{2}\right) - \frac{v_f}{a_b}\left(v_q - \frac{v_f}{2}\right) - q \\ v_m & = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\ t_a & = \frac{v_m - v_i}{a_a} \\ t_b & = \frac{v_f - v_m}{a_b} \end{cases} \tag{E.10}$$

Given the preconditions $a_a > 0$, $a_b < 0$, $0 < v_i, v_f < v_m$, we can assert that:

- $a > 0$
- $b < 0$
- Because we are assuming there is enough space to accelerate, keep speed and decelerate, q is large enough that $c < 0$
- $-4ac > 0$, so $\Delta = b^2 - 4ac > b^2$, so $\sqrt{b^2 - 4ac} > -b$
- This means the minus case is not possible, as otherwise v_m would be negative, since we already saw that $\sqrt{b^2 - 4ac} > -b$. As such, the only valid case in the context of this problem is the plus case.

E.2.1 Lane changing

A simple lane-changing model is also presented in alternative to the option of not adding any penalties in the model that was actually implemented (refer to Sec. 8.2.4).

This lane-changing model is a simple gap acceptance model as laid out in appendix B. A vehicle waits until a suitable gap in the destination lane is available, and the average wait time is given by Eq. (B.13), reproduced here for convenience:

$$\mathrm{E}[W] = \frac{1}{\lambda}(e^{\lambda t_{cr}} - 1) - t_{cr}$$

where λ is the flow in the destination lane. Typical values for t_{cr} are in range of $0.7\,\mathrm{s} to 1.0\,\mathrm{s}$.

This model is only applied if the driver is changing onto a section of the destination lane where traffic is moving; if the driver is trying to change onto a section of the destination lane where traffic is stopped, the vehicle travels to the end of the queue and then waits for a chance to change lanes.

Appendix F

Order-statistic queue

First of all, it is important to clarify why some data structures can be called *order-statistic*. An order-statistic data structure is an augmented data structure (Cormen et al., 2009, p. 339) that allows the following two operations:

- `rank(v)` or `order_of(v)`: returns the *rank* of value v in the data structure;
- `select(i)` or `at(i)`: returns the i-th value in the data structure.

For instance, an order-statistic set can be implemented as an extension of a binary tree that, besides the typical `insert(v)`, `erase(v)` and `find(v)`, also implements `order_of(v)` and `at(i)`.

In the case of the present work, it was relevant to have an implementation of an order-statistic queue that provides operations `push(v)`, `top()`, `pop()` and `order_of(v)`, as detailed in Sec. 8.2.2 (and also `at(i)` just because it is easily obtainable).

The basic queue functionalities and the `at(i)` operation are already implemented in the C++ `std::deque`, where all operations have a constant amortized time complexity, so all that is left is to implement `order_of(v)`.

This can be done by asserting that each element ever inserted in the queue has a unique, increasing ID. The ID of the first element that is pushed into the queue is 0. The ID of the first value in the queue is also tracked, which is designated $start$. Every time the queue is popped, $start$ is incremented to reflect that the value now at the front of the queue is the value with the next ID.

To implement `order_of(v)`, a separate map (using the C++ data structures `std::map` or `std::unordered_map`) is required, where each value v is mapped to its ID. This map can then be used to find the position of the element in the queue by subtracting the ID obtained from the map and the ID of the first element of the queue, $start$. This yields for operation `order_of(v)` a time complexity similar to that of getting an element from the map, which can be either $O(\log N)$ if using `std::map`, or $O(1)$ amortized if using `std::unordered_map`.

The pseudocode of the order-statistic queue is presented in List. F.1:

Listing F.1 Order-statistic queue

```
1:  q.deque = ⟨⟩
2:  q.map = {}
3:  q.start = 0
4:  function PUSH(q, v)
5:      i ← start + SIZE(q.deque)
6:      PUSHBACK(q.deque, v)
7:      SET(q.map, v, i)
8:  function TOP(q)
9:      return FRONT(q.deque)
10: function POP(q)
11:     v ← FRONT(q)
12:     ERASE(q.map, v)
13:     POPFRONT(q.deque)
14:     ++start
15: function AT(q, i)
16:     return AT(q.deque, i)
17: function ORDEROF(q, v)
18:     i ← GET(q.map, v)
19:     return i − q.start
```