Universitat Politècnica de Catalunya
Department of Statistics and Operations Research

A Mesoscopic Traffic Simulation Based Dynamic Traffic Assignment

Thesis Presented to Obtain the Qualification of Doctor

from the Universitat Politècnica de Catalunya

PhD Candidate:
M$^{\underline{a}}$Paz Linares Herreros

Advisor:
Jaume Barceló Bugeda, PhD

February 27, 2014

Abstract

In terms of sustainability, traffic is currently a significant challenge for urban areas, where the pollution, congestion and accidents are negative externalities which have strongly impacted the health and economy of cities. The increasing use of private vehicles has further exacerbated these problems.

In this context, the development of new strategies and policies for sustainable urban transport has made transport planning more relevant than ever before. Mathematical models have helped greatly in identifying solutions, as well as in enriching the process of making decisions and planning. In particular, dynamic network models provide a means for representing dynamic traffic behavior; in other words, they provide a temporally coherent means for measuring the interactions between travel decisions, traffic flows, travel time and travel cost. This thesis focuses on dynamic traffic assignment (DTA) models.

DTA has been studied extensively for decades, but much more so in the last twenty years since the emergence of Intelligent Transport Systems (ITS). The objective of this research is to study and analyze the prospects for improving this problem.

In an operational context, the objective of DTA models is to represent the evolution of traffic on a road network as conditions change. They seek to describe the assignment of the demand on different paths which connect every OD pair in a state of equilibrium.

The behaviour following each individual decision during a trip is a time-dependent generalization of Wardrop's First Principle, the Dynamic User Equilibrium (DUE). This hypothesis is based on the following idea: When current travel times are equal and minimal for vehicles that depart within the same time interval , the dynamic traffic flow through the network is in a DUE state based on travel times for each OD pair at each instant of time (Ran and Boyce (1996)).

This work begins with the time-continuous variational inequalities model proposed by Friesz et al. (1993) for solving the DUE problem. Different solutions can be used on the proposed DUE formulation. On the one hand, there are the so-called analytical approaches which use known mathematical optimization techniques for solving the problem directly. On the other hand, there are simulation-based formulations that approximate heuristic solutions at a reasonable computational cost. While analytical models concentrate mainly on deriving theoretical insights, simulation-based models focus on trying to build practical models for deployment in real networks. Thus, because the simulation-based formulation holds the most promise, we work on that approach in this thesis.

In the field of simulation-based DTA models, significant progress has been made by many researchers in recent decades. Our simulation-based formulation separates the proposed iterative process into two main components, which have been systematized by Florian et al. (2001) as follows:

- A method for determining the new time dependent path flows by using the travel times on these paths experienced in the previous iteration.
- A dynamic network loading (DNL) method, which determines how these paths flow propagate along the corresponding paths.

The flow reassignment algorithms can be grouped into two categories: preventive and reactive. However, it is important to note that not all computer implementations based on this algorithmic framework provide solutions that obtain DUE. Therefore, while we analyze both proposals in this thesis we focus on the preventive methods of flow reassignment because only those can guarantee DUE solutions.

Our proposed simulation-based DTA method requires a DNL component that can reproduce different vehicle classes, traffic light controls and lane changes. Therefore, this thesis develops a new multilane multiclass mesoscopic simulation model with these characteristics, which is embedded into the proposed DUE framework.

Finally, the developed mesoscopic simulation-based DTA approach is validated accordingly. The results obtained from the computational experiments demonstrate that the developed methods perform well.

Keywords: dynamic traffic assignment, dynamic user equilibrium, variational inequalities, mesoscopic traffic simulation, Method of Successive Averages.

Resumen

En los últimos tiempos, el problema del tráfico urbano ha situado a las áreas metropolitanas en una difícil situación en cuanto a sostenibilidad se refiere (en términos de la congestión, los accidentes y la contaminación). Este problema se ha visto acentuado por la creciente movilidad promovida por el aumento del uso del vehículo privado. Además, debido a que la mayor parte del tráfico es canalizada a través de los modos de carretera, el tiempo perdido por los usuarios al realizar sus viajes tiene un importante efecto económico sobre las ciudades.

En este contexto, la planificación de transporte se vuelve relevante a través del desarrollo de nuevas estrategias y políticas para conseguir un transporte urbano sostenible. Los modelos matemáticos son de gran ayuda ya que enriquecen las decisiones de los gestores de tráfico en el proceso de planificación. En particular podemos considerar los modelos de tráfico para la predicción, como por ejemplo los modelos de asignación dinámica de tráfico (ADT), los cuales proveen de una representación temporal coherente de las interacciones entre elecciones de tráfico, flujos de tráfico y medidas de tiempo y coste. Esta tesis se centra en los modelos ADT.

Durante las últimas décadas, los modelos ADT han sido intensamente estudiados. Este proceso se ha acelerado particularmente en los últimos veinte años debido a la aparición de los Sistemas Inteligentes de Transporte. El objetivo de esta investigación es estudiar y analizar diferentes posibilidades de mejorar la resolución del problema.

En un contexto operacional, el objetivo de los modelos ADT es representar la evolución de la red urbana cuando las condiciones de tráfico cambian. Estos modelos tratan de describir la asignación de la demanda en los diferentes caminos que conectan los pares OD siguiendo un estado de equilibrio.

En este caso se ha considerado que el comportamiento de los conductores en cada una de sus decisiones individuales tomadas durante el viaje es una generalización dependiente del tiempo del Primer Principio de Wardrop, denominada Equilibrio Dinámico de Usuario (EDU). Esta hipótesis se basa en la siguiente idea: para cada par OD para cada instante de tiempo, si los tiempos de viaje de todos los usuarios que han partido en ese intervalo de tiempo son iguales y mínimos, entonces el flujo dinámico de tráfico en la red se encuentra en un estado de EDU basado en los tiempos de viaje (Ran and Boyce (1996)).

El presente trabajo toma como punto de partida el modelo de inecuaciones variacionales continuo en el tiempo propuesto por Friesz et al. (1993) para resolver el problema de equilibrio dinámico de usuario. Por un lado, se encuentran los denominados enfoques analíticos que utilizan técnicas matemáticas de optimización para resolver el problema directamente. Por otro lado, están los modelos cuyas formulaciones están basadas en simulación que aproximan soluciones heurísticas con un coste computacional razonable. Mientras que modelos analíticos se concentran principalmente en demostrar las propiedades teóricas, los modelos basados en simulación se centran en intentar construir modelos que sean prácticos para su utilización en redes reales. Así pues, debido a que las formulaciones basadas en simulación son las que se muestran más prometedoras a la práctica, en esta tesis se ha elegido este enfoque para tratar el problema ADT.

En los últimos tiempos, el campo de los modelos ADT basados en simulación ha sido de especial interés. Nuestra formulación basada en simulación consiste en un proceso iterativo que consta de dos componentes principales, sistematizadas por Florian et al. (2001) como sigue:

- Un método para determinar los nuevos flujos (dependientes del tiempo) en los caminos utilizando los tiempos de viaje experimentados en esos caminos en la iteración previa.
- Un procedimiento de carga dinámica de la red (CDR) que determine cómo esos flujos se propagan a través de sus correspondientes caminos.

Los algoritmos de reasignación de flujo pueden ser agrupados en dos categorías: preventivos y reactivos. Es importante notar aquí que no todas las implementaciones computacionales basadas en el marco algorítmico propuesto proporcionan una solución EDU. Por lo tanto, aunque en esta tesis analizamos ambas propuestas, nos centraremos en los métodos preventivos de reasignación de flujo porque son los que nos garantizan alcanzar la hipótesis considerada (EDU).

Además, nuestro modelo ADT basado en simulación requiere de una componente de CDR que pueda reproducir diferentes clases de vehículos, controles semafóricos y cambios de carril. Así, uno de los objetivos de esta tesis es desarrollar un nuevo modelo de simulación de tráfico con dichas características (multiclase y multicarril), teniendo en cuenta que será una de las componentes principales del marco ADT propuesto.

Finalmente, el modelo ADT basado en simulación mesoscópica es validado en consecuencia. Los resultados obtenidos a partir de las experiencias computacionales realizadas demuestran el buen comportamienyo de los métodos desarrollados.

Palabras clave: asignación dinámica de tráfico, equilibrio dinámico de usuario, inecuaciones variacionales, simulación mesoscópica de tráfico, Método de las Medias Sucesivas.

Acknowledgements

A lo largo de los más de cinco años en los que he estado viviendo la aventura de hacer una tesis doctoral, muchos han sido los que me han apoyado, motivado o "simplemente" estado ahí. Seguro que no estáis todos los que sois, pero también seguro que aquellos que faltéis sabréis perdonarme.

Mi andadura en el mundo de la optimización y la simulación de tráfico no habría sido posible sin la confianza, que hace ya casi nueve años, Jaume Barceló depositó en mí. Gracias a mi director de tesis por creer en mis capacidades y por darme la oportunidad de investigar en este campo con tantos problemas aún por resolver.

My stay in Stockholm would not have been successful if it had not been for the professors and the PhD students from the Division of Traffic and Logistics at KTH. I thank specially Haris Koutsopoulos, Wilco Burghout and Carlos Morán for their welcome, assistance, advice and amusing company both at work and beyond.

To Matt Elmore and the external referees, I very much appreciate the time and effort you have invested in reviewing my work.

Llegados a este punto es el momento de admitir que aunque en la portada sólo aparezca una autora, este trabajo, al igual que muchos otros, es el producto de la interacción de un gran equipo sin el apoyo del cual no hubiera sido posible. Gracias a Helena, Gemma, Manuel y Mazas por su paciencia y apoyo. A Oriol por su comprensión y sus críticas siempre tan acertadas; y a Carlos, resumiéndolo en dos palabras, por todo.

Además quiero incluir en estas letras a todas las personas que me han acompañado a lo largo del camino. Particularmente y con especial cariño a las siguientes.

A mi trío de Crises por estar ahí. Cris Corchero por entenderme desde la distancia y estar atenta, foto del petit en mano, para ponerme una sonrisa en la cara cuando la he necesitado. Cristi por escuchar mi sinfín de lamentaciones y animarme a continuar perseverante hacia delante. Y Cris Montañola por compartir codo con codo conmigo esta última interminable etapa (mural con avatares incluido).

A mis primeros compañeros en el camino: los colegas, profesores y compañeros doctorandos del Departamento de Estadística e Investigación Operativa. Gracias por prestarme vuestra ayuda siempre que la he necesitado.

A mis últimos compañeros en esta aventura: todas las personas que forman ese gran equipo que es el Inlab. Gracias por acogerme en vuestro clan y hacerme sentir dentro desde el primer día. Mención especial a los que me han aguantado (día sí, día también) cuando me encontraba con dificultades, las solucionaba y las superaba: Vane, Josefran, Marta, Monty, Germán, Manu, Alberto, Gilbert, Ton y Sergio.

A mis amigos y familia. Porque aunque al final uno no quiere escuchar de nuevo esa pregunta, se agradecen los cientos de "¿Cómo va la tesis?". El interés y el cariño de todos me han empujado a seguir día a día. Gracias Alba y Pablo por vuestras toneladas de chocolate y cariño siempre a mi disposición, y Mónica y Javi por poder contar con vosotros en todo momento. Yoli, gracias por entenderme y por tus energías.

A mi hermano por marcarme un buen camino a seguir. La motivación de ser cómo él me ha valido para mejorar, me empuja en el presente y me seguirá dando impulso en el futuro. A Chieko por cuidar de él y formar parte de mi familia.

A mi madre por provocar en mí desde pequeña las ganas de aprender. Leer el libro del Michu o contar con garbanzos fueron mi gran comienzo en esto de vivir aprendiendo.

A mi padre por creer en mí y luchar por conseguir que sus hijos pudieran ser quienes quisieran ser, sin limitaciones.

Y por último, a Abel por ayudarme, apoyarme, inspirarme, darme calma y sobretodo escucharme durante horas y horas hablar de coches que cambian de carril o de cómo funcionan las rotondas. Como siempre te digo: eres un sol!

¡Gracias a todos!

M$^{\underline{a}}$Paz Linares Herreros

Barcelona, February 2014

Contents

List of Figures

List of Tables

Chapter 1

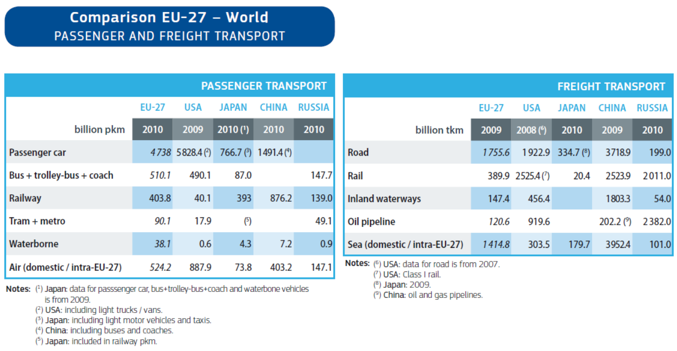
Introduction

1.1 Background

Urban traffic is a major problem that affects the quality of life for cities and metropolitan areas in developed countries. The problem has been exacerbated in recent decades by increased mobility stemming from the use of private vehicles. Furthermore, the European generalization of the diffused urban model (characteristically American) has relocated populations, trade and services to the nearby suburbs while also relegating industrial activity to the environs. As a result, the traditional city center has been reduced to a symbolic function. Subsequently, mobility is expanding in terms of both the number as well as the amplitude of journeys taken.

Although many modes of transport exist (train, waterways, flight, cycling and walking), most traffic consists of vehicles on the road network. Figure 1.1.1 shows published European Commission data on passenger and freight transport in EU-27 for all of 2012. According to this report, 83% of passenger transport (75% passenger vehicle and 8% bus, trolleybus or coach) and 45% of freight transport were channeled through urban roads modes.

The expansion described above, combined with the primary use of roads, has aggravated the issue of sustainability for urban areas, in which pollution, congestion and accidents are negative externalities that strongly impact their health and economy.



Comparison EU-27 – World
PASSENGER AND FREIGHT TRANSPORT

PASSENGER TRANSPORT

	EU-27	USA	JAPAN	CHINA	RUSSIA
billion pkm	2010	2009	2010 (1)	2010	2010
Passenger car	*4738*	5828.4 (2)	766.7 (3)	1491.4 (4)	
Bus + trolley-bus + coach	*510.1*	490.1	87.0		147.7
Railway	403.8	40.1	393	876.2	139.0
Tram + metro	*90.1*	17.9	(5)		49.1
Waterborne	*38.1*	0.6	4.3	7.2	0.9
Air (domestic / intra-EU-27)	*524.2*	887.9	73.8	403.2	147.1

Notes: (1) Japan: data for passsenger car, bus+trolley-bus+coach and waterbone vehicles is from 2009.
(2) USA: including light trucks / vans.
(3) Japan: including light motor vehicles and taxis.
(4) China: including buses and coaches.
(5) Japan: included in railway pkm.

FREIGHT TRANSPORT

	EU-27	USA	JAPAN	CHINA	RUSSIA
billion tkm	2009	2008 (6)	2010	2009	2010
Road	*1755.6*	1922.9	334.7 (8)	3718.9	199.0
Rail	389.9	2525.4 (7)	20.4	2523.9	2011.0
Inland waterways	147.4	456.4		1803.3	54.0
Oil pipeline	*120.6*	919.6		202.2 (9)	2382.0
Sea (domestic / intra-EU-27)	*1414.8*	303.5	179.7	3952.4	101.0

Notes: (6) USA: data for road is from 2007.
(7) USA: Class I rail.
(8) Japan: 2009.
(9) China: oil and gas pipelines.

Figure 1.1.1: European statistics on the use of different modes by passenger and freight transport.*Source: European Commission (2012). EU transport in figures. Statistical Pocket Book.*

Furthermore, traffic congestion causes a significant loss of time for private journeys and an inability to comply with timetables for public transport. This waste of time for users is significant to the economy of cities, as demonstrated clearly by various studies. For instance, in 2012 the UK estimated that traffic congestion cost Britain more than £4,3bn a year. (Figure 1.1.2).

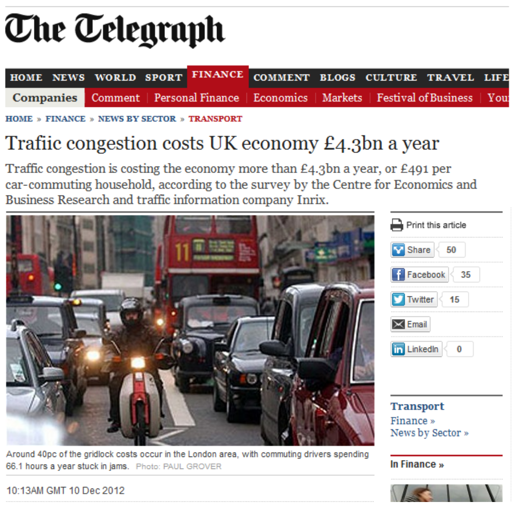
In the face of all of these facts, it has become imperative to develop new strategies and policies for sustainable urban transport. Equally important, continued growth in demand is expected for some time; so it is therefore essential to invest not only in the improvement of infrastructures, but in their planning and management as well.

The relevance and purpose of transport planning is to address the specific needs of a mobile population. Given the complexity of current transport systems, certain analytical tools have proven to be essential. These tools allow planners to inform themselves about the system and make predictions. Thus, mathematical models, statistics and algorithms have figured prominently in identifying solutions and enriching the process of planning and making decisions.

Traditionally, transport planning processhas been divided into the following four phases (Chicago Area Transportation Study, 1960):

1. Data collection.

2. Analysis and Setting of the Models (*Four-Step Model*)


The Telegraph
HOME NEWS WORLD SPORT FINANCE COMMENT BLOGS CULTURE TRAVEL LIFE
Companies | Comment | Personal Finance | Economics | Markets | Festival of Business | You
HOME » FINANCE » NEWS BY SECTOR » TRANSPORT
Print this article
Share 50
Facebook 35
Twitter 15
Email
LinkedIn 0
Transport
Finance »
News by Sector »
In Finance »

Trafiic congestion costs UK economy £4.3bn a year

Traffic congestion is costing the economy more than £4.3bn a year, or £491 per car-commuting household, according to the survey by the Centre for Economics and Business Research and traffic information company Inrix.

Around 40pc of the gridlock costs occur in the London area, with commuting drivers spending 66.1 hours a year stuck in jams. Photo: PAUL GROVER

10:13AM GMT 10 Dec 2012

Figure 1.1.2: News about traffic congestion costs in the UK. *Source: www.telegraph.co.uk*

(a) Trip Generation

(b) Trip Distribution

(c) Mode Choice

(d) Route Assignment

3. Demand Forecasts

4. Future Network Evaluation

Travel cost and time are critical factors to consider and key components of all trafficmodels,which are used for transportation planning. "Travel time and cost measures determined using static assignment procedures use variables that are time invariant. The procedures are inadequate as explanations of influences on travel choices and as measures used to evaluate impacts when deciding how to develop policies for managing transportation systems, how to fund transportation system improvements, and how to measure environmental impacts related to system-wide travel." (Chiu et al. (2010))

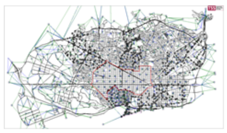
	Travel Demand Models	Mesoscopic Simulation Models	Microscopic Simulation Models
Geographic Coverage	Regional Network / Metropolitan Area 	Regional Network / Metropolitan Area	Small to medium size subarea networks
Demand	Static ODs	Dynamic ODs	Dynamic ODs
Traffic Control	No signal settings.	Detailed signal setting & phasing schemes.	Detailed signal setting & phasing schemes.
Analysis	User equilibrium assignment link costs defined in terms of volume-delay functions.	Dynamic user equilibrium based on simulation based network loading.	Behavioral modeling based on car-following, lane changing and route choice of individual vehicles.
Advantages	Available from local MPO; can analyze mode shift. Low calibration effort.	Can analyze regional dynamic diversion. Brings the dynamic dimension into planning analysis. Moderate calibration effort.	Suitable for detailed dynamic analysis of operational strategies such as ramp metering, traffic signal coordination and ITS.
Limitations	Not sensitive to operational strategies; not capable of analyzing regional dynamic diversion.	Not yet capable of analyzing mode shift.	Data availability for proper calibration.

Figure 1.1.3: Models used in integrated transport analysis.

In order to perform detailed dynamic analysis of operational strategies, time must be explicitly considered in the corresponding models. Traditionally, an effective solution has involved the use of microscopic simulation models that capture all the behaviours of each single vehicles in the network. (Barcelo et al. (2007), adapted from a presentation by Vassili Alexiadis at the Transportation Research Board, Modelling Workshop, 2006). This approach is suitable for small- to medium-sized subarea networks. However, there also exists a clear need for working at an operational level with regional networks or metropolitan areas. So, the main issues of flow dynamics, such as the formation and dissipation of queues, must be incorporatedwithout detailing the interactions of each single vehicle in the network. This is because there is a need to analyze regional dynamic diversion with moderate calibration effort. This is the starting point of the mesoscopic traffic simulation to be used in the network loading component of the Dynamic Traffic Assignment (DTA) models.

The primary limitation of mesoscopic simulation models is that they are incapable of analyzing mode shifts. This thesis focuses on dynamic traffic assignment (DTA) models that are based on mesoscopic simulations using different vehicle classes. This will allow multimodal simulations which will improve on the previously stated limitations of the mesoscopic models.

Dynamic network models provide temporally coherent meansfor representing the interactions between travel choices and traffic flows, as well as the time and cost of travel. DTA models describe time-varying networks and demand interactions using a behaviorally sound approach. The DTA model analysis can be used to evaluate many more measures related to individual travel cost and travel time, as well as system-wide network measures for regional planning. So, DTA models are better suited for travel forecasting models and transportation planning studies.

DTA has been extensively studied for decades, but even more so in the last twenty years since Intelligent Transport Systems (ITS) have emerged. Among other applications, ITS provides pre-trip and en-route travel information that allows drivers to make informed decisions. It also reduces delays and provides more reliable travel times. A necessary requirement of such a system is its ability to predict future traffic in order to proactively respond to drivers' concerns and avoid potentially undesirable network conditions. DTA systems simulate and forecast network conditions under various demands and influences such as accidents, weather, special events, etc.

In an operational context, the objective of DTA models is to represent traffic evolution on road networks when conditions change. They seek to describe the assignment of demand on different paths connecting every OD pair in an equilibrium state. This research takes a behavioural approach by using a time-dependent generalization of Wardrop's First Principle, the Dynamic User Equilibrium (DUE). This hypothesis is based on the idea that, for each OD pair at each instant of time, current travel times experienced by vehicles that depart in the same time interval are equal and minimal (Ran and Boyce (1996)).

DTA systems model complex and dynamic interactions between transportation supply and demand. To account for the effects of supply and demand variations of the assignment, DTA models are classically composed of two submodels:

- The dynamic network loading component which captures flow dynamics and their temporal variations according to the dynamic fluctuation of the demand.
- An assignment model which governs the behavioral rule for traffic assignment. It uses the previously calculated travel costs to determine flow assignments among all possible paths for each OD pair.

As stated before, DTA models have many applications. Ranging from offline transport planning, design and policy evaluation, to online ITS applications such as real-time traffic control, travel information services, time-varying tolling, etc. While offline applications require only that DTA models give an accurate evaluation or prediction, online applications require DTA models to obtain a reasonably accurate solution within a short period of time. This is because of the need to manage sudden changes in traffic conditions and to provide real-traffic information to drivers. So, for real-time applications, DTA models have to strike a balance between the speed of the solution and accuracy in modeling.

Over 95% of the total computational time of a DTA procedure comes from the dynamic network loading component (Carey and Ge (2012)). Thus, it is important to include an efficient process of dynamic network loading with the shortest possible computational times, even in the case of medium-sized or large traffic networks.

1.2 Motivation and Thesis Objectives

The main objective of the research presented in this thesis is to develop a DTA model based on the dynamic extension of the Wardrop Principle referred to as Dynamic User Equilibrium (DUE). For this purpose, we solve a variational inequalities formulation by employing a preventive approach based on an iterative solution algorithm, which is a modification of the well-known Method of Successive Averages (MSA). The developed approach explicitly considers the time and variable traffic demand on each path of the network within the flow propagation and assignment processes. In particular, this thesis develops a DTA model that uses a mesoscopic traffic simulator to reproduce complex traffic flow dynamics.

One of the main motivations for addressing this objective is the need to develop a decision support tool to assist the decision-maker in strategic and operational decisions, particularly in an urban context. Thus, in order to ensure that the developed DTA achieves this functional objective, it is fundamental that the dynamic network loading component is able to reproduce traffic light controls, lane changes, special lanes and different vehicles classes. Therefore, a new multiclass, multilane mesoscopic simulation model is developed with these characteristics in mind, since available existing simulation-based dynamic network loading models were deemed unsuitable for integration into a DTA intended for this use.

In addition, another motivation exists for the research proposed in this thesis. For some time now, our research group has been addressing other traffic problems related to strategic and operational transport planning. During the resolution process of these traffic problems, we realized the high importance of having our own DTA model. In this way, we could easily integrate the DTA results into the resolution approaches of other traffic problems. For instance, we are currently dealing with the dynamic OD matrices estimation problem. One of our proposed solutions includes an iterative approach that performs a complete DTA under the DUE hypothesis at each iteration of the global process. Thus, our DTA model must be capable of becoming an interactive component in a more complex model by exchanging information with the other components. This ability is usually unavailable in most commercial software; and in the few that do allow the required coupled working node, the capabilities are limited.

Thus, we need a flexible and configurable DTA that is more adaptable than existing commercial DTA tools.

Aside from the primary objective, we are interested in the following research questions:

- The original MSA indiscriminately diverts traffic from the paths used in the last iteration to the current optimum paths. The process extracts the same amount of flow from each used path, regardless of whether the path is the worst or only slightly worse than the optimal. Does this diversion process affect the achievement of DUE? And, if so, in which way?
- From a transportation modeling standpoint, is it realistic to accept unlimited growth in the number of OD paths? And, if not, how does an upper bound in the number of paths for each OD pair for each departure time interval affect the quality of the reassignment process and its convergence?
- In the MSA approach, how does the initial number of paths through each OD pair for each interval influence convergence towards dynamic user equilibrium?
- The convergence of the MSA approach depends on the values assigned to the structural parameters of the algorithm (MSA parameter). What is the influence of the different alternatives on convergence to dynamic user equilibrium?

The previous research questions all concentrate on the reassignment flow process. Apart from these contributions of our work, we complete this thesis by considering the following research questions about the dynamic network loading component:

- The aim of this component is to describe traffic flow. In the literature, there are basically two approaches for solving it: models that are based on macroscopic traffic theory and models that try to simplify the performance of each single vehicle, such as the car following models. Is there an approach between these two proposals that is most suitable for representing these traffic flow dynamics?

- Can this approach adequately reproduce the phenomena observed in macroscopic traffic flow behavior? (For example, shock waves and the traffic fundamental diagram?) In addition, can the model that represents the node behaviour accurately reproduce delays and queue spill-backs?

- Which is the most appropriate characterization to describe these temporal dynamics? Time-based or event simulation?

- How important is the model representation of the network topology? For instance, the experiment shows that roundabouts and short-length links could affect the results of the urban traffic simulation.

1.3 Thesis Contributions

In order to accomplish the objectives listed in the previous section, we first needed to investigate in detail the relevant simulation-based DTA models present in the literature. Therefore, the first contribution of this thesis is a complete literature review of the three types of DTA models based on simulation. They are distinguished according to the level of detail with which they represent the studied system, from low to high fidelity: macroscopic, mesoscopic and microscopic simulation models. For each examined DTA model, the review analyzed in-depth not only the approach used to reassign the flow, but also the dynamic network loading component. To the best of our knowledge, no state of the art existed in the literature regarding simulation-based DTA models with any high level of detail.

The second and main contribution of this dissertation is a mesoscopic simulation-based DTA method based on the DUE behaviour approach. The proposed DTA scheme iterates between the two main components until the convergence criterion is satisfied. These two components are referred to as dynamic network loading and path flow reassignment. Additionally, this scheme includes: a time-dependent shortest path component, in order to add new paths throughout the procedure when it becomes necessary; and a K-static shortest path algorithm that determines an initial path set to be used in the first iteration of the process.

DTA models based on simulation use a traffic simulator to reproduce complex traffic flow dynamics (dynamic network loading component). As a result, the third contribution of this research is a multiclass, multilane mesoscopic traffic simulation model well suited for embedding into a DTA scheme. This mesoscopic simulator considers a continuous-time link-based approach with a complete demand discretization. It is the first proposal to focus on the problem from this perspective. Considering disaggregated treatment of each individual vehicle allows the use of different vehicle classes in the problem. Moreover, in order to reproduce the transversal movements described by vehicles changing lanes, which can considerably augment link congestion, the proposed model allows longitudinal discretization of links in lanes.

Finally, this dissertation also proposes a modification of the known Method of Successive Averages (MSA) for the flow reassignment process. This extension takes into account the cost of alternative paths during the diversion process, unlike the standard MSA, which does it indiscriminately regardless of whether the path is the worst or only slightly worse than the optimal. Thus, the proposed method uses a diversion factor based on actual travel times in order to complete a more realistic flow reassignment from among the alternative paths. Moreover, the method combines this with the well-known limitation of the maximum number of available paths for each OD pair for each departure time interval, in order to reduce the computational storage needed in the original MSA.

1.4 Thesis Outline

This thesis is organized according to the scheme presented in Figure 1.4.1.

In the following, we discuss the different components of the scheme in more detail.

Dynamic Traffic Assignment Problem

Before we can start with the development of the DTA model, we need to investigate this problem in more detail. Based on the existing literature, in Chapter 2 we will describe the problem, starting with apresentation of both dynamic user

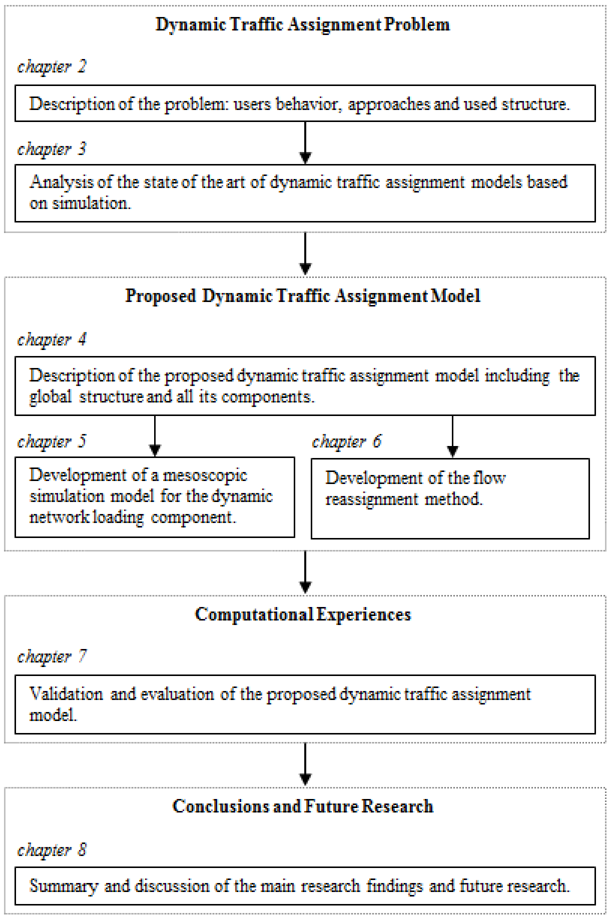

Dynamic Traffic Assignment Problem
chapter 2
Description of the problem: users behavior, approaches and used structure.
chapter 3
Analysis of the state of the art of dynamic traffic assignment models based on simulation.
Proposed Dynamic Traffic Assignment Model
chapter 4
Description of the proposed dynamic traffic assignment model including the global structure and all its components.
chapter 5
Development of a mesoscopic simulation model for the dynamic network loading component.
chapter 6
Development of the flow reassignment method.
Computational Experiences
chapter 7
Validation and evaluation of the proposed dynamic traffic assignment model.
Conclusions and Future Research
chapter 8
Summary and discussion of the main research findings and future research.

Figure 1.4.1: Schematic presentation of the thesis outline.

behaviors: dynamic system optimum and dynamic user equilibrium. Consecutively, we will show how a DTA problem can be solved through four different approaches. We will furthermore illustrate the DTA structure that will follow our development, which has two main components: dynamic network loading and flow reassignment.

Having established the DTA structure, we will analyze relevant DTA models in Chapter 3. Here, we will concentrate on the models that use simulation for the dynamic network loading component, as well as demonstrate that considerable differences exist among the models that use microscopic, mesoscopic or macroscopic simulations for this component.

Proposed Dynamic Traffic Assignment Model

The previous in-depth discussion of the simulation-based DTA models will assist us in developing the proposed algorithm that solves the DTA problem. Chapter 4 will provide the fundamentals of the new proposed DTA model based on DUE hypothesis. In addition, it will explain all the components of the proposed algorithm in detail, paying special attention to the shortest path algorithms: k-static shortest path and time-dependent shortest path. In Chapters 5 and 6, we will develop the main components of the proposed DTA model, the dynamic network loading, and the flow reassignment, each one independently.

In Chapter 5 we will present our multilane multiclass mesoscopic simulation model, which will be used to solve the dynamic network loading problem embedded in the DTA. Additionally, as a means of reviewing the dynamic network loading models present in the literature, we will justify the development of a mesoscopic simulation model. We will then concentrate on on how the presented model operates and will conclude with the corresponding computational experiences, whose results support the proposal.

Moreover, we will present the flow reassignment algorithm separately. In Chapter 6, following a discussion of the corresponding state of the art, we will present our modified Method of Successive Averages. We will furthermore demonstrate the good performance of the proposed flow reassignment process by using an external dynamic network loading model to validate it independently of our mesoscopic simulation model.

Computational Experiences

In order to validate the proposed DTA model, and also to evaluate whether it is correct, we will propose some computational experiments. For that purpose, in Chapter 7 we run the developed DTA model on a real urban network. In order to investigate the behaviour of the model, we will take the results from the proposed model and compare them with the results from a non-iterative assignment simulation. In addition, we will study whether the initial number of paths are important at the beginning of the global DTA procedure. We will also study the role played by the MSA parameter in achieving DUE.

Chapter 7 will describe the test network we used and the experiment design, followed by ananalysis of their corresponding results.

Conclusions and Future Research

Finally, Chapter 8 will provide a review of the main achievements in this dissertation thesis.

We will summarize the main research aims and the approach used to achieve these aims. After this, we will discuss the main research findings. We will finish with a discussion of future research that may naturally follow the research described in this thesis.

Chapter 2

Dynamic Traffic Assignment

2.1 Introduction

The Dynamic Traffic Assignment (DTA) problem, which researchers have been studying since the early 80's, is an extension of the traffic assignment problem that describes the evolution of traffic patterns in the transport network over time and in space (Mahmassani (2001)).

Traffic Assignment Traffic assignment determines how demand is loaded into the road network providing a way to calculate traffic intensity on network sections. Demand is usually defined in terms of origin-destination (OD) travels matrices. The underlying modeling assumption is that trips between origins and destinations use the routes that connect them. The characteristics of a traffic assignment process are determined by a hypothesis of how travelers use the routes.

Traffic assignment in equilibrium is the backbone of the entire methodology. References by Patriksson (1994) and Florian and Hearn (1995) continue to provide the most comprehensive overview of the fundamentals of the models and the structure of the main algorithms.

The paradigm modeling hypothesis is based on the concept of user equilibrium, which means that travelers try to minimize their individual travel times; i.e., they tend to use those routes that they perceive as the shortest under prevailing traffic conditions. This modeling hypothesis is formulated in terms of Wardrops's First Principle (1952): " The travel times in all routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route". Each user non-cooperatively seeks to minimize his cost of transportation.

The traffic assignment models based on this principle are known as user equilibrium models, as opposed to models where the objective is to optimize total travel in the system, regardless of individual preferences.

Dynamic Traffic Assignment As we mentioned in Chapter 1, the higher market penetration of the underlying technologies in Intelligent Transport Systems (ITS) has motivated research on DTA problems. This is due to the important applications that dynamic assignment has for:

- Advanced systems for traffic management.
- Advanced systems for traveler information.

These transport systems are designed to manage traffic networks in real time through: measurement, detection, communication, control and information provision (in real time).

The DTA problem can determine time-dependent link flows or time-dependent path flows in a congested road network, satisfying a set of individual objectives and/or an overall system goal.

2.2 Behavioral Hypothesis

According to the behavioral hypothesis of each decision during a trip, the DTA problems can be classified as follows:

- Dynamic System Optimum assignment (DSO)
- Dynamic User Equilibrium assignment (DUE)

2.2.1 Problem Statement

Let a graph $G = (N, A)$, with a finite nodes set N and a finite set of oriented links A. The studied time period T is discretized in a set of time intervals, $T = \{t_1 = (t_0, t_0 + \triangle t), t_2 = (t_0 + \triangle t, t_0 + 2\triangle t), .., t_S = (t_0 + (S-1)\triangle t, t_0 + S\triangle t)\}$, where t_0 is the earliest possible departure time from any origin node, $\triangle t$ is a small time interval during which no changes in traffic conditions or travel costs are perceived, and S is an integer constant such that the intervals from t_0 to $t_0 + S\triangle t$ cover the entire period of interest T.

The rest of the notation and the variable definitions are shown below, followed by an approach to two possible DTA problems, depending on the considered behavior hypothesis (DUE or DSO).

Notation and variables definition

O	Origin nodes, $O \subseteq N$.
D	Destination nodes, $D \subseteq N$.
T	Departure time intervals.
o	Subscript for the origin nodes, $o \in O$.
d	Subscript for the destination nodes, $d \in D$.
t	Subscript for the departure time intervals, $t \in T$.
P_{odt}	Set of feasible paths for each OD pair (o, d) and for each departure time interval t.
p	Subscript for path $p \in P_{odt}$.
q_{odt}	Travel demand from origin o to destination d departing at time interval t.
f_{odpt}	Number of trips from origin o to destination d departing at time interval t using the assigned path p.
f	Flow vector $f = \{f_{odpt}, \forall o, d, t, p \in P_{odt}\}$.
c_{odpt}	Travel time experienced by vehicles departing from origin o to destination d at time interval t using the assigned path p.

$\hat{c}_{odpt}$	Marginal travel time experienced by vehicles departing from origin o to destination d at time interval t using assigned path p.
π_{odt}	Minimum travel time experienced by vehicles departing from origin o to destination d at time interval t using the shortest path.
$\hat{\pi}_{odt}$	Marginal minimum travel time experienced by vehicles departing from origin o to destination d at time interval t using the shortest path.

Marginal travel time refers to the additional cost that one more trip would incur between origin o and destination d in departure time interval t through the assigned path p. To calculate the marginal travel time, not only the additional travel time added for that vehicle must be taken into account, but also the delays that might occur for other vehicles on the same path p.

2.2.2 Dynamic System Optimum (DSO)

When basing the assignment problem on a system optimization behaviour hypothesis, the solution suggests that travelers will aim to minimize the total cost to the system by behaving cooperatively when choosing their paths and departure time intervals. This assumption is based on Wardrop's Second Principle.

The dynamic system optimization conditions can be mathematically represented by system Equation 2.2.1.

$$\begin{cases} \hat{c}_{odpt} - \hat{\pi}_{odt} \geq 0 \\ f_{odpt} \cdot (\hat{c}_{odpt} - \hat{\pi}_{odt}) = 0 \\ f_{odpt} \geq 0 \end{cases} \tag{2.2.1}$$

Equation 2.2.1 states that if the flow in the path is strictly positive, then the marginal travel time experienced on the path must be equal to the minimum marginal travel time for the same (o, d, t).

This formulation implies that we will have complete information about demand and time variation during the period of interest, i.e., we will know a priori q_{odt} $\forall o, d, t$. This information is used by the traffic assignment problem to globally optimize the system by finding a feasible flow pattern in time-dependent shortest paths $f^* = \{f_{odpt}, \forall o, d, t, p \in P_{odt}\}$. In this way, it minimizes the total travel time of the system.

2.2.3 Dynamic User Equilibrium (DUE)

In 1996, Ran and Boyce expressed the condition of DUE as a time-dependent generalization of Wardrop's First Principle: If, for each OD pair at each instant of time, current travel times experienced by vehicles that have departure in the same time interval are equal and minimal, then the dynamic traffic flow through the network is in a DUE state based on travel times Ran and Boyce (1996).

Analytically, the DUE conditions may be formulated by the system Equation 2.2.2.

$$\begin{cases} c_{odpt} - \pi_{odt} \geq 0 \\ f_{odpt} \cdot (c_{odpt} - \pi_{odt}) = 0 \\ f_{odpt} \geq 0 \end{cases} \tag{2.2.2}$$

Under these conditions, it is implicit that a traveler is rational and chooses the path that minimizes its travel cost. Equivalently, the DUE traffic assignment tries to find feasible flow patterns on a time-dependent shortest path $f^* = \left\{ \sum_{p \in P_{odt}} f_{odpt} = q_{odt}, \forall o, d, t \right\}$. This satisfies Equation 2.2.2, flow propagation constraints, path flow conservation, and non-negativity constraints, as defined in the DTA problem proposed by Peeta and Mahmassani (1995).

All trips on the network are balanced in terms of current costs experienced on the paths, so it is necessary to determine the function relating flow and cost in a DUE assignment model.

To determine these functions, a dynamic traffic model based on simulation is often used to evaluate the cost of the paths experienced by a certain flow vector f. It should be noted that the choice of a dynamic traffic model is not restricted to a particular model; any dynamic traffic model capable of capturing the complex dynamics of traffic flow can be included in the proposed solution algorithm for DUE. This is particularly true of models that capture the dynamics which generate and propagate congestion as well as those which form and dissipate queues.

Before proceeding, it is worth noting here that DTA processes do not always agree with the dynamic version of the Wardrop Principle, Friesz et al. (1993), Ran and Boyce (1996), and therefore optimality cannot be guaranteed.

In these cases, the route selection mechanism attempts to optimize route selection using current information. One frequent variant simulates the uncertainties of the available information by using choice functions, as prescribed by the discrete choice theory. This may also include en-route changes.

These approaches are considered DTA procedures, but they do not achieve DUE. If the assumptions about how travelers choose their routes are consistent with the dynamic principle of the formulated user equilibrium, then a DTA can become a DUE assignment. This topic will be addressed in Section 6.1.

2.3 DTA Approaches

Despite the mathematical difficulty of treating DTA problems, this is not an obstacle to the various real-world solutions available. Different approaches can address different functional needs with varying degrees of strength. The consensus is that a DTA approach must adequately represent real traffic conditions in order to achieve the objectives.

This section reviews the literature about the DTA problem by classifying the various approaches into four broad methodological groups:

- Mathematical programming
- Optimal control
- Variational inequalities
- Simulation

The first three groups are typically considered analytical approaches. In these cases, DTA is formulated as a mathematical problem which is solved directly with known mathematical optimization techniques.

In DTA context, mathematical programming and optimal control theory have significant limitations, while variational inequalities offer some advantages. Because of this, analytical DTA models have migrated toward a variational inequalities approach in recent years. An extended overview of mathematical programming, optimal control and variational inequalities can be found in Peeta and Ziliaskopoulos (2001).

The advantages of the analytical approach are: an ability to apply existing mathematical solution algorithms to the DTA problem, and the ability to determine solution properties such as existence and uniqueness. However, a theoretical guarantee of properties such as existence and uniqueness imposes restrictions on the mapping function (mapping route flows for travel times), and this prevents a realistic representation of traffic propagation. Another drawback is the limited applicability of analytical models. The successful use of analytical models is usually limited to small hypothesized networks, as these models use solution procedures that do not take advantage of the specific characteristics of transportation problems.

Simulation-based DTA models focus on enabling practical deployment for real networks and are designed to handle traffic problems in real-life networks. Solutions existence and uniqueness propertiesare of secondary importance. They have the advantage of being applicable to real-life networks and of being able to adequately capture traffic dynamics and driver behavior. However, one drawback is that theoretical insights cannot be analytically derived, since traffic propagation is modeled through simulation. Another is that properties like existence and uniqueness are not guaranteed in the solution.

Below, we present the existing literature about the different formulations of the DTA problem, classified according to their approach to the problem.

2.3.1 Mathematical Programming Formulation

DTA models based on mathematical programming formulate the problem in a discretized time-setting.

Merchant and Nemhauser were the first to try formulating DTA as a mathematical programming problem (Merchant and Nemhauser (1978a) and Merchant and Nemhauser (1978b)). The formulation was limited to: the system optimum behavior case; a fixed demand, single destination node; and a single commodity. The model used an exit link function in order to propagate traffic and a static link performance function to represent travel cost as a function of link volume. Thus, it resulted in a non-convex non-linear formulation based on flows and with discrete times. This model generalized the static assignment problem based on the system optimumization behavior.

In 1987, Carey reformulated the Merchant and Nemhauser model into a convex non-linear problem by manipulating the exit link functions (Carey (1987)). This new formulation offered more mathematical and algorithmic advantages than the original formulation. The basic part of the formulation was the same as the original,but the exit link functions were used to bound the exit flow through the link instead of using the exit flow itself, as in the Merchant and Nemhauser model. This was done in order to achieve a convex formulation.

In his reformulation, Carey also introduced extensions of the basic model to handle multiple destinations and commodities. However, these extended formulations remained problematic because of non-convexity arising from a first-in first-out (FIFO) requirement. These difficulties appeared to be inherent in all mathematical programming formulations of the DTA problem, both for dynamic user equilibrium and for system optimum cases.

Multiple destination nodes require models that explicitly satisfy FIFO requirements; that is, they are essential for realistic traffic simulation. The FIFO rule has two dimensions: physical and algorithmic. Since in current traffic networks, vehicles can occasionally violate FIFO, it can only approximate traffic propagation. The problematic FIFO violation is induced by its own algorithm, in which some vehicle physically jumps over another to reduce system cost. This is also inconsistent with realistic traffic behaviour.

In 1992, Carey said that the FIFO requirement is easily satisfied in formulations that take into account single node destinations as well as for certain special network structures (Carey (1992)). However, in a general network, this requirement would introduce additional constraints that yield to a set of non-convex constraints, which would destroy many of the nice properties of the formulation and increase the computational requirements of any solution algorithm.

There is another phenomenon related to the mathematical programming models formulating DTA problem under system optimum behavior.That is the "holding-back" of vehicles in links. This arises when link exit constraints are satisfied as strict inequalities. Basically, this phenomenon occurs when vehicles are artificially delayed on a link for an unreasonable period of time. Such a solution is probably not acceptable socially nor is it operationally realistic. From a modeling standpoint, this phenomenon can be avoided by including explicit constraints that preclude it. In 2000, Carey and Subrahmanian illustrated some of these constraints, as well as others, in order to solve the previously mentioned problem related to the FIFO rule (Carey and Subrahmanian (2000)).

In 1991, Janson represented one of the first attempts to model the DTA problem under the DUE behavior hypothesis. He formulated it as a mathematical program (Janson (1991)). One feature of his approach was that it sought equilibrium in terms of experienced path travel times rather than the instantaneous travel times assumed in prior works. In order to ensure the temporal continuity of OD flows, this formulation proposed non-linear, mixed integer constraints, although these constraints can be violated during the solution process.,It was an extension of the incremental assignment heuristic for static formulations. The properties of this procedure are not sufficiently well-defined, so traffic behaviour may be unrealistic.

In 2000, Ziliaskopoulos formulated a linear program for a single destination case of the DTA problem with system optimum behavior (Ziliaskopoulos (2000)). This formulation was based on the Cell Transmission Model (CTM) proposed by Daganzo for the traffic propagation problem (Daganzo (1994)). Although the Ziliaskopoulos model could not operate in real-world applications, it provided some insights into the properties of the DTA problem.

In general, mathematical programming DTA formulations tend to have difficulties related to:

- The use of link performance and/or exit link functions.
- The FIFO rule problems.
- The holding back traffic phenomenon.
- Inefficient solutions for real-time deployment in large-scale traffic networks.

2.3.2 Optimal Control Formulations

In optimal control theory DTA formulations, the OD travel rates are assumed to be known continuous time functions, and the link flows also are continuous functions of time. The constraints are analogous to those for the mathematical programming, but they are defined in a continuous-time scenario. This results in a continuous time optimal control formulation rather than a discrete time mathematical program.

In 1989, Friesz discussed this kind of formulation for two behavior cases: system optimum and dynamic user equilibrium. But he took into account only the single destination case (Friesz et al. (1989)). The models assumed that the adjustment from one system state to another might occur concurrently as the system conditions changed, that is, as routing decisions were made according to current network conditions of the network. However, the decisions could also be continuously modified as conditions changed.

Friesz also proposed a time-dependent generalization of Beckmann's equivalent optimization problem (Beckmann et al. (1956)) for the traffic assignment problem, and he based it on static user equilibrium. These models used link exit functions to propagate traffic flow, and link performance functions to determine travel costs. Moreover, this formulation treated the link inflow as a control variable and the link outflow as a function. This was problematic, since the non-linearity of the exit functions made it difficult to generalize Wardrop's First Principle to the time-dependent case for a network with multiple origins and destinations.

Wie (1989) extended the user equilibrium model to include elastic time-dependent travel demand,which led to implicit consideration of departure time choices.Wie also enumerated several limitations of this approach.

Ran and Shimazaki (1989b) used the optimal control approach in order to develop a system optimum optimization model based on links for an urban transport network with multiple origins and destinations . They also defined the outflow links as linear functions, while the link performance functions were defined with quadratic functions. They found that the model reproduced congestion in an unrealistic way. In addition, they did not consider the FIFO issue that arose from having multiple destinations.

The same year, Ran and Shimazaki presented an optimal control theory based on an instantaneous user equilibrium DTA model (Ran and Shimazaki (1989a)). In this case, the exit link flows were treated as a set of control variables. No efficient algorithms were available to solve these formulations.

Ran used the optimal control approach to obtain a convex model for the DTA problem based on instantaneous user equilibrium behavior, doing so by defining inflows and outflows as control variables. He acknowledged that the usual cost functions failed to take into account dynamic queues and congestion costs, so he proposed splitting link travel costs into two components: queuing and moving. However, these functions are assumed to be non-negative, increasing, and differentiable.Therefore, they may not represent real traffic.

In 1995, Boyce et al. proposed a methodology for solving the discretized version of the problem using the Frank and Wolfe algorithm along with a spatial and temporal extension for representing the network(Boyce et al. (1995)). However, they did not implement the procedure or illustrate it through examples. Also, the use of static link performance functions was a limitation of this proposed model.

In the same year, Codina and Barceló presented a comprehensive and critical review of the most relevant formulations of the DTA problem and algorithms used in both discrete and continuous models. They emphasized the underlying modelling hypothesis and also described a continuous modelling proposal that overcame one of the major reported shortcomings: the FIFO discipline observed in traffic flows. (Codina and Barcelo (1995)).

Thus, despite the attractiveness of an optimal control approach in representing dynamic systems, it suffers from many limitations for DTA formulation:

- The lack of explicit constraints to ensure FIFO and preclude holding vehicles at nodes.
- The inadequate and possibly unrealistic modeling of traffic congestion.
- The lack of a solution procedure for general networks.

Because of the limitations of optimal control theory for DTA and the advantages offered by variational inequalities, analytical DTA models migrated toward the variational inequalities approach, which are discussed in the following Section 2.3.3.

2.3.3 Variational Inequalities Formulations

Variational inequalities generally formulate several classes of problems for DTA. They provide a unified mechanism for addressing equilibrium and other equivalent optimization problems. Also, this approach demonstrates certain mathematical properties such as uniqueness.

In 1979, Smith introduced the variational inequalities approach for the static equilibrium traffic assignment (Smith (1979)). He based his study on the previous work of Dafermos (1971), where the author showed the importance of symmetrical travel cost Jacobian matrices when transforming a traffic assignment model into a non-linear convex optimization problem. Smith proposed a DTA model under a variational inequalities approach based on link flows instead of path flows, as was typical up until that time. In fact, he demonstrated that both approaches were equal,, as well as the existence, the uniqueness and the stability of the solution of the new raisedvariational inequalities.

During the 80s, later independent works by Dafermos and Smith inspired other authors to rapidly develop new theories, applications and resolutions for the variational inequalities approach.

The variational inequalities approach avoids the problems caused by asymmetric link interactions when analytically treating constrained optimization formulations. In that sense, it can handle more realistic traffic scenarios. While the variational inequalities approach is more general and better equipped than other analytical approaches presented for addressing different aspects of DTA problems, the previously discussed broader limitations associated with analytical models remain.

In 1993, Friesz et al. formulated the time-continuous variational inequalities model in order to solve the problem of departure time/route selection by equilibrating the experienced travel times (Friesz et al. (1993)). The model used link performance functions, penalty functions for early or late arrivals, travel demand, desired arrival times, and all the possible paths among origins and destinations. The path cost was a combination of travel cost (determined by the link performance function) and the penalty for early or late arrivals (due to traveling along the path). This formulation reflected traveler behaviour more realistically, but it had not solved any of the issues. For example, there was no proof of solution existence or uniqueness. Also, the characteristics of the formulation required it to solve a complex system of simultaneous integral equations. However, no efficient algorithm existed for this resolution.

In 1995, Wie et al. introduced discrete variational inequalities formulation for the simultaneous route-departure equilibrium problem. They also proposed a heuristic algorithm to approximate it (Wie et al. (1995)). They showed solution existence under certain regularity conditions and used outflow functions instead of the exit time functions proposed by Friesz et al. (1993). The use of outflow functions caused the usual problems of unrealistic traffic flow. Furthermore, the formulation was based on paths, and since complete enumeration of the paths is computationally costly, an efficient method for identifying a subset of relevant paths was needed.

In order to avoid the problems of variational inequalities based on paths, Ran and Boyce proposed in 1996 a discrete formulation based on links with fixed departure times (Ran and Boyce (1996)). They included a queuing delay component to partly alleviate the problems of traffic realism stemming from analytical approaches. However, capacity and over-saturation constraints increased the computational burden, leading to computational feasibility issues for realistic networks.

In 1998, Chen and Hsueh also proposed a DUE model with a variational inequalities formulation based on links (Chen and Hsueh (1998)). They showed that, without loss of generality, link travel time could be represented as a function of link inflow only, and not as a function of inflow, outflow and the number of vehicles on the link. A solution algorithm based on a nested diagonalization procedure was proposed. However, it was still prohibitively expensive for real networks.

In 2002, Lo and Szeto developed a DTA problem formulation, based on cells, that followed a variational inequalities approach under the system optimal behaviour hypothesis (Lo and Szeto (2002)). In order to improve the accuracy of the dynamic traffic modeling, this formulation included a version of the Cell Transmission Model of Daganzo. Also, this formulation satisfied the FIFO condition. The experiment results obtained with this formulation showed that it was able to capture dynamic traffic phenomena such as shock waves or the formation and dissipation of queues.

The variational inequalities approach is more general than the analytical approaches previously described. They provide greater analytical flexibility and are more convenient for dealing with DTA problems. This approach has been used to illustrate the notion of experienced travel time, which was used in DTA under ideal or simultaneous user equilibrium approach.

However, variational inequalities approaches are not a panacea. They are more computationally intensive than optimization models and are computationally intractable in real-time deployment. This issue is further magnified by variational inequalities formulations based on paths, because they require complete path enumeration. In addition, although they can better represent interactions between links, the analytical models continue to have problems representing realistic traffic.

2.3.4 Simulation-Based Models

DTA models based on simulation use a traffic simulator to reproduce the complex traffic flow dynamicsin order to develop meaningful operational strategies for real-time situations.

The terminology "simulation-based models" may be a misnomer. This is because the mathematical abstraction of the problem is a typical analytical formulation. However, the critical constraints that describe traffic flow propagation and spatial-temporal interactions (such as flow conservation and vehicle movement) are addressed through simulation instead of analytical evaluation. This is because current analytical traffic flow representations cannot adequately replicate theoretical traffic relationships. Hence, the term "simulation-based" connotes problem solution methodology instead of problem formulation.

Simulation-based models embedded into DTA resolution are limited by theoretical mathematical properties that cannot be analytically derived because complex traffic interactions are modeled through simulation. Moreover, due to the inherent bad behavior of the DTA problem, notions of convergence and uniqueness in the solution may not be particularly practical. However, due to their better fidelity with the real traffic situations, simulation-based models have gained greater acceptability in the real-world context deployment (Ben-Akiva et al. (1998)).

In addition to the use of a simulator in a descriptive mode in order to determine traffic flow propagation, most existing simulation-based models also use it partially to search for the optimal solution. When the simulator works in predictive-iterative mode, each iteration projects future traffic conditions as a part of the direction-finding mechanism in the search process.

Given the substantial computational burden imposed by the simulator, the choice of "granularity" (macroscopic, mesoscopic or microscopic) is significant in simulation-based models for real-time deployment.

In 1993 and 1995, Mahmassani and Peeta developed DTA models that use a mesoscopic traffic simulator as part of an iterative algorithm in order to solve the problem under the system optimum or dynamic user equilibrium behavior hypothesis. They applied it to a demand with fixed departure times (Mahmassani et al. (1993); Mahmassani and Peeta (1995)). The logic of the adopted simulation combines a microscopic representation of individual drivers with a macroscopic description of some interactions in the traffic flow. This allows an acceptably accurate solution at a fraction of the computational cost of a microscopic representation of traffic maneuvers. The use of a traffic simulator avoids the problems of unrealistic traffic arising from analytical formulations while at the same time it works for general networks.

In 1993, Mahmassani extended the single user class models to the more realistic multiple user class scenarios. Here multiple user classes are assumed, specifically regarding: available information, information about supply strategy, and driver response to the provided information (Mahmassani et al. (1993)). However, as with the analytical functions, these models are not operationally adequate for providing real time optimal path information and/or instructions for network users responding to unpredicted variations in network conditions.

In addition to the conceptual and algorithmic aspects of the models, the real-time and large scale nature of the models are integral to the computational problems. In particular, developing algorithmic procedures must address computational efficiency. As these solutions incorporate a simulator into the iterative research process, they are unfeasible for real-time deployment unless they are modified further.

In 1992, Ghali and Smith proposed a single user class formulation for the deterministic DTA problem under the system optimum hypothesis behavior in which congestion arises exclusively at specified bottlenecks modeled as deterministic queues (Smith and Ghali (1992)). A simulation-based solution is proposed, where the vehicles are routed individually on paths determined by using marginal link travel costs. Although the approach does not ensure system optimality and it has limitations due to assumptions on queuing, it addresses some traffic modeling issues that limit the realism and validity of the analytical formulations.

Ghali and Smith also discussed different levels for computing approximate marginal travel times. However, their methodology involves a brute force simulation of alternative scenarios, which is neither feasible for real-time applications nor efficient in large scale, real-world networks.

In 1994, Smith addressed the DTA problem under dynamic user equilibrium and the system optimal behavior hypothesis by implementing its solution through the CONTRAM simulation model (Leonard et al. (1989)). (See Section 3.2.2).

With the objective of generating a useful real-time model, in 1995 Peeta and Mahmassani developed rolling horizon DTA models, where they explicitly incorporated real time variations in network conditions and fostered computational efficiency in order to allow real time treatment (Peeta and Mahmassani (1995)). The rolling horizon approach provides a practical method for addressing problems which ideally require information about future demand for the entire planning horizon. This is a characteristic of DTA problems. Its advantage is the capacity to use current available information and make short term predictions with some degree of reliability. In this way it can solve a problem in quasi real-time while preserving the effectiveness of the computational procedure in finding good control strategies. From an operational perspective, the information needed is more realistic than the perfect knowledge required by deterministic models. Since it is based on stages, the rolling horizon approach ensures that unpredicted variations in on-line traffic conditions can be adequately represented in subsequent stages. However, if the actual origin-destination trips desired differ significantly from the forecasts, the solution is suboptimal.

In 1998, Ben Akiva et al. proposed DynaMIT (see Section 3.2.5) as a DTA system for estimating and predicting future traffic conditions in real time (Ben-Akiva et al. (1998)). Its interactive supply and demand simulators generate user equilibrium routes under a rolling horizon framework. The demand simulator estimates and predicts origin-destination demand using a Kalman Filtering methodology, and it considers both historical information and the driver response to information. The supply simulator determines flow pattern based on demand. It is a mesoscopic traffic simulator, where vehicles are moved in packets, and the links are divided into segments that include a moving part and a queuing part, in order to model traffic flow.

In 2000, Ziliaskopoulos and Waller introduced an internet-based GIS system that integrated data and models into one framework (Ziliaskopoulos (2000)). The simulation-based DTA model in this system used RouteSim as the traffic simulator. RouteSim is a mesoscopic model based on the cell transmission model (Daganzo (1994)) for traffic propagation. The model was most realistic because it included many realities in the network, such as traffic signals, by using time-dependent cell capacities and saturation flow rates.

Also in 2000, Tong and Wong formulated a DTA model for a network situation with restricted capacity lanes due to congestion (Tong and Wong (2000)). In order to load traffic demand into the network incrementally, they developed a traffic simulator; therefore, the traffic network conditions were dynamically updated. The proposed method was a promising evaluation tool that captured the traffic flow pattern in a dynamic user equilibrium situation. It also identified the network area with congestion in a way that was probably subjected to time-dependent demand.

In 2001, Florian et al. presented DTASQ, which later evolved into the commercial software Dynameq. It was a DTA model that hybridized optimization and mesoscopic traffic simulation under a user equilibrium behavior hypothesis (Florian et al. (2001)) (see Section 3.2.4). In this case, they used an event-based approach to simulate vehicle movementand other temporal components of the simulation. It is important to note here that the performance of this model depends only on the total vehicle flow that crosses the link. Thus, it is not affected by high densities in the links, unlike most of the existing mesoscopic simulators, where the computational effort for each link is proportional to the total time wasted by vehicles when they cross the link.

In 2004, Varia and Dhingra proposed the development of a DTA model under user equilibrium behavior. It was designed for urban congested networks with signalized intersections (Varia and Dhingra (2004)). In this case, they used the simulation-based approach for the case of traffic flow with multiple origins and destinations. They concluded that the formulated model avoided all-or-nothing assignments and provided results close to equilibrium conditions.

Simulation-based DTA models address some of the observed problems in analytical formulations. When trying to reproduce dynamic traffic phenomena, a simulation model incorporates theoretical traffic relationships into the traffic flow model and thereby avoids the limitations of analytical results based on link performances and output functions.

A traffic simulator also captures the complex interactions between vehicles and evaluates the non-linear objective function more satisfactorily than the idealized cost function. In addition, a simulation model can implicitly satisfy the FIFO constraint and circumvent the holding-back phenomenon. All these factors, and the ability to keep track of paths of individual vehicles, represent the advantages of simulation-based approaches to DTA.

Hence, while analytical models have concentrated mainly on deriving theoretical insights, simulation-based models have focused on trying to build practical models for deployment in real networks. Therefore, their key issues address: multiple user classes, information availability, driver response, computational efficiency, robustness, calibration and consistency. Simulation is an option that is especially useful in realistically capturing the complex interactions of different user classes, which are very frequent in actual traffic networks. Given the traffic modeling problems that arise for analytical models in the single-user class, introducing multiple user classes adds substantially to their intractability.

The main limitation of simulation-based models for resolving DTAis the inability to obtain the associated mathematical properties. This is not significant for general traffic networks because the DTA problem for general traffic networks is inherently bad behaviour and intractibility. This restricts the capacity of analytical models and complicates analysis with simplified assumptions that reduce realism.

Another limitation of using a simulator in the DTA context, is in the deployment context. The computational charge associated with the use of a simulator as a part of an iterative mechanism to predict the future can result operationally restrictive. Hence, many recent research about simulation-based DTA models treat to achieve an equilibrium among the accuracy and the computational efficiency.

As mentioned above, many different approaches to DTA exist. Each approach has its own specific characteristics and each approach addresses specific problems or questions that other approaches cannot address very well. Therefore, it seems interesting to continue developing the different approaches simultaneously. Compared to realistic simulations of real-life networks, analytical models are especially useful for generating theoretical insights, analyzing system properties, and exploring new directions to address problems. Based on simulation models are most useful for this purpose. For example, microsimulation-based DTA models should only be used if the application asks for a detailed representation of each individual vehicle.

In the field of simulation-based DTA models, significant progress has been made by many researchers in the last decade. Moreover, our scientific experiences on the four DTA formulations presentedhere have led us to conclude that the simulation-based formulation is the most promising.Consequently, this is the approach we will work on in this thesis.

2.4 DTA Structure

As discussed above in Section 2.3, the proposed approaches for solving the DTA problem may be classified into two broad classes:

- Mathematical formulations that seek analytical solutions.
- Simulation-based formulations that search for approximate heuristic solutions at a reasonable computational cost.

Simulation-based formulations, such as Tong and Wong (2000), Florian et al. (2001) and Varia and Dhingra (2004), decomposed the proposed iterative process into two main components, which Florian et al. (2001) and Cascetta (2001) systematized as follows:

- A method for determining new, time-dependent paths flows by using travel times experienced on these paths in the previous iteration.

- A dynamic network loading method which determines how these path flows propagate along the corresponding paths. It generates time-dependent traffic intensities on network sections, link travel times and path travel times, among others, for the current iteration.

These two components correspond to the "demand" and "supply" sides of the model, respectively. Different DTA systems may choose to adopt different combinations of demand and supply models, but the demand and supply sides are generally not independent. Network loading depends on the demand (e.g., drivers' route choices), yet the route choice is also a function of the supply (travel time).

Figure 2.4.1 shows a solution algorithm structure for a DTA model based on the previously described process. The proposed structure has a general purpose that includes both the dynamic assignment processes that are not equilibrium solutions (for example, cases where stochastic discrete choice models distribute flows among routes), as dynamic user equilibrium solutions. This scheme also includes both analytical models and simulation models for dynamic network loading.

This thesis develops a DTA model based on the presented scheme. As we discussed above, we will choose a simulation model for dynamic network loading, while the path reassignment flow component will be based on algorithms that respect the dynamic user equilibrium hypothesis. Our proposed DTA model is specified in Chapter 4.

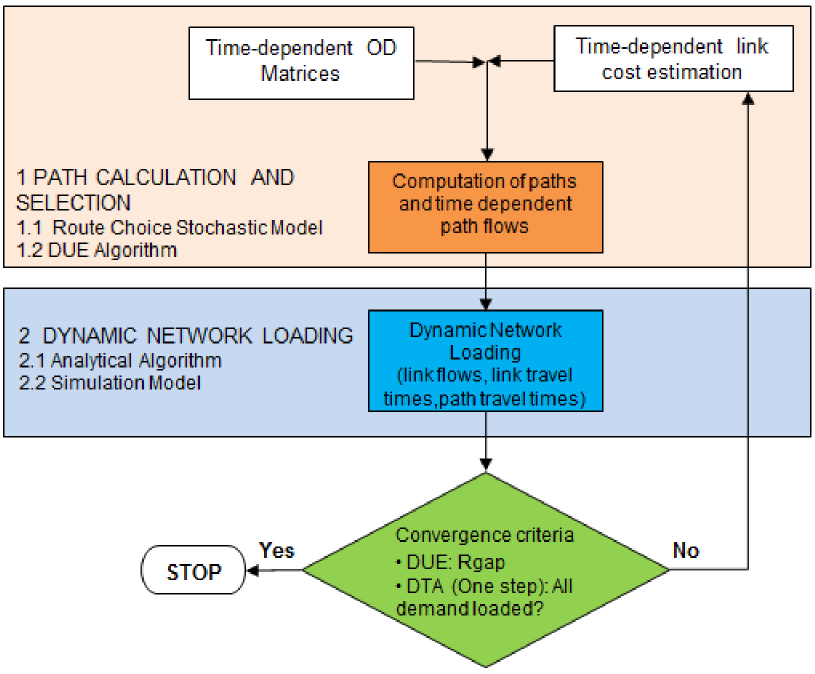

Time-dependent OD Matrices
Time-dependent link cost estimation
1 PATH CALCULATION AND SELECTION
1.1 Route Choice Stochastic Model
1.2 DUE Algorithm
Computation of paths and time dependent path flows
2 DYNAMIC NETWORK LOADING
2.1 Analytical Algorithm
2.2 Simulation Model
Dynamic Network Loading (link flows, link travel times, path travel times)
Convergence criteria
• DUE: Rgap
• DTA (One step): All demand loaded?
Yes
No
STOP

Figure 2.4.1: DTA model structure. *Source: Barceló (2010) (Adapted from Florian et al. (2001)).*

Chapter 3

Simulation-Based DTA Literature Review

The preceding chapter classified the DTA approaches into two groups: analytical and simulation-based. While analytical models are mathematically rigorous, they rely on simplified assumptions to account for traffic dynamics, making them unsuitable for large-scale real-world applications. These applications require models that capture the stochastic characteristics of traffic dynamics in detail by estimating and predicting OD flows, travel times, queues and spill-backs. These capabilities are generally beyond those of existing analytical models and, therefore, simulation is required.

This chapter reviews current simulation-based DTA models.

3.1 Simulation-based DTA Classification

There are three types of simulation-based DTA models, which are distinguished by the level of granularity that represents the studied system. They range from low fidelity to high fidelity as follows:

- Macroscopic simulation models
- Mesoscopic simulations models

- Microscopic simulations models

To begin with, macroscopic models treat traffic in an aggregate manner, such as a uniform or homogeneous flow, without considering each constituent particle (individual vehicle). They approximate flow propagation throughout the network using physical concepts or analytical methods. Microscopic approaches model individual entities, decisions and interactions with a higher degree of detail. Each vehicle maneuvers at a specific simulation time step based on estimations derived from a set of behavioral models, such as car following, lane changing, merging and yielding. Mesoscopic models combine elements from microscopic and macroscopic approaches, representing the activities and interactions of each vehicle with less detail, but still enough to account for the essentials of traffic dynamics.

3.1.1 Macroscopic Simulation-based DTA Models

Macroscopic DTA models apply physical analogies that are usually based on hydrodynamic theory in describing traffic dynamics. The evolution of traffic over time and space is represented by a set of differential equations, the continuum equation, the fundamental diagram and other equations (in the case of second-order models), which are solved by numerical methods that are limited in including appropriate interactions between vehicles and roads. This is why some macroscopic DTA models resort to simulation.

Macroscopic simulation-based DTA models, such as METANET (Messner and Papageorgiou (1990)) or INDY (Bliemer et al. (2004)), aggregately describe both traffic flow propagation and traveler decisions, i.e., traffic is considered to be a steady-state continuum.

The accurate representation of traffic flow in macroscopic models depends on the dynamic network loading mechanism used. Trip-maker decisions are simulated with macroscopic models. These models are usually deterministic, delivering one repeatable average result for a given data set. Computation times no longer depend on the amount of vehicles in the network, and DTA is possible for large networks with millions of vehicles. To be able to simulate the result of route choice decisions, traffic flows in INDY are disaggregated by route so that traffic can actually be assigned to a specific route, as governed by the route choice model. Flows that are disaggregated by route are often referred to as 'multi-commodity' flows. Traffic flows in METANET are not disaggregated by route.

Single commodity flows are routed by splitting proportions at network nodes. The routes followed in a random network loading are generally not consistent with the route flow rates resulting from the route choice model. This is because split proportions, which correspond to a given route flow set, are not easily determined. Existing macroscopic DTA models typically use an equilibrium approach, requiring traffic flows to be disaggregated by route.

3.1.2 Mesoscopic Simulation-based DTA Models

Mesoscopic simulation-based DTA models can move either individual vehicles or packets of vehicles at an aggregate level, while trip-maker decisions are made individually; i.e., individual traveler decisions are represented microscopically in combination with a macroscopic description of traffic flow propagation or dynamic network loading.

Computation times in mesoscopic models are significantly reduced compared to microscopic models, due to the aggregate description of traffic flow. Mesoscopic models can computationally succeed in the analysis of medium-sized networks. These models typically carry out both equilibrium assignments and en-route assignments.

One way to simulate traffic is to group vehicles into packets, and then route these packets through the network (CONTRAM, Leonard et al. (1989)). The packet of vehicles acts as one entity, sharing both the density on the link at the moment of entry and the speed derived from the speed density function defined for each link. The dynamic network loading model in CONTRAM is based on time-dependent queuing theory. Link-end capacity constraints yield queues whenever demand exceeds capacity.

Another mesoscopic paradigm is the queue-server approach used by some models, such as DynaMIT (Ben-Akiva et al. (1998)), FASTLANE (Gawron (1998)) and DYNASMART (Mahmassani (2001)). In this approach, road segments are modeled with both a queuing and a moving part. The vehicles travel through the moving part with the speed calculated by a macroscopic speed-density function. DYNASMART and DYNAMIT use traffic models based on flow that propagate vehicles on links according to a modified Greenshields speed-density relationship. As the vehicle moves through the segment, a queue-server represents congestion by either transferring the vehicles downstream to another segment or forming queues on the current segment. This approach combines the advantages of dynamic disaggregated traffic modeling (individually modeled vehicles) with easy calibration and the use of macroscopic speed-density relationships.

Other DTA models based on mesoscopic simulation are METROPOLIS (de Palma and Marchal (1996)) and MEZZO (Burghout (2004)). The latter was developed at the Royal Institute of Technology as the mesoscopic component of a hybrid mesoscopic-microscopic simulation model and is open-source software.

3.1.3 Microscopic Simulation-based DTA Models

Microscopic simulation-based DTA models describe traffic flow propagation with a higher degree of detail for individual vehicles. They model the movement of each vehicle as well as their interactions with each other and with the infrastructure. Trip-maker decisions such as route choice are represented at the individual level as well.

Traffic flow propagation in microsimulation models combines mathematical car-following and gap acceptance models with heuristics representing driver behavior (e.g., lane-changing behavior and gap acceptance). In addition, traffic control (including signal operation), location and traffic detectors, are also modeled in detail. These models require the detailed calibration of model parameters, which is a very time consuming and computationally costly task.

In the microscopic TRANSIMS model (Nagel et al. (1999)), traffic propagation is based on a cellular automata technique for car-following and lane-changing, enhanced by additional rules for elements such as signals, weaving lanes, unprotected turns, etc. The lanes of all network links are divided into cells of equal size that are either empty or occupied by a single vehicle. Local rules determine the speed and position of each individual vehicle. Since there are fewer model parameters, cellular automata models are easier to calibrate. Maerivoet (2006) also proposes a cellular automaton to propagate traffic for simulation-based DTA.

In the microscopic-based DTA model DTASQ, which later became the commercial software DYNAMEQ, (Florian et al. (2001)), attempts to capture the effects of car-following, gap acceptance and lane changing with a minimum number of parameters. This leads to a reduction in calibration effort. The simulation is a discrete-event procedure and is based on a simplified car-following relationship, which is ultimately somewhat closer to cellular automata models. This also leads to a sharp reduction in computational effort when compared to microscopic discrete time approaches. Mahut, in his thesis, provides a detailed description of this DNL model (Mahut (2000)). Though categorized as a microscopic simulation-based DTA model, DYNAMEQ properties (such as calibration efforts or computation times) have more in common with mesoscopic simulation than with typical microscopic approaches.

Most microscopic simulation models use an en-route approach, but Barceló et al. Barceló and Casas (2004) showed that the microsimulation model Aimsun (Barceló, 1992) could also be used in conjunction with an iterative, equilibrium-like assignment method . Recently, Aimsun included the possibility of using mesoscopic simulation instead of microsimulation to perform the dynamic network loading process into a DTA scheme under the DUE behaviour hypothesis. TRANSIMS and DYNAMEQ deal with equilibrium assignment as well.

Other microscopic, simulation-based DTA models share the conceptual approaches discussed here. In 1993, DRACULA (Liu (2010)) was developed at the University of Leeds; in 1983, INTEGRATION by Van Aerde (Van Aerde et al. (1996)); and finally, the most recent, DynusT by the University of Arizona in 2011.

3.2 Relevant Simulation-based DTA Models

Simulation-based DTA models were developed mainly for evaluating Intelligent Transport Systems as a planning tool that generates and tests scenarios, optimizes control, and forecasts network behaviour at the operational level. Due to rapid technological advancements, simulation-based DTA models have become increasingly popular as a real-world transportation planning tool. The leading advantage of these models is they can test new management strategies without interrupting traffic.

In the following, we briefly summarize relevant DTA models based on simulation and which were presented in the previous classification. For each of these models, we analyze the approach used in the main parts of the DTA procedure: the dynamic network loading component and the flow reassignment or route selection methods.

The DTA models discussed are (in alphabetical order):

- Aimsun

- CONTRAM
- DRACULA
- Dynameq
- DynaMIT
- DYNASMART
- DynusT
- INDY
- INTEGRATION
- METANET
- METROPOLIS
- MEZZO
- VISTA

3.2.1 Aimsun

(Advanced Integrated Microscopic Simulation and Urban Networks)

The traffic simulation software Aimsun was created in 1985 by a research group led by professor Jaume Barceló at the Technical University of Catalonia. Since 1998, the company Transport Simulation Systems has continued to develop and improve the software.

Aimsun answers the need to create a common framework for different analysis approaches. Thus, with the addition of a mesoscopic simulator, Aimsun has become a tool that integrates into a single application three types of transport models: traffic assignment tools, a mesoscopic and a microscopic simulator.

The mesoscopic simulator provides an additional option for professionals who wish to model the dynamic aspects of traffic in large networks. The model works with individual vehicles and uses simplified car-following and lane changing models. When compared with the microscopic simulation, mesoscopic models remove most of the computational load of calibrating during the loading procedure. It models the sections of the network considering two parts: free flow and queueing. For this reason, the mesoscopic approach focuses on the main events and it is able to advance time simulation among different events. This simplifies the computational load and considerably improves process performance.

Aside from the Aimsun's ability to perform a static equilibrium assignment, the software also has the option of performing a DTA based on the computational DTA framework proposed by Florian et al. (2001). It consists of the two above-mentioned components: dynamic network loading and route selection.

Aimsun's Dynamic Experiment Editor offers two options for dynamic network loading: microscopic or mesoscopic. After selecting one of these, the DTA folder offers two different dynamic assignment options:

- Stochastic route choice: based on a stochastic heuristic that uses stochastic route choice models.
- DUE: in this case, Aimsun has different approaches, depending on the simulation selected for the dynamic network loading component. For microscopic simulation, Aimsun uses an iterative heuristic. For mesoscopic simulation, it uses the Method of Successive Averages (MSA).

The stochastic route choice and the DUE parameters are defined inside the DTA folder with great accuracy.

To implement this conceptual approach in a computationally efficient manner, the analytical part of the process (which calculates the routes) needs to be independent of the selected dynamic network loading procedure. The DTA exploits the common network representation for mesoscopic and microscopic approaches. This allows it to calculate in the same way the shortest paths for both approaches: meso or micro. The calculation is based on the time-dependent link cost functions evaluated in terms of the current link costs or the link costs stored in previous iterations. The only difference is the argument values of the link cost functions, which should be provided, respectively, by the microscopic or mesoscopic traffic simulator used in the dynamic network loading phase of the algorithm, depending on the selection made by the user.

As we mentioned above, Aimsun uses a common network representation, object model and database accessible by all models. In addition, both the microscopic

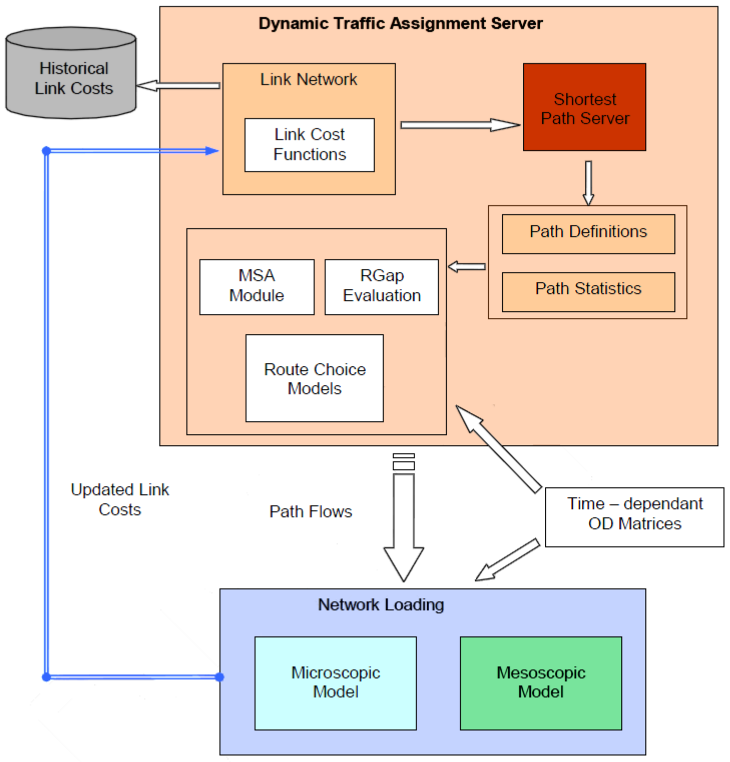


Figure 3.2.1: DTA server in Aimsun. *Source: Barceló et al. (2006).*

and mesoscopic models are based on individual vehicles. This makes it possible to implement a DTA server (Barceló et al. (2006)), whose conceptual structure is depicted in Figure 3.2.1.

The convergence criterion depends on the selected alternative: 1) the demand loading is completed in a one-off DTA based on route choice models, and 2) either when the number of defined iterations is reached or when the Relative Gap function reaches the desired accuracy in the DUE-based DTA.

In the latest version of Aimsun (AIMSUN 7), a hybrid simulator is included. It enables simultaneous microscopic and mesoscopic simulation. This means that large areas can be modeled mesoscopically and, afterwards, the user can

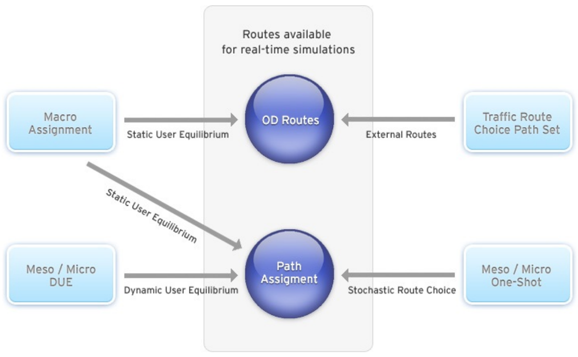


Figure 3.2.2: Rich DTA framework in Aimsun v7. *Source: www.aimsun.com*

zoom in on pockets that require a finer level of detail. The hybrid simulator combines the computational efficiency of an event-based mesoscopic model with the precise representation of traffic dynamics provided by a more detailed, time-sliced microsimulator.

This framework provides model flexibility for recurring and non-recurring conditions at the appropriate level of accuracy. Moreover, by reusing previously obtained static or dynamic equilibrium assignment routes in a new simulation, and combining them with discrete route choice (as shown in Figure 3.2.2), Aimsun is able to reproduce the blend of habitual driving and congestion avoidance that occurs in reality.

Dynamic Network Loading

The mesoscopic model in Aimsun works with individual vehicles, but it adopts a discrete-event simulation approach. Specifically, this mesoscopic simulator includes the following types of events:

- Vehicle generation
- Vehicle system entrance
- Vehicle node movement
- Change in traffic light status

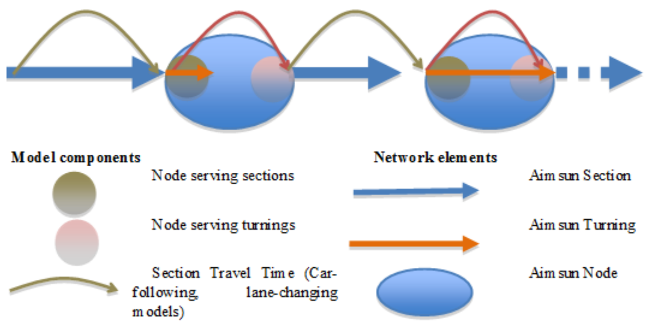


Figure 3.2.3: Modeling vehicle movements in Aimsun mesoscopic. *Source: www.aimsun.com*

- Statistics collection
- Matrix change.

These events model the vehicle movement through sections and lanes by using a simplification of the car-following, lane-changing and gap-acceptance models used in Aimsun's microscopic simulator. On the other hand, nodes are modeled as queue servers.

The movement of vehicles in the Aimsun mesoscopic model depends on a vehicle's location:

- Vehicle movement in a section
- Vehicle movement at nodes:
  - Vehicle movement at turnings
  - Vehicle movement from sections to turnings
  - Vehicle movement from turnings to sections

Figure 3.2.3 illustrates mesoscopic vehicle movements in Aimsun.

In Aimsun mesoscopic, vehicles are assumed to move through sections and turnings. Section capacity is calculated using jam density, multiplied by the length of the section and the number of lanes. Turning capacity is calculated in a similar way, but using the feasible connections in the node instead of the number of lanes.

To calculate section travel time, car-following and lane-changing models are applied. The modeling of vehicle movements inside sections in the Aimsun mesoscopic simulator is based on Mahut (1999, 2000). Aimsun is based on this node model that moves vehicles from one section to the next section of its route. In this model, two actions take place in all nodes: calculating the next vehicle to enter the node and calculating the next vehicle to leave the node.

Before a vehicle enters into a section, the event called "node event from turning" is treated in order to calculate the origin and destination lanes. For this, Aimsun mesoscopic uses two heuristics in order to decide the next lane movement: the status of the next section and a look-ahead model.

The gap-acceptance model is used to model give-way behaviour. The generic rule is the FIFO rule, except when there is a traffic sign, in which case a simplified gap-acceptance model in the Aimsun microsimulator is applied.

Flow Assignment

Three DTA schemes may be employed by Aimsun in a modeling study, depending on the objectives:

- Stochastic route choice
- Stochastic route choice with memory
- DUE

Stochastic route choice The DTA, based on discrete choice theory, used the route choice mechanism in order to try to optimize route selection decisions based on currently available information. The stochastic route choice in Aimsun is based on Barceló and Casas (2004). A path selection process based on a discrete route choice modelselects the route and determines the path flow rates.

Given a finite set of alternative paths, the path selection calculates the probability of each available path and then the driver's decision is modeled by randomly selecting an alternative path according to the probabilities assigned to each alternative. Route choice functions model user behaviour according to the most likely criteria employed by drivers for deciding on what appear to be the most useful alternatives routes in terms of the perceived utility of the user, defined in terms of perceived travel times, route lengths, expected traffic conditions, etc.

The logit, C-logit and proportional route choice functions are the default route choice functions available in Aimsun. It allows additional functions to also be introduced by the user.

Stochastic route choice with memory This is a model of how travelers adjust their current information with conjectures about the expected traffic conditions ahead. It corresponds to how commuters adapt their behaviour through day-to-day learning, depending on fluctuations in traffic patterns, up to the point that no further improvement seems possible.

Aimsun combines dynamic microscopic or mesoscopic simulations with an iterative heuristic procedure that mimics the day-to-day learning process. It attempts to reach DUE, but with no guarantee of convergence (Barceló and Casas (2002), Barceló and Casas (2006)).

Dynamic user equilibrium In this case, the planning level intends to approximate the final state by assuming that Wardrop's generalized principle applies, i.e., by reaching or approaching DUE.

Aimsun uses the Method of Successive Averages (MSA), which redistributes the flows among the available paths in an iterative procedure (see Section 6.2.3). In particular, the MSA implemented in Aimsun is the version of the algorithm proposed by Florian et al. (2001). This is one of the most efficient computational modifications of the MSA for keeping the number of alternative paths bounded in order to account for each OD pair. As we explain later (Section 6.2.3), this variant of the algorithm initializes the process on the basis of an incremental loading scheme that distributes the demand among the available shortest paths; the process is repeated for a predefined number of iterations, after which no new paths are added and the corresponding fraction of the demand is redistributed according to the classical MSA scheme.

3.2.2 CONTRAM

(CONtinuous TRaffic Assignment Model)

CONTRAM, developed by Leonard and Power in the early 80s, was one of the first mesoscopic models. Its main objective is to establish a tool that can reproduce the dynamic aspects of DTA models under an equilibrium approach. CONTRAM has a wide range of modeling tools (that are continuously under development) for representing different situations ranging from congested urban networks to inter-city regions.

As demand for travel increases and congestion occurs over longer time periods and larger network areas, planners and engineers require models that can reflect the complex travel behaviour in these conditions as well as new and innovative ways of controlling traffic. For this reason, CONTRAM is designed to model the varying traffic demand and congestion that occurs during the day and will represent the peaks of congestion as well as off-peak conditions within a single model.

The CONTRAM model falls into the category of DTA models. Its distinctive approach is to combine a form of microscopic simulation of traffic quanta, called "packets", by way of analogy with communication networks, with a macroscopic time-dependent traffic model

By interpreting packets as analogous to individual vehicles that behave beyond optimum route seeking (which leads to equilibrium solutions), it is possible to simulate a response to ITS. CONTRAM is tactical in the sense that it deals with the assignment of a given demand, not with how that demand is created.

In CONTRAM, vehicle packet histories are continuous over time, but they interact only through an underlying time-sliced macroscopic traffic model. This eliminates the need to model all events in a strict sequence. So, the CONTRAM model is intermediate between macroscopic equilibrium and microscopic models. In other words, it is a mesoscopic approach.

CONTRAM demand is divided into a stream of small packets, which are routed independently and assigned sequentially in order of the journey start time. Iteration is needed because each packet can influence other packets that start their journeys earlier as well as later. Knowledge of the network's state develops as if through day to day experience. Figure 3.2.4 illustrates this.

Trip B is assigned computationally after trip A, but arrives at a part of the network's congested center before A. B contributes extra flow and queueing at

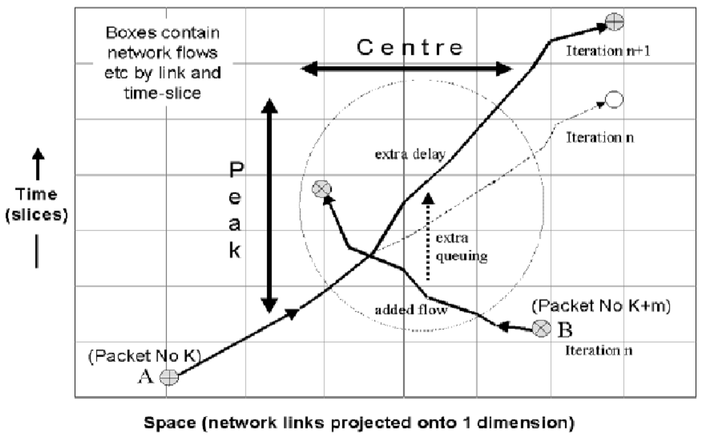


Figure 3.2.4: Schematic representation of packet interaction within the traffic model in CONTRAM. *Source: Taylor (2003).*

a certain iteration (n) by causing additional delay to A in the next iteration ($n+1$). Each packet follows its minimum cost route in each iteration. If the assignment converges, no packet can switch unilaterally to a route of less time or cost. So, in theory, Wardrop's dynamic user equilibrium (DUE) applies.

Dynamic Network Loading

CONTRAM divides traffic into a maximum of 32 user classes in order to represent different vehicle types and journey purposes. Each class has properties which include its generalized cost function given by Equation 3.2.1.

$$C = aD + bT + cV^2D + pT_q, \tag{3.2.1}$$

where:

D is the distance traveled.

T is the total time spent.

V is the average speed.

T_q is the time spent queuing.

a, b, c and d are weight parameters.

In CONTRAM, demand data contains time-dependent flow rates for each origin-destination class and optionally fixed route combination, producing similar time-sliced profiles. If a route is not specified, then the demand is assigned to one or more routes according to minimum cost, as defined by the generalized cost function for that class.

To create the packet stream, CONTRAM generates a stream of more or less equal sized packets for each classified OD movement, according to its demand profile. These packets are then interleaved and assigned in order of their journey start times. The packet size can be calculated automatically. For each classified OD movement, the packets are all about the same size, so changes in flow rate between different time slices are reflected in changes in departure frequency.

The formula for calculating the target packet size of an OD movement is intended to balance the size and number of packets, as shown in Equation 3.2.2.

$$P = k \cdot \frac{\sum_{i=1}^{n} Q_i}{\sum_{i=1}^{n} \sqrt{Q_i}}, \tag{3.2.2}$$

where:

Q_i is the total volume loaded on that OD movement in each time slice i.

n is the number of time slices with data.

k is a scaling factor (supplied by the user).

The structure of the model is based on a network (nodes and links),where the behavior of the group of vehicles at a certain link is determined either by the free-flow speed in that link or by a speed-flow macroscopic relationship. Moreover, links have flow saturation limits. The additional delay experienced at the nodes by the group of vehicles is calculated based on traffic light control plans, delay averages in the give ways, stops, etc. Depending on the performance of the node, the queue can exceed the storage space given by the length of the link and by the number of lanes. At that point, the queue blocks the entrance of the link and the upstream node.

The speed-flow functions used in CONTRAM represent friction in uncongested conditions on motorways and other high-speed links. Continuous, piecewise

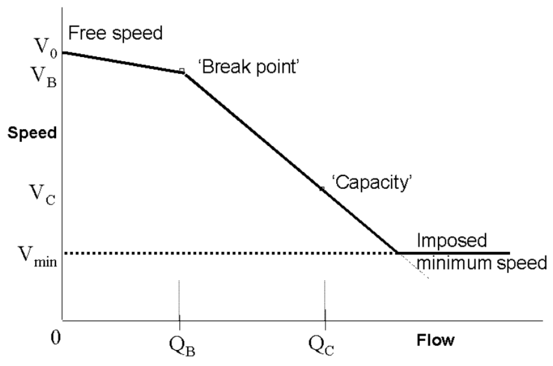


Figure 3.2.5: COBA uncongested speed-flow relationship. *Source: Taylor (2003).*

linear COBA (COst Benefit Analysis) functions are supported (see Figure 3.2.5). These are static functions which modify the cruise speed along a link in each time slice according to the average flow entering the link in that time slice. COBA speed-flow relationships are unsuitable for modeling congestion because they do not cover queuing conditions. Queuing on motorways is modeled in CONTRAM using an explicit bottleneck represented by the "stop line" of a link. This gives the correct total delay, which is a function of the excess of demand over capacity; but it is not entirely realistic where the capacity loss is due to flow breakdown.

COBA functions have the effect of altering the average cruise speed along the running section of a link. Thus, the effective delay is shown in Equation 3.2.3.

$$d_{SF} = D\left(\frac{1}{V_{SF}} - \frac{1}{V_0}\right), \tag{3.2.3}$$

where:

D is the distance traveled.

V_0 is the speed at zero flow.

Queuing delay is accounted separately. A packet-based model needs to calculate the delay to each individual packet. For constant capacity C, the mean time spent in a queue length L is given by the Equation 3.2.4.

$$t_D = \frac{L}{xC}, \tag{3.2.4}$$

where the product xC is the rate of throughput and C can be interpreted as the time-average mean degree of saturation of the stop line, or utilization of capacity. The presence of C in Equation 3.2.4 reflects the fact that capacity is generally unused.

Total delay to a vehicle, possibly accumulated over several time slices, is calculated by applying Equation 3.2.4 iteratively to the queue remaining ahead at the start of each time slice until clearance.

Flow Assignment

CONTRAM finds the minimum cost routes using a time-dependent link-based on the all-or-nothing tree-building method (Dijsktra 1959, Whiting and Hillier 1960). Because network conditions can change significantly even for small shifts in start time, a new route tree has to be built for each packet. Since only the minimum cost route to the actual destination is required, there is no advantage to using more efficient algorithms.

Each packet seeking its minimum cost route has “perfect information”. Packet reassignment consists of first deducting the flow of the packets from the network, and updating the network state; then finding its new route, loading it and updating the network again. When calculating the expected travel time along any link, the packet flow is temporarily added to the link flow to compensate for its finite size. This proposed method of loading results in small changes, provided that the packet sizes are not too big.

One iteration of the model is constituted by running through the entire packet sequence. Iterations are repeated a number of times (as specified by the user), or until the model converges (gap), or until a number of stability criteria prescribed in the data are satisfied (Root Mean Square change in link flows, Average Absolute Difference or Percentage Relative).

In contrast with other DTA models (such as DynaMIT or DYNASMART), CONTRAM has not been developed to be applied in an online context with the aim of forecasting. Moreover, the iterative nature of the proposed assignment process makes the model less suitable for modeling driver behaviour whenchoosing the route during the travel. This is very important when the objective is the treatment of incidents. Although CONTRAM has a special version for incidents, it is limited in the way that the model represents queues, shock-waves and its deterministic assignment.

3.2.3 DRACULA

(Dynamic Route Assignment Combining User Learning and microsimulation)

The dynamic network microsimulation framework DRACULA has been undergoing development at the University of Leeds since 1993. It was created as a tool for investigating the dynamics between demand and supply interactions in road networks. The emphasis is on the integrated microsimulation of individual decisions of the trip makers. This is represented through a microscopic DTA model based on the explicit modeling of individuals' day-to-day route and departure time choices and how their past experience and knowledge of the network influence their future choices.

The DRACULA framework integrates a number of sub-models of traffic flow and driver choices for a given day-to-day driver learning sub-model. The overall structure of the DRACULA model framework is illustrated in Figure 3.2.6.

The sub-models and the dynamic evolution of the demand-supply interactions are as follows:

- A population sub-model which synthesizes the population in the study area and generates all the potential drivers from a traditional origin-destination matrix.
- A demand sub-model which represents the day-to-day variability in total demand. It predicts the level of individual demand for a certain day from a full population of potential drivers.
- A DTA sub-model determines the routes and departure times of the individual drivers based on their past travel experience and their perceived knowledge of the network conditions. The results are individuals' trip plans.
- The costs experienced by drivers for the specific day and for each passing link are then re-entered into their individual "knowledge bases". In

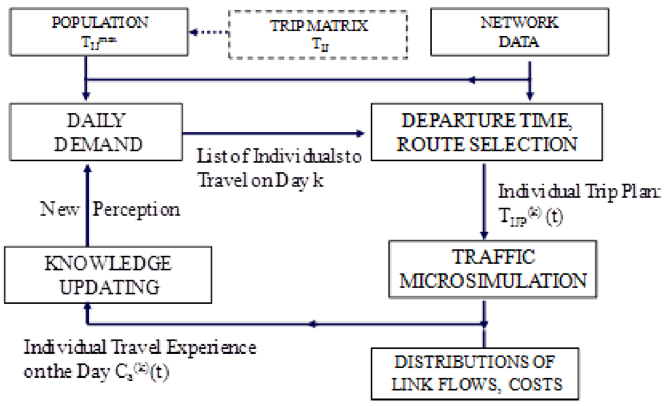

POPULATION
TRIP MATRIX
NETWORK DATA
DAILY DEMAND
DEPARTURE TIME, ROUTE SELECTION
List of Individuals to Travel on Day k
Individual Trip Plan:
New Perception
KNOWLEDGE UPDATING
TRAFFIC MICROSIMULATION
Individual Travel Experience on the Day
DISTRIBUTIONS OF LINK FLOWS, COSTS

Figure 3.2.6: The day-to-day evolution represented in DRACULA.*Source: Liu (2010).*

the traffic microsimulation sub-model, the individual vehicles are then moved through the network following their chosen routes according to car-following and lane-changing rules.

- The learning sub-model updates driver perceptions of network conditions, which in turn affects their decision for the next day.

- A data collection sub-module collects measures of travel time, congestion, incidents, emissions, etc.

The system evolves continuously from one day to the next until a pre-defined number of days or a balanced state between the demand and the supply is reached.

Dynamic Network Loading

The essential property of the DRACULA traffic simulation model is that the vehicles move in real-time and their space-time trajectories are determined by car-following and lane-changing models. Moreover, DRACULA interacts strongly with the demand model.

The car-following model in DRACULA is based on the car-following rules of Gipps (1981) and can represent individual vehicle trajectories in a real way. Their aggregated impact is also measured for saturation flow and discharge profiles at a traffic signal controlled intersection.

The speeds which the Gipps model represents are relatively low and mainly correspond to traffic speeds usually observed on urban streets. For this reason, the traffic simulation model in DRACULA has been extended to represent dynamic traffic in other situations, for example the "un-interrupted" traffic flow dynamics typically seen on long, high-speed motorway links. The new car-following model, described in Wang (2005), aims to capture some of the motorway flow characteristics, namely traffic breakdown, hysteresis, shock wave propagation and close-following behaviour.

The new car-following model was built on the concept that drivers in different traffic conditions behave differently. The traffic conditions considered were:

- Traffic build-up: from free towards congestion.
- Close-following: at high speed and short headway.
- Traffic breakdown: flow and speed reductions, and increasing density.
- Traffic recovery

Behaviorally, the drivers are assumed to be in different states of alertness under different traffic conditions and therefore apply different reaction times and accelerations accordingly. The car-following behaviour is represented using the Gipps model, but with different reaction times and accelerations for different states.

The model is shown to be able to realistically capture key motorway traffic flow characteristics, including speed drop and traffic hysteresis (Wang (2005)).

The lane-changing model in DRACULA is rule-based and stems from models of two very different causes or desires for lane-changing:

- Mandatory lane-changing: the vehicle is in the wrong lane, therefore it has to change lanes. In this case, the move has to be made by a certain position on the link.
- Discretionary lane-changing: the vehicle wants to change lanes for more comfort or some other desire. This movement may or may not need to be carried out, depending on the actual traffic conditions.

Flow Assignment

DRACULA was developed as a tool to test the fundamental properties of DTAs. To this end, various DTA sub-models have been implemented and studied in DRACULA. These models vary by detail and behavioral assumptions. At a more aggregate level, a simple dynamic route choice model based on the aggregated response to overall system performances was also implemented in DRACULA. On the other hand, at its most detailed level is the day-to-day, microscopic model of individual route and departure time choices. It also has an individual-based learning model.

All the route choice models presented in DRACULA are pre-trip DTA; there is no en-route route choice represented in DRACULA.

Simple dynamic route choice model In this model each individual is represented. Their daily route choices are modeled explicitly and they are based on each individual's past experience and perception of the network state. Two route choice models are implemented in DRACULA:

- Bounded rational model: assumes that drivers will use the same route as on the day they last traveled, unless the cost of travel on the minimum cost route is significantly better. The threshold is that a driver will use his habitual route unless

  $$C_{p_1} - C_{p_2} > \max\{\eta \cdot C_{p_1}, \varphi\},$$

  where C_{p_1} and C_{p_2} are the costs of the habitual and the minimum cost routes, respectively, and η and φ are global parameters representing the relative and absolute cost improvement required for a route switch.

- Myopic switch: is a special version of the above defined model in which the threshold is zero, i.e., a driver would always take the least cost route. In this case,

  $$C_{p_1} - C_{p_2} > 0$$

The individual costs C_{p_1} and C_{p_2} were updated from their own past travel experience using the day-to-day individual learning model. At trip $(k-1)$, the perceived cost of the driver on a certain link would be a weighted average of costs incurred in the previous N trips, where N is the maximum number of remembered experiences.

The day-to-day model The simplest model available in DRACULA is calculated from the route choice probabilities of the given costs on alternative routes. This model assumes that route choice proportion is in accordance with the choice model shown in Equation 3.2.5.

$$P_{ijk}^{n+1} = \frac{\left(C_{ijk}^{n}\right)^{-\alpha}}{\sum_{l=1}^{m}\left(C_{ijl}^{n}\right)^{-\alpha}} \tag{3.2.5}$$

where:

P_{ijk}^{n+1} is the proportion of trips from the OD pair ij along route k on the following day $(n+1)$.

C_{ijk}^{n} is the cost along route k from origin i tor destination j on day n.

m is the number of routes used on day n for OD pair ij.

$\alpha > 0$ is a dispersion parameter that represents the degree of heterogeneity in a driver's route selection and it is used to disperse drivers among alternative routes for a given OD pair.

3.2.4 Dynameq

Dynameq (Mahut (2000); Florian et al. (2001)2) was developed by INRO Consultants, who also developed the static traffic assignment (in equilibrium) model EMME/2 (Emme 3 and Emme 4). Dynameq is a DTA model based on the DUE hypothesis.

In an equilibrium DTA model, the objective is to minimize the travel time of each vehicle so that, for each origin-destination pair, the vehicles that depart from this origin at the same time should take approximately the same amount of time to reach their destination. Dynameq accomplishes this through an iterative process, where each iteration executes, one time each, the route selection model and the traffic simulation. The traffic simulation receives flow ratios in the time-dependent paths from the route selection model and simulates the resulting traffic patterns in the network. Then, the simulator returns network travel time information to the route selection model, which induces new path selections in the next iteration of the global procedure. Therefore, the output data from each of the models are the input data for the other. The process continues cyclically until it converges towards a proper DUE state.

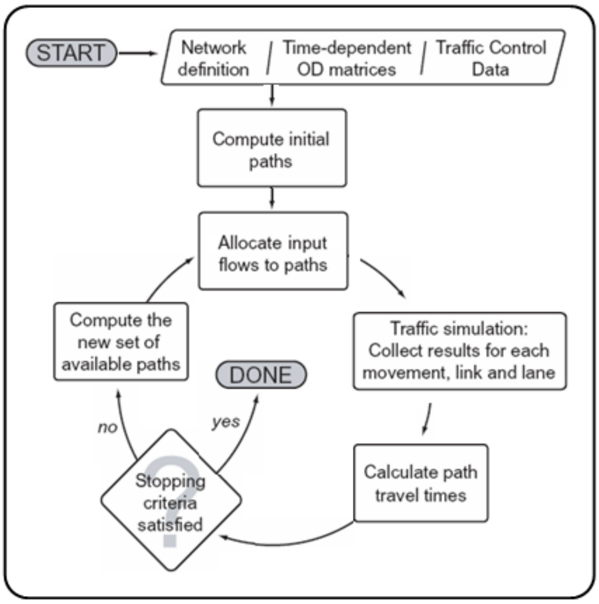

START
Network definition
Time-dependent OD matrices
Traffic Control Data
Compute initial paths
Allocate input flows to paths
Traffic simulation: Collect results for each movement, link and lane
Calculate path travel times
Stopping criteria satisfied
yes
DONE
no
Compute the new set of available paths

Figure 3.2.7: DTA algorithm scheme in Dynameq. *Source: Mahut and Florian (2010)*

The iterations can be thought of as a sequence of consecutive days, where drivers depart the first day with knowledge of the network but without knowledge of the traffic patterns resulting from their route selections. Each day, after experiencing the results of congestion, each driver considers the possibility to choice different route for the next day. After a certain number of days, drivers stop, search for new routes, and restrict their choices to the paths that have been previously selected. The iterative process is shown in Figure 3.2.7.

Dynamic Network Loading

The core of the traffic simulation model is a simplified car following model proposed by Mahut (2000) to be an improvement on the popular Gipps model.

In 2001, Mahut proposed a multi-lane version of the previous model, maintaining the properties of the original. This new model also captures the interactions among vehicles due to lane change maneuvers. The multi-lane model also has a complex set of heuristics for modeling driver decisions about lane selection. These take into account:

- Driver intention to change lane downstream from its current position.
- The lane that should be used in order to perform the next planned turn.
- The traffic conditions in each lane, from the current driver position to the end of the link where the driver is.

The simplified car-following model proposed by Mahut is recursively extended by applying a sequence of vehicles. For example, instead of modeling the relationship between vehicles 1 and 2, and the next relationship between vehicles 2 and 3, the model can express the effect that vehicle 1 has on vehicle 3. This can be extended to any number of vehicles. Conceptually, as the network delays are sourced mainly at the nodes, the role of link dynamics is to correctly propagate the delays caused by vehicles downstream fromthe link as compared to the vehicles at the entrance of the link. Instead of explicitly modeling the position of each vehicle (in order to determine when link congestion begins to affect link entrance time), the output delays propagate directly to the link entrance.

This traffic dynamics model is a time-continuous, space-continuous, discrete-flow model. This is combined with a node model that explicitly represents traffic control systems. This combined system is solved by an algorithm based on discrete events that allow modeling the network completely.

The dynamic network loading model used in Dynameq is an event-based model that is computationally very efficient. This model also respects the basic laws of traffic flows and represents congestion mechanisms that occur in real traffic. The simplification of the car-following model is necessary for an effective, event-based procedure. The computation time is particularly important in DTA, which performs many repetitions of the simulation model when solving the equilibrium assignment. The fact that DTA models tend to be used in large networks, where there are significant route selection processes, makes computation time more critical.

Furthermore, it should be mentioned that another remarkable property of Dynameq is that it is a multiclass model, i.e., multiple classes of vehicles can be specified. Each vehicle class has its own paths set for each assignment interval. And different permitted movements are defined at intersections for each vehicle class and for each different lane of the network sections.

Flow Assignment

In real life, many drivers make their decisions based on first-hand knowledge of typical traffic conditions in the network. These traffic conditions are not used as input for Dynameq DTA. Since traffic conditions are the result of driver route selections, they are the output data of the DTA model, like the selected routes.

The modeled decisions about route selection use an iterative approach. The DTA iterations can be thought of as days. Each iteration (or day), every driver makes a decision on which route to use (from its origin to its destination) at the desired departure time, based on the knowledge of traffic conditions in the previous iteration (or day). This is called the "day-to-day learning process". The information used for traffic conditions is the travel times of the routes between each origin-destination pair for the same period of time that contains the desired departure time. This period of time is called the assignment interval. The percentage of drivers who choose each of the available paths, for a given OD pair is constant for the duration of each assignment interval. These percentages, like the routes, may change from one assignment interval to the next. Travel times are obtained using a traffic simulator that models the traffic conditions that would occur in the network, due to the driver route selections given for each iteration of the model.

The entire structure of the Dynameq DTA model is shown in Figure 3.2.8, which illustrates the interaction between the components of the route selection and the traffic simulation.

Dynameq performs the first iteration in a special way because, at that time, previous traffic conditions are unknown. So, all drivers choose the shortest path assuming that traffic flows are produced under free flow conditions at each link. Then, the simulation component models the movement of all the vehicles in the network through these routes. At the end of the simulation, the resulting link travel times are used to find the shortest path between each OD pair for each assignment interval.

In the second iteration, half of the vehicles use the original shortest paths for each assignment interval and the other half use the new shortest path. The process continues by adding a new shortest path at each iteration, until a maximum number (previously defined) is achieved. Thus, if the maximum number is five, the first five iterations are used to find the best minimum five paths for each OD pair for each assignment interval. In this case, in the fifth iteration,1/5 of

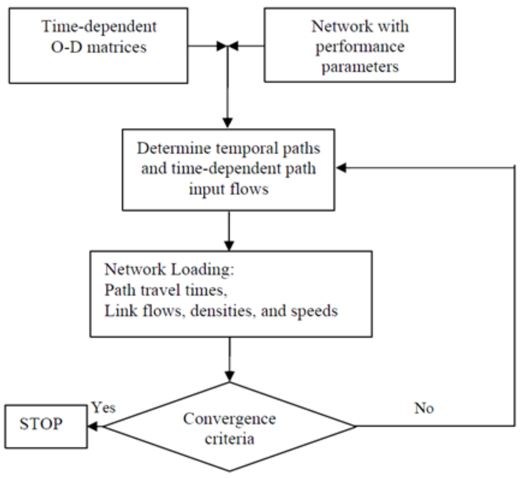

Time-dependent O-D matrices
Network with performance parameters
Determine temporal paths and time-dependent path input flows
Network Loading: Path travel times, Link flows, densities, and speeds
Convergence criteria
Yes
No
STOP

Figure 3.2.8: Complete structure of Dynameq model. *Source: Mahut et al. (2004).*

the vehicles use each path for each given assignment interval. These iterations belong to the path generation phase of the DTA procedure.

During the remaining iterations (which are called the DTA convergence stage), the number of vehicles that use each path for each OD pair and for each assignment interval is adjusted before each iteration in order to balance travel times. In Dynameq, there are two algorithms available to perform this process, both based on the known Method of Successive Averages (MSA). Finally, when the route selections are such that the travel times on all paths are approximately equal for each assignment interval and for each OD pair, then the network is in a DUE state.

Dynameq has the following assignment algorithms available:

- Regular MSA: this algorithm adjusts path flows by identifying the minimum path for each OD pair and assignment interval, after each complete execution of the simulation. The flow of the minimum path will be increased and the flow for the remaining paths in the network will be decreased. The amount of flow added to the shortest path is proportional to $\frac{1}{n}$, where n is the number of iterations.

- Flow Balancing MSA: this algorithm adjusts path flows by evaluating the travel times of all the used paths and by calculating the average travel times. The flow is increased in all paths with travel times lower than this time average, and is decremented in all paths with travel times above the average. The amount of added or removed flow is proportional to the difference between path travel time and travel time average.

Since route choices are adjusted at each iteration, the DTA model must converge to equilibrium conditions. The convergence is measured by comparing measured travel time with the minimum travel time, for each OD pair and for each assignment interval. The convergence measure is the the difference among these two values splitted by the minimum travel time.. This measure takes into account the current travel time, so it can determine if the difference is significant or not. Dynameq calculates this Relative Gap for each assignment interval and for each DTA iteration.

3.2.5 DynaMIT

(Dynamic Network Assignment for the Management of Information to Travelers)

DynaMIT was developed by Moshe Ben-Akiva et al., at the Massachusetts Institute of Technology (Cambridge, Massachusetts, USA) in 1997. DynaMIT is a real-time dynamic traffic system that provides traffic state forecasts and travel guidance, generates information for drivers and guides them to take the best decisions. In order to generate consistent information, two main functions are developed: state estimation and prediction. Two simulators (demand and supply) interact to perform these tasks. DynaMIT compensates for the level of network detail by using a reduced computational load without compromising the integrity of the results.

The global structure of DynaMIT is organized around the two previously mentioned main functions: state estimation and prediction-based guidance generation. The model uses both real-time information (which comes from the surveillance and control systems) and “off-line” information (which comes from the network description, database of the historical network conditions, etc.). In the diagram shown in Figure 3.2.9, we can see the DynaMIT structure.

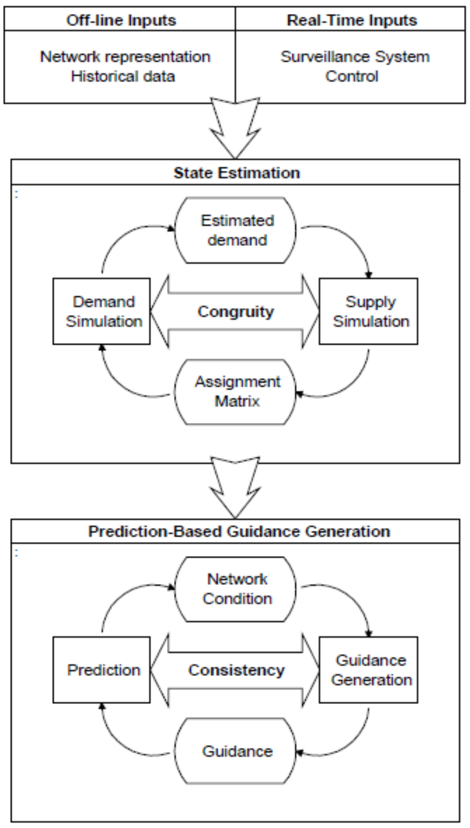


Figure 3.2.9: DynaMIT structure. *Source: Ben-Akiva et al. (1998).*

The state estimation module determines the status of the network and the demand levels through historical and surveillance system data. Two simulation tools are used iterativelyused: the supply and the demand simulators.

The demand simulator estimates and forecasts origin-destination flows and drivers decisions in terms of departure time, mode and route choices. Using this data, a first demand estimation is performed. Then, the supply simulator simulates the interaction between the demand and the network, producing assignment matrices and transforming the OD flows into link flows. Then, the demand simulator uses the assignment matrices and the real-time observations in order to better estimate demand. This cycle is repeated until consistency between demand and supply is achieved.

The prediction-based guidance generation module provides anticipatory guidance using data from the state estimation. Traffic forecasting is performed for a given horizon (for example, one hour). The demand simulator and the supply simulator are also used for the prediction. The guidance generation is based on an iterative process between traffic forecasting and candidate guidance strategies. The system imposes consistency between travel times, based on orientation and travel times resulting from traveler reactions to the guidance.

The quality of the prediction depends on the quality of the state description and on the time horizon. The network state is usually estimated in order to incorporate the available information in a timely fashion before a new prediction is computed.

DynaMIT simulators must be able to simulate different levels of aggregation. Furthermore, capturing driver response to the information requires disaggregated representation, where almost every type of driver behaviour is included. OD estimation and prediction is performed at an aggregated level, and the models must be consistent with the input data of surveillance systems, which is available at an aggregate level.

The mesoscopic simulator transforms historical demand into “informed” demand by capturing the effect of the information. It then estimates the demand as a reflection of daily fluctuations. This process is shown in Figure 3.2.10. Each row corresponds to a different aggregation level, and each column to a specific demand.

The demand simulator transforms OD historical matrices (base 1) into a disaggregated description of the estimated demand (base 6). The historical OD matrices are disaggregated into an explicit list of drivers (base 2), using social-

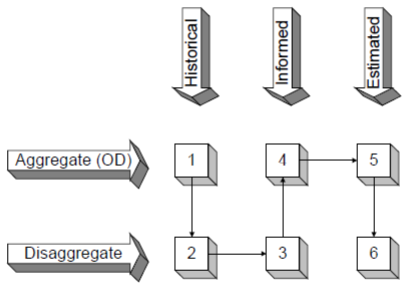


Figure 3.2.10: Demand simulator. *Source: Ben-Akiva et al. (1998).*

economic external information and behaviour models that capture habitual behaviour. The impact of the information and guidance for driver decisions is simulated using disaggregated behavioural models in order to obtain informed demand for base 3. This disaggregated informed demand is successively aggregated to the OD level (base 4). The OD estimation algorithms use data from the surveillance systems to compute the difference between the aggregate representation of informed demand (base 4) and estimated demand (base 5). From the final disaggregation, a complete list of drivers is obtained (base 6).

The informed demand (base 3) is obtained by applying disaggregated behavioural models that capture various route decisions, including whether or not to make a particular trip from the origin to the destination. They also capture departure time, mode and route choice. DynaMIT models route choices in three specific contexts:

- The usual choice of departure time, mode and route.
- The decision to change some of the usual choices as a response to information received before the departure time from the trip origin. This decision is called the "pre-trip decision".
- The decision to change the currently-followed route, in response to the information received during the trip. This is called the "en-route decision".

DynaMIT includes disaggregated behavioral models for the above-mentioned situations. Each person confronted with a choice is individually represented so that the model can be applied and eventually translated into vehicle movements on the network.

The estimated demand (base 5) is obtained using statistical models with the aim of replicating observed data collected in real-time from the surveillance systems. DynaMIT presents a dynamic OD estimation process based on a Kalman filter algorithm and on an auto-regressive process (Ashok and Ben-Akiva (1993)). The auto-regressive process captures the dynamic evolution over time for the state variables in the Kalman filter.

The prediction module performs in the same way as the estimation module. Statistical models are the only thing that change because they use data from the surveillance system in order to estimate OD matrices. On the other hand, the OD matrices are predicted by applying to the deviations an auto-regression process between the informed and the estimated OD matrices. This is similar to the technique proposed by Ashok and Ben-Akiva (1993).

The mesoscopic simulator obtains its input from the list of drivers produced by the demand simulator, and it simulates trips across the network. As a result, it obtains a wide range of network performance indicators like travel times, flows, and densities.

This supply simulator combines a microscopic traffic representation (where each car is individually represented) with macroscopic models that capture the traffic dynamics. Vehicles are represented individually in order to model the effect of en-route information on driver decisions. Macroscopic models of traffic dynamics meet the requirement of real-time performance.

The network representation consists of static and dynamic components. The static component represents the network topology. It consists of a set of links, nodes and loading elements. The nodes correspond to network intersections, while the links represent unidirectional paths between them. The loading elements represent locations where demand is generated or attracted.

Dynamic Network Loading

Traffic dynamics are captured by two main models: a queuing deterministic model and a speed model.

The dynamic components are designed to capture some aspects of the traffic dynamics, and they are continuously updated. Each link is divided into segments

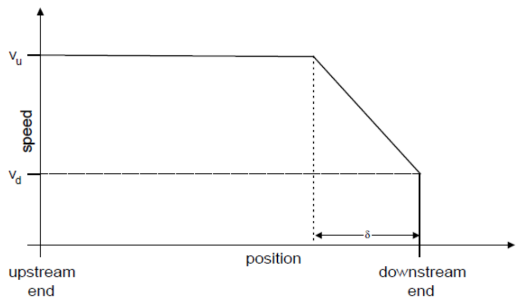


Figure 3.2.11: Speed model. *Source: Ben-Akiva et al. (1998).*

that capture variations in traffic conditions along the link. While many segments are previously defined, additional segments can be dynamically created to capture the presence of incidents. Each segment has a capacity constraint at its end (the end where the flow is directed). Depending on the nature of the segment, this capacity limit can stem from the physical characteristics of the road or the dynamic occurrence of an incident. Each segment has a "moving part" and a "queuing part". The moving part represents the portion of the segment where vehicles are moving at certain speeds. The queuing part represents vehicles that are queued up.

The speed model is based on the following hypothesis: for the moving part of the segment, there are two speeds defined. The speed of the initial part of the segment (upstream) (v_u) is a function of the average density in the moving part of the segment. The speed at the end of the segment (v_d) is the speed at the start of the next segment. An acceleration/deceleration zone is defined with length δ at the end of the moving part. Before this zone, each vehicle is moving at a constant speed. Inside the zone, the speed of vehicles varies linearly as a function of position, as shown in Figure 3.2.11.

The speed function can be written according to Equation 3.2.6.

$$v\left(z\right)=\begin{cases} v_u & if\, 0\leq z\leq L-\delta \\ \lambda\left(z-L\right)+v_d & if\, L-\delta\leq z\leq L \end{cases} \tag{3.2.6}$$

where:

$\lambda=\frac{v_d-v_u}{\delta}$

L = section length

The relationship between density and speed can be given by Equation 3.2.7.

$$v_u = v_0 \left(1 - \left(\frac{k}{k_{jam}}\right)^{\beta}\right)^{\alpha} \tag{3.2.7}$$

where:

v_0 is the free flow speed.

k is the density.

k_{jam} is the jam density.

α, β are parameters.

The queuing model is a family of models. Each specific queue status (formation, dissipation, blockage, etc.) is captured by a different model. As an example, the position $q(t)$ of a given vehicle joining a dissipating queue at time t is given by Equation 3.2.8.

$$q(t) = q(0) + l(ct - m) \tag{3.2.8}$$

where:

$q(0)$ is the position of the end of the queue at time 0.

l is the average length of vehicles.

c is the output capacity

m is the number of moving vehicles between the considered vehicle and the end of the queue at time 0.

This is illustrated in Figure 3.2.12.

Flow Assignment

The guidance generation component provides descriptive and prescriptive guidance. Descriptive informs travelers about traffic conditions they are likely to find

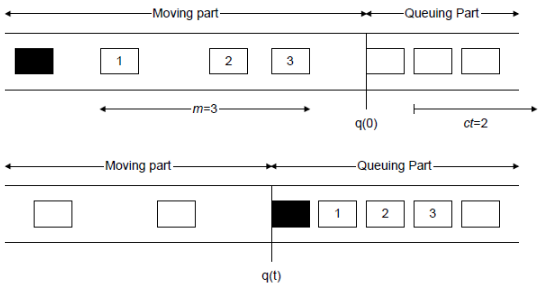


Figure 3.2.12: Queuing model in DynaMIT. *Source: Ben-Akiva et al. (1998).*

in the possible routes from their current location to their destination. Prescriptive recommends a route for passengers based on the expected traffic conditions along the alternative routes. In both cases, the guidance is called "descriptive or predictive" because it is derived from forecasts of anticipated traffic conditions and from the link travel times when the travelers pass through them.

The role of the estimation process is to reproduce a description of the network state that fits as much as possible to the available real-time data. As DynaMIT is designed to operate in real time, it is not possible to calibrate all the parameters. Therefore, it focuses on estimating the OD matrices using real-time data. As mentioned above, disaggregated behavioral models are combined with an aggregated OD estimation algorithm in order to obtain a complete description of the "pre-trip" demand at a single driver level. This demand is loaded into the network by the supply simulator. The behavioral models also apply demand route simulation, in order to capture the impact of the information on travelers who are already in the network. The resulting network state is compared with the observed data from surveillance systems. In case of discrepancy, a new "on-line" state is estimated, with the supply simulator providing a better assignment matrix. Moreover, the supply simulator parameters (such as capacities or speed/density model) have to be adapted to reduce these discrepancies. The last calibration is performed off-line. The OD estimation algorithm requires assignment matrices that transform the OD flows to link flows. These matrices can be obtained from the supply simulator, because they describe how OD flows have been distributed throughout the network using several routes connecting origins and destinations.

The information generation module aims to generate information and guidance, which are consistent with traffic predictions for travelers. The last estimated network state is used as a starting point. Tested guidance strategies and recently disseminated information are also used. The demand and supply simulators are used in almost the same way for prediction and for estimation. The main difference is the OD matrices. In order to predict future OD matrices, DynaMIT needs the calibrated autoregressive process.

Traffic simulation is performed in two phases: the update phase and the advance phase. During the update phase, the most time-consuming calculations are performed, where traffic dynamic parameters (densities, speeds, etc.) are updated. The advance phase calculates at a microscopic level, where the vehicles are driven to their new positions.

Based on the resulting predictions, an information strategy is generated which is based on travel times using a combination of new and original predictions. This strategy is also used by the demand and supply simulators to make a new prediction. This process is iterated until convergence is achieved. The algorithm is an extension of the MSA.

3.2.6 DYNASMART

(Dynamic Network Assignment Simulation Model for Advanced Roadway Telematics)

DYNASMART was developed by the research team of Hani Mahmassani at the University of Texas (Austin, Texas, USA), partly as a result of the thesis of Jayakrishnan (1996). It was designed as an assignment and simulation model for analyzing and applying ITS in traffic networks, for which a description of flow traffic dynamics in time and space is required. In this model, traffic flow is simulated in a microscopic way, based on the continuity equation and on a modified speed-density relationship by Greenshields. The simulation logic combines a microscopic simulation level, for the vehicles representation, with some macroscopic descriptions of the traffic interactions. This obtains very accurate solutions with only a fraction of the computational cost required by only microscopic simulation.

Time-dependent assignment consists of allocating the desired trips into the network, in a way that is consistent with the spatial and temporal traffic processes taking place in that network. In the case of DYNASMART, the vehicles are

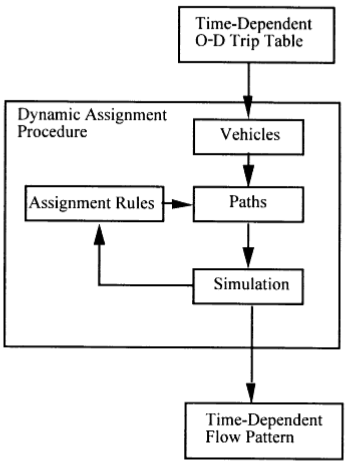

Time-Dependent O-D Trip Table
Dynamic Assignment Procedure
Vehicles
Assignment Rules
Paths
Simulation
Time-Dependent Flow Pattern

Figure 3.2.13: Diagram of DTA procedure in DYNASMART. . *Source: Mahmassani (2001).*

generated following origin-destination time-dependent matrices and they are allocated into the routes specified by the reassignment rules. The time-dependent flow pattern is simulated by loading the vehicles into the network and by representing their movements (using a dynamic network loading model). The obtained results can be used in the next iteration of the assignment procedure.

Figure 3.2.13 shows a diagram of the DTA procedure in DYNASMART.

DYNASMART considers four assignment rules based on different behaviour assumptions and the interpretation of the time-dependent flow patterns in the network.

The first rule determines user paths through the network with an aim toward minimizing global cost to the system (in this case, total travel time). The system optimal behaviour hypothesis corresponds to a strategy of providing information to the users in order to guide them through individual paths that are optimal for the system as a whole. A system optimal assignment pattern usually does not correspond to an equilibrium solution, where the users can improve their individual travel times without taking into account the increasing travel time of the total system.

The second assignment rule agrees with the DUE behaviour hypothesis, previously commented on. In this case, no user can unilaterally improve their individual cost by changing routes. Such a state is a result of the system's evolution, where the user can learn from the provided information.

The third rule corresponds to a set of response rules to the strategy of providing information under which users receive information describing the links travel times. This set consists on limited changes of path and selection rules, that include as a special case: a rule of change that always choices the shortest path based on the current conditions.

The fourth assignment rule is a special case of the previous rule, in which a vehicle is assigned the best path at the origin of the trip. This type of assignment can occur when the traveler can see an information system at the moment of departure and selects the shortest route under current traffic conditions. However, it doesn't take into account future system congestion.

Dynamic Network Loading

Once the network is represented with link characteristics and control parameters included, the simulation component chooses a time-dependent loading pattern and processes vehicle movements through the links. Transfers among these links take into account the specified control parameters. These transfers among links, which are determined by path processing and path selection rules, require instructions that direct the vehicles to the desired link as they approach the output node.

As mentioned above, DYNASMART uses macroscopic traffic flow relationships to model the flow of vehicles into the network. However, while other macroscopic models do not take into account individual vehicles, DYNASMART moves vehicles individually or in packets; it is for this reason that DYNASMART is considered a mesoscopic simulation model. Usually, traffic simulation consists of two main components: link flow propagation and node transfers. Figure 3.2.14 shows an outline of the DYNASMART model's main steps.

Link flow propagation
This process moves the vehicles into the network links during each time interval (simulation step). The network links can be subdivided into sections or smaller segments, according to the simulation's purposes. In DYNASMART, the link density of vehicles during the simulation step is determined from the

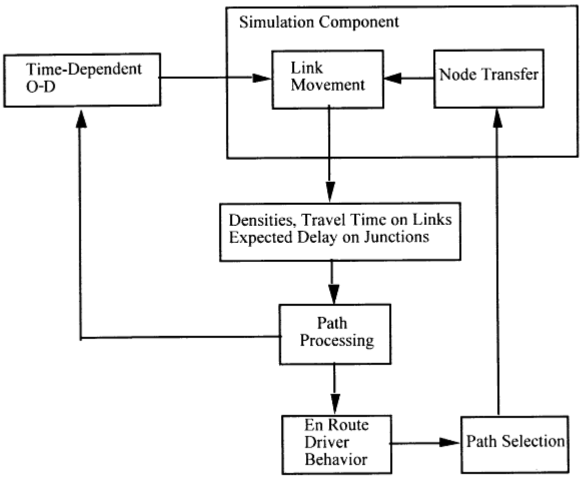


Figure 3.2.14: Structure of the simulation-assignment model DYNASMART. *Source: Mahmassani (2001).*

solution of the continuity equation using the finite differences method. It provides the density,the outflow, and the inflow from the previous step. Using the last determined density, the speeds of the corresponding sections are calculated by modifying the Greenshields speed-density relationship, shown in Equation 3.2.9.

$$V_i^t = (V_f - V_0)\left(1 - \frac{K_i^t}{K_0}\right)^{\alpha} + V_0 \tag{3.2.9}$$

where:

V_i^t is the average speed in section i during simulation step t.

K_i^t is the average density in section i during simulation step t.

V_f is the maximum average speed.

V_0 is the minimum average speed.

K_0 is the jam density.

α is the parameter used to capture speed sensitivity from density.

Node behaviour
This dynamic network loading component is responsible for transferring the vehicles from one link to the next link in its assigned route, or from one section to the next. This transfer is performed at the intersections of the network, i.e., at nodes. The flow is assigned or divided according to the dominant control strategy in that node. Some of the outputs of this component are the number of vehicles that rest in queue or the number of vehicles added or subtracted from each link section at each simulation step. The limitation of the inflow/outflow capacities reflects a wide range of traffic control measures, for both intersections and highways.

Flow Assignment

One of the important characteristics of DYNASMART is its explicit representation of individual traveler decisions, especially route choices, at departure time and also en-route. DYNASMART incorporates behavioural rules governing route choices, including the special case when drivers have to follow specific instructions from the route-guidance. Experimental tests presented by Mahmassani and Stephan (1988) and later by Mahmassani and Liu (1999) suggest that the route selection behaviour of commuters is rationally limited. This means that drivers seek to profit only above a certain threshold, within which the results are satisfactory and sufficient. This can translate into model rerouting (Mahmassani and Jayakrishnan (1991)), presented in Equation 3.2.10 .

$$\delta_j = \begin{cases} 1 & if\, TTC_j\left(k\right) - TTB_j\left(k\right) > \max\left\{\eta_j \cdot TTC_j\left(k\right), \tau_j\right\} \\ 0 & otherwise \end{cases}, \tag{3.2.10}$$

where:

δ_j is equal to 1 when the user j changes from the last path to the best alternative path, and 0 if it remains into the last path.

$TTC_j\left(k\right)$ is the travel time that user j experiences along the current path, from node k to the destination node.

$TTB_j\left(k\right)$ is the travel time that user j experiences along the best path, from node k to the destination node.

η_j is the relative indifference threshold.

τ_j is the minimum absolute improvement in the travel time that needs to change.

The threshold level may reflect perceptible factors such as indifferenceof preferences, persistence or aversion to change. The different values of the threshold govern user responses according to the information provided and their propensity for change. The minimum acceptable improvement is assumed equal for all users. The results of the experiments performed by Mahmassani and Liu (1999) indicate that it has a minimum value of one minute, while for urban commuters it is about 0,2 minutes.

The route processing component determines attributes (like travel time) for the user behaviour component, given the attributes of the links obtained in the simulator. For this, a multiclass k-shortest path algorithm interacts with the simulation model in order to calculate k different paths for each OD pair. However, to improve the computational performance of the model, the k-paths are not recalculated at every step, but at each pre-specified interval. Meanwhile, the travel times of the last k-path set are updated using the link travel times from each step.

The described model is the main part of a multiclass algorithmic procedure which attempts to optimize network performance by distributing routing information in real time to equipped vehicles, taking into account the different user classes for information availability, provision strategyand behavioural response.

Assuming complete a priori information in the form of time-dependent origins and destinations desired by the user of each class, DYNASMART seeks a DTA to provide the number of vehicles of each class in the links and in the paths of the network, satisfying the system objectives and the conditions corresponding to the behavioural characteristics of each user class. The different user classes are:

Class 1: Equipped vehicles that follow paths in order to achieve a system optimal approach.

Class 2: Equipped vehicles that follow paths which achieve a DUE approach.

Class 3: Equipped vehicles that follow certain rules of changing paths that are limited by their response to the information provided. In this case, the model emulates the behaviour of drivers who receive real time information, such as the best paths based on current link travel times. However, they cannot recognize future conditions.

Class 4: Unequipped drivers that follow specific external paths. In this case,

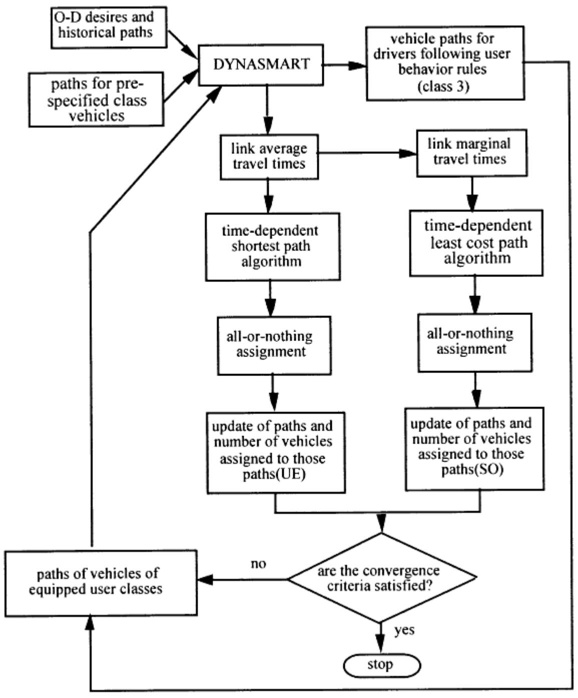

O-D desires and historical paths
paths for pre-specified class vehicles
DYNASMART
vehicle paths for drivers following user behavior rules (class 3)
link average travel times
link marginal travel times
time-dependent shortest path algorithm
time-dependent least cost path algorithm
all-or-nothing assignment
all-or-nothing assignment
update of paths and number of vehicles assigned to those paths(UE)
update of paths and number of vehicles assigned to those paths(SO)
paths of vehicles of equipped user classes
no
are the convergence criteria satisfied?
yes
stop

Figure 3.2.15: Solution algorithm for user multiclass scenario. *Source: Mahmassani (2001).*

the model knows the number of vehicles of this class for each origin, destination, path, and departure time.

The simulation algorithm is solved by extending the reassignment algorithm for a single user class (Mahmassani and Peeta, 1993 and 1995), as shown in Figure 3.2.15.

The structure of the algorithm consists of an inner loop that incorporates a mechanism for system optimal and DUE classes (1 and 2) based on the results of the last iteration of the simulation, experienced travel times, and the associated marginal travel times. The algorithm seeks to address convergence for each class. The third class is not directly involved in the procedure. The paths for these users are obtained from the traffic pattern developed in the network due to the last routing. This differs from classes 1 and 2, which get paths from directions sought in previous iterations. From an algorithmic point of view, there is no direct guidance mechanism involved in obtaining the paths for the users of class 3, other than those based on the assignment strategies of classes 1 and 2. As shown in Figure 3.2.15, they form the outer iterative loop. Users who are not equipped (class 4) are external to the search. They represent background information while the paths remain unchanged.

The simulator included in the DTA model DYNASMART captures the interactions that take place among the four classes of users in the network. This allows evaluation of the resulting flow patterns in the network while the system obtains a solution for the multiclass assignment problem. Moreover, the simulator allows the algorithm to extract information for the search process. What is more, it determines the assignment solution for class 3 users by considering their behaviour rules.

3.2.7 DynusT

(DYNamic Urban Systems for Transportation)

DynusT was developed by researchers at the DynusT Laboratory in the Department of Civil Engineering and Engineering Mechanics at the University of Arizona. The DynusT research team also collaborated with Prof. Xuesong Zhou, of the University of Utah, to develop the graphical user interface NEXTA. In 2012, the RST International Corp. released a new commercial GUI for DynusT, called DynuStudio, which was developed by Dr. Robert Tung at RST International Inc. DynuStudio offers expanded visualization and data management capabilities. DynuStudio is also more scalable and more efficient for large-scale networks.

DynusT is a dynamic traffic simulation and assignment software designed to support engineers and planners in addressing emerging issues in transportation planning and traffic operations. One of their goals is to makeDynusT an open source project so university researchers can conduct further DTA research based on the DynusT platform.

DynusT integrates itself with travel demand models and microscopic simulation models in order to support applications which require realistic traffic dynamic

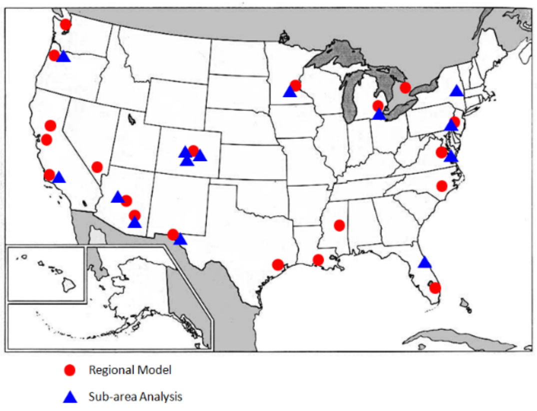


Figure 3.2.16: Where DynusT is used. *Source: www. dynust.net*

representation for a large-scale regional or corridor network. With DynusT, engineers and planners can estimate the evolution of system-wide traffic flow dynamic patterns that are generated by individual drivers seeking the best routes to their destinations as they respond to changes in network demand, supply, or control conditions. Figure 3.2.16 shows where DynusT is (or is going to be) used in the U.S.

The objective of DynusT is to determine network traffic flows and conditions resulting from demand/supply interactions via route choices from travelers in DUE. The DynusT assignment algorithm finds these interactions to determine route volumes, link volumes, and travel times that satisfy this equilibrium condition through iterative procedures.

Dynamic Network Loading

DynusT's dynamic network loading is an anisotropic[1] mesoscopic simulation (AMS) model. The AMS model departs from previous models in that it is a vehicle-based mesoscopic traffic simulation approach that explicitly considers the anisotropic property of traffic flow in the vehicle state update at each simulation step. The advantage of AMS is its ability to address a variety of

[1]Anisotropy: Is the property of being directionally dependent, as opposed to isotropy, which implies identical properties in all directions.

uninterrupted flow conditions in a simple, unified and computationally efficient manner. AMS represents a node-link network, which is generally more memory efficient and temporally scalable than cell-based models that store network-related attributes.

A common problem with current mesoscopic models is their representation of anisotropic traffic flow properties. In the case of a long link, the lead vehicle speeds will be affected by the inflow of the link, implying that an infinitely forward moving shock wave of speed is always generated during the simulation. This violates the basic properties of the AMS model.

AMS is based on two traffic characteristics:

- At any time, a prevailing speed of the vehicle is influenced by the vehicles in front of it, including those that are in the same or adjacent lanes.
- The influence of traffic downstream upon a vehicle decreases with increased distance.

The influence of traffic on a single vehicle is shown as a Speed Influence Region (SIR). The SIR for vehicle i is defined as immediately downstream from vehicle i, in which the stimulus significantly influences the speed response of vehicle i. (Figure 3.2.17). The advantage of the AMS model is that SIR length is independent of the link boundary and network connectivity in the uninterrupted flow condition.

For each vehicle i at the beginning of each simulation interval, the prevailing speed of this vehicle is determined by the Equation 3.2.11, which is the non-increasing speed-density relationship function.

$$v_i^t = \varphi\left(k_i^{t-1}\right), \tag{3.2.11}$$

where:

i is the subscript denoting a vehicle. The index i decreases with vehicles traveling in the same direction on the same link.

t is the superscript denoting a simulation interval.

v_i^t is the prevailing speed of vehicle i during simulation interval t.

k_i^{t-1} is the density of the SIR for vehicle i.

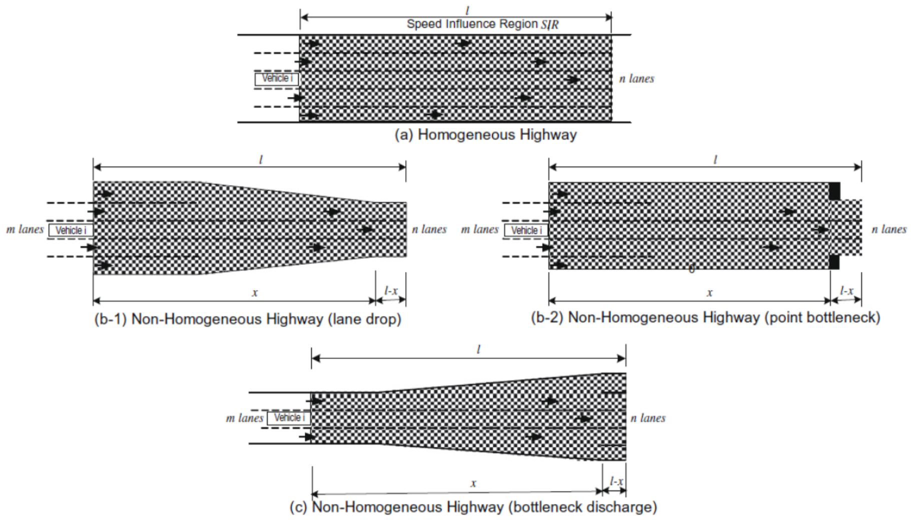

l
Speed Influence Region SIR
Vehicle i
n lanes
(a) Homogeneous Highway
l
m lanes
Vehicle i
n lanes
x
l-x
(b-1) Non-Homogeneous Highway (lane drop)
l
m lanes
Vehicle i
n lanes
x
l-x
(b-2) Non-Homogeneous Highway (point bottleneck)
l
m lanes
Vehicle i
n lanes
x
l-x
(c) Non-Homogeneous Highway (bottleneck discharge)

Figure 3.2.17: Different Speed Influence Region (SIR) situations. *Source: www.dynust.net*

φ is the speed-density function where $\varphi(0) = v_f$ and $\varphi(k_{jam}) = 0$.

At every simulation interval, the AMS model evaluates the speed of the vehicle based on the SIR density of the previous simulation interval. If the SIR spans a homogeneous highway segment, the SIR density is calculated based on Equation 3.2.12; otherwise, Equation 3.2.13 is used.

$$k_i^{t-1} = min\left[k_{jam}, \frac{N_i^{t-1}}{nl}\right] \tag{3.2.12}$$

$$k_i^{t-1} = min\left[k_{jam}, \frac{N_i^{t-1}}{mx_i^{t-1} + n\left(l - x_i^{t-1}\right)}\right] \tag{3.2.13}$$

where:

l is the SIR length.

N_i^{t-1} is the number of vehicles in the SIR at the beginning of the time interval $t-1$.

x_i^{t-1} is the distance between vehicle i and the upstream edge of the lane-drop or the location of the bottleneck point within SIR at time $t-1$.

m is the number of lanes for the SIR area designated by x_i^{t-1}.

n is the number of lanes for the SIR area outside, designated by x_i^{t-1}.

k_{jam} is the queue density.

The AMS model does exhibit both microscopic and macroscopic analytical properties that are observable. The microscopic analytical properties include overtaking conditions and deceleration/acceleration rates. The macroscopic analytical property demonstrated by the AMS model also includes the shock wave.

Flow Assignment

DynusT uses the gap function vehicle-based (GFV) solution algorithm for solving the simulation-based DTA problem. In contrast to the approach based on the Method of Successive Averages, the number of vehicles to be updated with a new path for each iteration and for each origin-destination departure time combination depends on the relative gap function value (the proximity of the current solution to the DUE condition). Vehicles with longer travel time are prioritized for path update selection.

The proposed approach allows DynusT to converge faster than the MSA-based approach, since each origin-destination departure time combination has an individual search direction and step size. When vehicles are loaded with the previously solved baseline (the DUE solution for alternative scenario analysis), the solution appears to be more consistent than the MSA-based approach, as the proposed algorithm avoids over-adjusting flows that are not significantly affected by network changes to the alternative scenario.

DynusT adopts a gradient projection concept in the GFV algorithm, where path flow updates are comprised of both descent direction and step size. Step size is determined directly by the relative gap measurement calculated for the path set of each origin-destination departure time combination. For each combination, paths are sorted in ascending order by travel time, including the auxiliary path for the current iteration. By determining what paths are not performing well, vehicles from such paths are re-assigned to other, better performing paths within the path set. The paths that are found not to perform well will then be removed from the path set.

Due to the fact that individual vehicles within the DynusT framework maintain their own vehicle properties and assigned paths, the GFV algorithm has unique and efficient methods for performing DTA, one of which is that it is the center-point of the Method of Isochronal Vehicle Assignment (MIVA). MIVA is a temporal decomposition scheme for large spatial- and temporal-scale DTA. As the analysis period is divided into what is known as Epochs, the vehicle assignment is performed sequentially in each Epoch, thus improving the model's scalability and confining the peak run-time memory requirement without regard for the total analysis period. The computational requirement continues to be one of the great challenges for DTA in large-scale networks with a long analysis period. A self-tuning scheme adaptively searches for the run-time-optimal Epoch setting during iterations, regardless of network characteristics.

3.2.8 INDY

INDY is a macroscopic DUE model developed in 2004 by Delft University of Technology (M.C.J. Bliemer), TNO-FEL (A.I. Barros and R.J. Castenmiller), and TNO Inro (K.M. Malone and E.H. Versteegt). INDY shows which locations congestion occurs and how the congestion propagates throughout the network. The equilibrium approach of INDY DTA produces chosen paths that are consistent with driver desire to minimize travel costs.

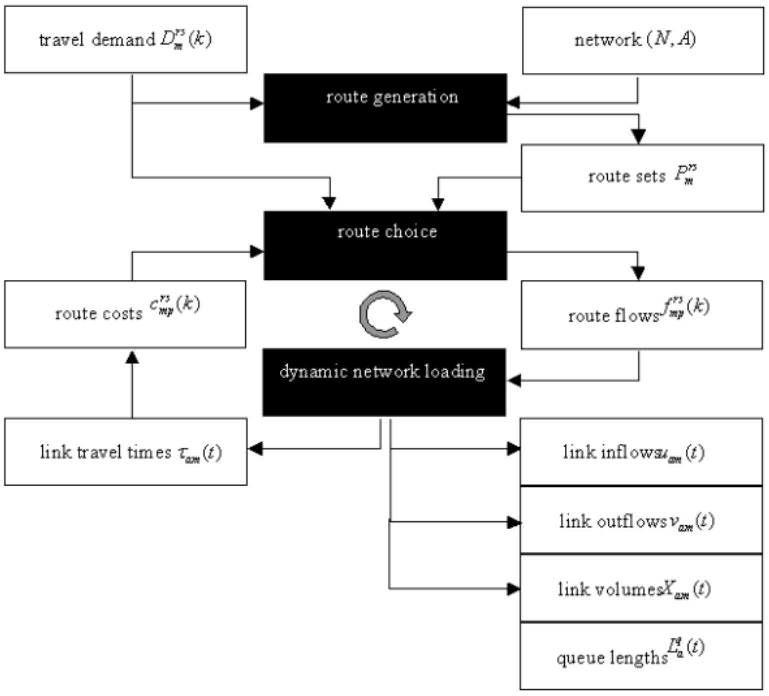


Figure 3.2.18: Flow chart INDY. *Source: Snelder (2009).*

Figure 3.2.18 depicts the model framework of INDY. It consists of three main modules:

1. Route generation.
2. Route choice.
3. Dynamic network loading.

First, the route generation module determines the routes based on network characteristics and travel demand. INDY has three methods implemented for generating the routes: a Monte Carlo approach, an approach using a static traffic assignment and an approach in which a pre-specified route set is used.

The output of the route generation module will be route sets for all vehicle classes describing the available routes between each origin-destination pair.

A disadvantage of a priori route generation is that one is never sure that all relevant routes are included in the route set. Therefore, it is important to create a sufficiently large route choice set so that all relevant routes are included. After running the model it is possible to check whether more routes should have been included or not by running a dynamic shortest path algorithm on the dynamic link travel times.

Second, the route choice module models the behavior of travelers by choosing the best route for them from the set of available routes as determined by the route generation model. The best alternative route depends on the route costs for each of the alternatives and consists mainly of the route travel time, but can include other cost components such as tolls. The outputs are dynamic route flow rates between each origin-destination pair in the network.

Third, the dynamic network loading module is the heart of the INDY model propagating traffic along the chosen routes. Outputs are link inflows, outflows, volume, queue lengths and travel times. These link travel times can be used to compute the route costs. INDY uses three different approaches in the dynamic network loading model.

Last, the new route costs provide feedback to the route choice module, leading to new route flow rates and then performing a new dynamic network loading. These two modules are iteratively performed until convergence is reached.

Dynamic Network Loading

The dynamic network loading model is the heart of the DTA model in INDY. It simulates route flows over the links of the network. The dynamic network loading model is macroscopic, which considers vehicle flow rates.

INDY implements three different approaches to dynamic network loading. The first model uses link performance functions for computing the link travel times in order to propagate the flow through the network. The second model explicitly assumes hard capacity constraints on link inflows and outflows, leading to a dynamic queuing model. Finally, the third model is the so called link transmission model.

Link performance functions The model based on link performance functions is an extension of the single class dynamic network loading model proposed in Chabini (2001), which uses multiclass dynamic link travel time functions. It can handle multiple vehicle classes that have different travel characteristics such as different speeds. These link travel time functions predict, at the time of link entrance, the time it will take vehicles to exit the link. The capacity of the links is taken into account only implicitly: when there are more vehicles on the link, link travel time will increase.

The dynamic network loading model proposed in INDY is a combination of the model proposed by Chabini (1998) and the model proposed by Bliemer and Bovy (2003). The multiclass ideas of the latter work are used to extend the single class model of the first proposal.

Dynamic queuing The dynamic network loading model based on dynamic queuing takes link capacity constraints explicitly into account and determines queues. This model type will predict queues and be able to deal with spillback, unlike the model based on link performance functions which cannot reproduce these phenomena.

In this model, instead of predicting link travel time at the time of link entrance, only the flows are determined on the links based on true traffic conditions. At the end, the link travel times can be derived from the link flows.

Many queuing models that have been proposed in the literature adopt the principle of a moving part and a queuing part on a link. This means that a link is split into two parts: one where the queuing part grows from the head of the link, and another where the remainder of the link forms the moving part. The INDY proposal differs from other queuing models proposed in the literature in the instant of time at which link travel time is determined: the time of exiting the link. This is a key component of its strength, because the later link travel time is determined, the more that can be known from the past about true traffic states over time, and the more accurate is the determined link travel time.

Link transmission The link transmission model implemented in INDY works with fixed time steps. The maximum time step that can be used is equal to the shortest free flow time on all links of the network.

Given the time-dependent route flow rates for a fixed time period, the model determines time-dependent link volumes, link travel times and route travel times

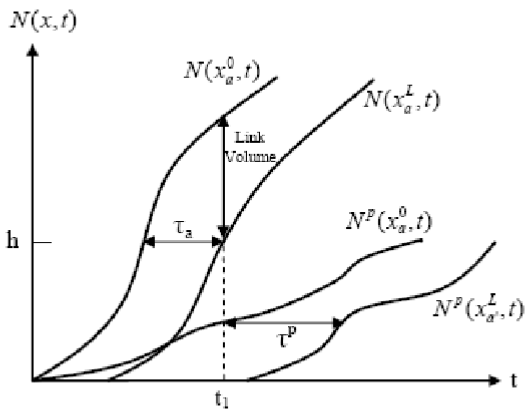


Figure 3.2.19: Cumulative vehicles number as a function of time (INDY). *Source: Snelder (2009).*

in traffic networks.

Traffic networks consist of homogeneous unidirectional links, that can have any length, and they are connected to each other via nodes. A route is a series of links and nodes between the origin and destination nodes. Nodes have no physical length and act merely as a flow exchange medium.

The model determines primarily the cumulative number of vehicles $N(x,t)$ that pass the beginning $\left(x_a^0\right)$ and end $\left(x_a^L\right)$ of link a at certain time t. Only afterward, when vehicles have left the link, link volumes and link travel times are derived from these cumulative vehicles numbers, as we can see in Figure 3.2.19.

This link transmission model is a multi-commodity model, where each commodity corresponds to a specific route. Vehicles are disaggregated by route. INDY keeps track of the routes of the vehicles at all times, when describing the collective motion of the traffic stream. This disaggregation by routes is necessary for using route choice information within the model. For all locations and times, the cumulative vehicle number is the sum of the cumulative vehicle numbers disaggregated by route $N^p\left(x,t\right)$.

The inverse function of the cumulative vehicle number determines the time $t_x\left(N\right)$ at which a vehicle number passed this location. Since the link transmission model solution algorithm only calculates cumulative vehicle numbers

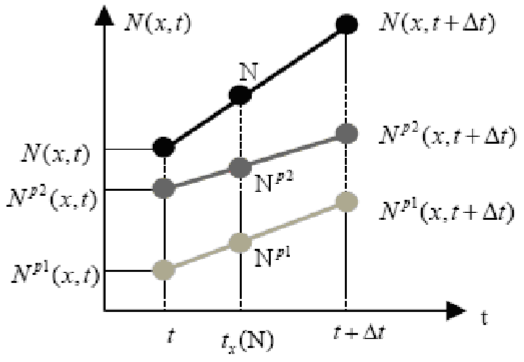


Figure 3.2.20: Linear interpolation of cumulative vehicle numbers. *Source: Snelder (2009).*

in discrete time steps, an interpolation procedure (Equation 3.2.14) might be necessary to calculate $t_x\left(N\right)$, as shown in Figure 3.2.20.

$$N^p\left(x, t_x\left(N\right)\right) = N^p\left(x, t\right) + \alpha\left(N^p\left(x, t + \triangle t\right) - N^p\left(x, t\right)\right) \qquad (3.2.14)$$

for all $p \in routes$.

Flow Assignment

INDY models traveler behavior regarding the choices in a given route set. The route choice module aims to find an equilibrium state in which each traveler individually aims to minimize this route cost. Different vehicle classes and driver types can be considered.

INDY assumes that drivers face a predefined set of available alternative routes from which they choose the best option. It should be noted that route choice is directly influenced by driver type and indirectly influenced by different vehicle types due to different route travel times.The route costs consist of route travel times and possibly other components.

An iterative procedure using the Method of Successive Averages solves the route choice problem for a deterministic or stochastic dynamic user equilibrium, or also for a combination of assignment types corresponding to distinct driver types. The INDY method computes intermediate route flow rates based on the current actual route travel costs. Then these rates are averaged with the route flow rates from the previous iteration and used as the new route flow rates for the current iteration. By performing a new dynamic network loading, these new route flow rates determine new route travel costs and is repeated until they converge. Since INDY starts with a predefined route choice set starting from the first iteration, it already has a good initial solution that converges quickly.

3.2.9 INTEGRATION

INTEGRATION is a trip-based microscopic traffic assignment, simulation and optimization model that is capable of modeling big networks (up to 10,000 links and a half million vehicle departures). The model simulates the departure of vehicles from time-varying, origin-destination matrices. Vehicles are assigned to the networks using a time-dependent multiclass traffic assignment. The model is designed to trace individual movements of a vehicle from its origin to its destination with high time resolution (approx. 0.1 sec.) by modeling vehicle car-following, lane-changing and gap acceptance behaviour.

The model was initially developed in 1983 by Michel Van Aerde, who continued its development until 1999. Since that year, Hesham Rakha has led its development. The name of the model ("INTEGRATION") stems from the fact that the model was developed to integrate freeway and arterial corridor modeling as well as traffic assignment with traffic simulation.

Initially, INTEGRATION was developed as a mesoscopic model. Later, it evolved into a microscopic model in 1995. Further enhancements to the model incorporated a multimodal DTA, tolls, and high occupancy lanes, as well as models for adaptive signal control, transit vehicles, transit priority, microscopic energy, emission, and crash risk.

The main features of the INTEGRATION model are:

- The model combines traffic assignment with microscopic simulation.
- The model captures vehicle dynamics in terms of acceleration behavior.
- The model includes detailed microscopic energy, emission and safety models.
- The model provides flexibility in modeling numerous ITS applications.

Dynamic Network Loading

INTEGRATION combines DTA with microscopic simulation using car-following to model dynamic network loading.

A vehicle within the microscopic simulation model computes its desired speed on the basis of the distance headway and speed differential between a vehicle and the vehicle immediately downstream of it in the same lane. The car-following model explicitly captures the spatial and temporal formation of queues upstream from bottlenecks and the formation of shock waves. The explicit modeling of lane changing behavior means that the model can capture the dynamic change in the capacity of a merging, diverging, or weaving section.

This car-following model was calibrated macroscopically using loop detector data. The capability of the vehicle to achieve its desired speed is constrained by its power to weight ratio, aerodynamic resistance, rolling resistance, and grade resistance. A total of 25 vehicle types are incorporated in the model to capture different vehicle dynamics.

Flow Assignment

In regard to the route assignment under DUE approach, INTEGRATION provides seven basic user equilibrium traffic assignment/routing options:

- Option 1: Time-Dependent Method of Successive Averages.
- Option 2: Time-Dependent Sub-Population Feedback Assignment.
- Option 3: Time-Dependent Individual Feedback Assignment.
- Option 4: Time-Dependent Dynamic Traffic Assignment
- Option 5: Time-Dependent Frank-Wolf Algorithm.
- Option 6: Time-Dependent External Routing 1.
- Option 7: Time-Dependent External Routing 2.

INTEGRATION currently includes pre-trip and en-route driver decisions, i.e., decisions taken prior to the start of the trip or while it is in progress, starting from the time a driver elects to depart from a particular origin with the objective of reaching a particular destination, at a particular time, and by means of a specific vehicle class.

The model has been and continues to be used in Canada, the US and the Netherlands. The model is also being used in Brazil, Korea, Singapore, Italy, and France. The model has been used by researchers, transportation planners, and traffic engineers from both the private and public sector.

3.2.10 METANET

The first version of METANET was developed in 1989 by Messmer and Papageorgiou as a program for motorway network simulation based on a purely macroscopic modeling approach. METACOR (Elloumi et al. (1994)) is an extension of METANET. It also addresses a signal-controlled urban road network based on a realistic macroscopic modeling approach for urban links and junctions.

METANET has two distinct modes of operation. When traffic assignment is not considered, then it may be operated in the non-destination-oriented mode. When traffic assignment is an issue, it must be operated in the destination-oriented mode.

Dynamic network loading

A second-order macroscopic discretized traffic flow model is used for the description of traffic flow on a normal motorway link. This model is suitable for free-flow, critical conditions, and congested traffic. The macroscopic description of traffic flow implies the definition of variables expressing aggregated behavior at certain times and locations. The time and space arguments are discretized. Each segment is macroscopically characterized by: traffic density, mean speed, and traffic volume or flow.

Non-destination-oriented mode In the non-destination-oriented mode, the previously defined traffic variables are calculated for each link segment at each time step by the second-order model proposed by Payne (1971) and extended by Papageorgiou (1990).

For origin links (links that receive traffic demand and forward it into the motorway network), a simple queue model is used (Figure 3.2.21). The outflow

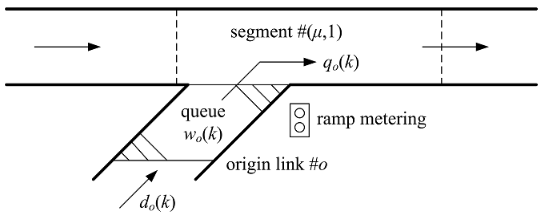


Figure 3.2.21: The origin link queue node in METANET. *Source: Papageorgiou et al. (2010)*

of an origin link depends on the traffic conditions of the corresponding mainstream segment and the existence of ramp metering control measures. If ramp metering is applied, then the outflow that leaves the origin during certain period is a portion of the maximum outflow that would leave in the absence of ramp metering.

Traffic conditions in destination links are influenced by the downstream traffic conditions which may be provided as boundary conditions for the whole time horizon. Options for boundary conditions at destination links include:

- Free outflow: downstream traffic has no influence on the modeled network traffic.
- Pre-specified maximum possible outflow: a queue may build up if the arriving flows exceed the maximum value.
- Boundary traffic density: this influences upstream traffic accordingly.

Motorway bifurcations and junctions are represented by nodes. Traffic enters a node through a number of input links and is distributed to the output links according to: the set of links entering the node, the set of links leaving the node, the total traffic volume entering the node in that period of time, the traffic volume that leaves the node via a certain outlink, and the portion of the total flow entering the node that leaves the node through the given link (called the turning rate of this node). If a node has more than one leaving link, then the upstream influence of density has to be taken into account in the last segment of the incoming link.

Destination-oriented mode When DTA or route guidance is an issue, the second-order macroscopic model (previously mentioned) must be operated in the destination-oriented mode, in which the following additional variables are introduced:

- Partial density $\rho_{m,i,j}(k)$ is the density of vehicles in segment i on link m at a certain time destined to destination $j \in J_m$, where J_m is the set of destinations reachable via link m.
- Composition rate $\gamma_{m,i,j}(k)$ is the portion of traffic volume or traffic density which is destined to destination $j \in J_m$.

The notion of turning rates is generalized to the notion of splitting rates. The splitting rate $\beta_{n,j}^{m}$ is the portion of the traffic volume entering node n at a certain period which is destined to j and which leaves node n at that period through link m. Hence $0 \leq \beta_{n,j}^{m} \leq 1$.

In the case where route guidance takes place at node n with respect to destination j, drivers are directed toward this destination. Since the routing message refers to particular destinations, the influence on route choice is projected to the corresponding splitting rate of the node.

The destination-oriented model version enables DTA or route guidance to be considered if the introduced splitting rates are appropriately specified. METANET has three available user options:

- The user may enter splitting rate values via a corresponding input file. This option may be used for testing specific driver routing scenarios.
- The user may program a real-time route guidance strategy by applying pre-specified couples (node-link).
- If the user is interested in DUE conditions, METANET can calculate automatically the corresponding splitting rates by using appropriate feedback algorithms proposed by Papageorgiou (1990) and Messner and Papageorgiou (1990), which are based on instantaneous travel times. This leads to an approximate dynamic equilibrium since no iterations are involved.

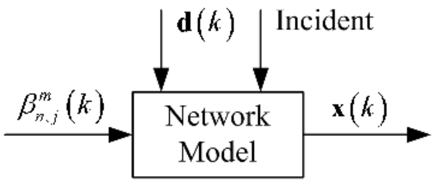


Figure 3.2.22: METANET simulation with given splitting rates. *Source: Papageorgiou et al. (2010).*

Flow Assignment

METANET approximates DTA in the sense of DUE by using a feedback "one-shot" procedure mentioned above. Iterative procedures that may aim at establishing either user-optimal or system-optimal conditions are an alternative to feedback algorithms that can be used for DTA or route guidance.

On the other hand, the typical structure of an iterative procedure toward exact user-optimal conditions is the following (Wang et al. (2001)):

1. Set the initial splitting rates.
2. Run the simulation model (Figure 3.2.22) over a time horizon.
3. Evaluate the experienced (predictive) travel times on alternative routes; if all travel time differences are sufficiently small, stop with the final solution.
4. Modify the splitting rates appropriately to reduce travel time differences; go to (2). The splitting rates can be modeled using decentralized formulas like the Frank&Wolfe (Wisten and Smith (1997)) or PI formula (Wang et al. (2001)).

The described iterative algorithm is employed by an extension of METANET called METANET-DTA. The METANET-DTA software may be used for exact DUE, but at the cost of increased computational time (compared to METANET), which increases by a factor roughly equal to the number of required iterations.

3.2.11 METROPOLIS

METROPOLIS is a traffic management model that simulates vehicle traffic flows and congestion in an urban area. The first version of METROPOLIS was developed by Andre de Palma (Université de Cergy-Pontoise (France)) and Fabrice Marchal (Swiss Federal Institute of Technology-Laussane (Switzerland)) in 1998.

METROPOLIS proposes an interactive environment that simulates automobile traffic in large urban areas. The core of the system is a dynamic simulator that integrates departure time and the route choice decisions of drivers over large networks. Drivers are assumed to minimize a generalized travel cost function that depends on travel time and schedule delay. This simulator is based on behavioral driver information. Route choices are undertaken sequentially by drivers during the journey to work. This system is based on a disaggregated description of commuter behavior. It allows real-time and off-line simulations.

The approach used by METROPOLIS is based on a set of known models:

- Logit model of departure time.
- Dynamic shortest path model.
- Dynamic cost function proposed by Vickrey.

These components are combined with simple behavioral rules. In the dynamics of the simulator, every driver faces two decisions. The first, which is to decide on departure time, follows a multinomial logit model. The second decision, which is the direction, must be taken when the driver arrives at each of the intersections of the network. In this case, METROPOLIS can use a directional decision process deterministic or stochastic to generate the selected routes for the vehicles.

Thus, the METROPOLIS simulator proposes an iterative process that converges to stationary dynamic traffic regimes if the conditions of traffic do not vary.

The computing architecture is depicted in Figure 3.2.23 and is quite similar to that of the initial AIMSUN simulation environment.

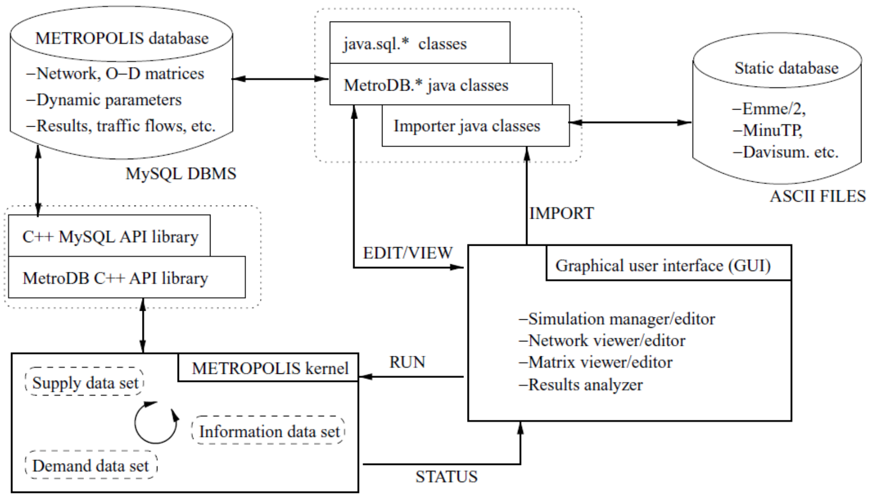


Figure 3.2.23: METROPOLIS software architecture. *Source: De Palma and Marchal (2002).*

Dynamic Network Loading

The METROPOLIS model uses a mesoscopic approach to apply the supply model that describes how vehicles evolve in the road network, given the driving choices of the users. It consists of link specific functions that relate travel time τ_i to either instantaneous inflow $\phi_i(t)$ or instantaneous occupancy $D_i(t)$ and other link parameters $\overrightarrow{P}_i$ such as length and capacity:

$$\tau_i(t) = f\left(\phi_i(t), D_i(t), \overrightarrow{P}_i\right) \tag{3.2.15}$$

When a simulated vehicle enters link i, the simulator computes the travel time to cross this link $\tau_i(t)$. This implies having instantaneous access to any relevant traffic variables on this link, including density $D_i(t)$ and incoming flow $\phi_i(t)$, among others. The computation of the link travel time is therefore deterministic, since it is completely computed at the entry point, given the instantaneous values of the traffic variables on the downstream link. Therefore, the stochastic nature of congestion is not taken into account, but the link parameters could depend on random external events like weather or road condition. This mesoscopic description is a cellular automata description where each link and node would correspond to a unique cell: route decisions happen in node cells and congestion occurs in link cells.

Inside a given link, there is no underlying representation of the vehicles except for the fact that they enter a FIFO stack. The vehicle interaction is limited and described only by a congestion function $f(\cdot)$. This approach implies vertical queuing, since there is no restriction imposed on $\phi_i(t)$, $Di(t)$. To refine the congestion model, they impose that the roads can only support a maximal density that depends on the average length of vehicles. If a vehicle is about to enter a saturated link, it should wait for the downstream queue to discharge so as to accommodate any new incoming vehicle. Lastly, no distinction is made between freeways or urban streets, except that the function $f(\cdot)$ can be different for each link. METROPOLIS is tool-box oriented and does not impose a mandatory form for $f(\cdot)$.

Flow Assignment

A heuristic procedure describes a day-to-day adjustment process for stationary user equilibrium. The core of this system's architecture is the simulation of drivers processing traffic information. The information consists of any type of knowledge about traffic conditions in some part of the network and over a given time period. This information is stored as a separate data set and constitutes a "pool" that drivers can access and update, based on their driving experience. Drivers access the information pool when they perform a travel choice because they need relevant decision data such as expected journey travel time. The idea is that this pool of information is constantly updated and improved as drivers learn from their traveling experience.

The information is processed during within-day simulation as well as from day-to-day. After each simulation of a peak period (or day or iteration), the information pool is refreshed by day-to-day strategies that are available to the end user. At the end of each iteration (day), users can build the best traffic pattern estimate for the following day by aggregating the travel pattern of the last day with the historical traffic patternsof previous days. Such a system converges toward a stationary regime within a few weeks (20 or 50 iterations, depending on the data).

The static assignment models usually consider that the choice of the departure time does not join into the assignment procedure. This is not the case of METROPOLIS, which takes into account the departure time through Equation 3.2.16, in the case of early arrivals to the destination, and through Equation 3.2.17, in the case of late arrivals to destination.

$$C_{ij}\left(t\right) = \alpha tt_{ij}\left(t\right) + \beta\left(t^{*} - \left(t + tt_{ij}\left(t\right)\right)\right) \tag{3.2.16}$$

$$C_{ij}\left(t\right) = \alpha tt_{ij}\left(t\right) + \gamma\left(-t^{*} + \left(t + tt_{ij}\left(t\right)\right)\right) \tag{3.2.17}$$

where:

$tt_{ij}\left(t\right)$ is the minimum travel time at origin i and destination j.

α is the unit cost time (set by the user).

β is the unit cost of early arrivals (set by the user).

γ is the unit cost of late arrivals (set by the user).

t^{*} is the official start time

These different costs are added to the route cost. It is important to note here that this assignment procedure can apply only to those who have a strong interest in arriving at their destination on time, which is generally the case for commuters. Instead, some drivers can have very low values of β and γ (which may be the case for leisure travelers, for example).

The total cost of the trip is equal to the sum costs of early arrival, late arrival, and travel. A driver is scored more negatively for late than early arrival, which can be advantageous in certain situations for shifting departure time. As illustrated in Figures 3.2.24, 3.2.25 and 3.2.26, METROPOLIS influences spreading demand.

As a reference point, Figure 3.2.24 shows traffic demand at the peak hour of the morning. If the capacity of this section is increased by adding a new lane, traffic becomes more fluid, travel time decreases, and a greater number of commuters can leave a little lateryet arrive at the same time. So, in Figure 3.2.25 we can observe an increase in the number of vehicles in a short time. Similarly, a reduction in section capacity leads to transportation demand spreading during peak hour, as shown in Figure 3.2.26.

The METROPOLIS algorithm allows day-to-day learning of traffic conditions.

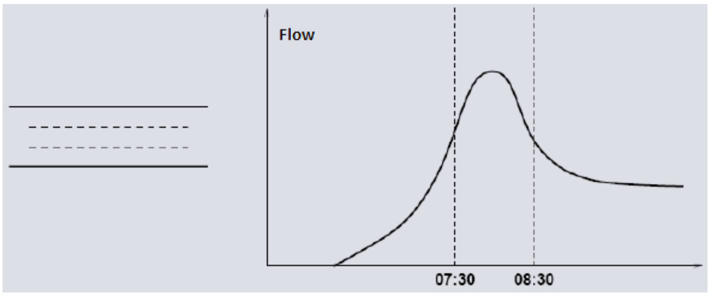


Figure 3.2.24: Profile peak hour demand (morning). *Source: de Palma and Marchal (1996).*

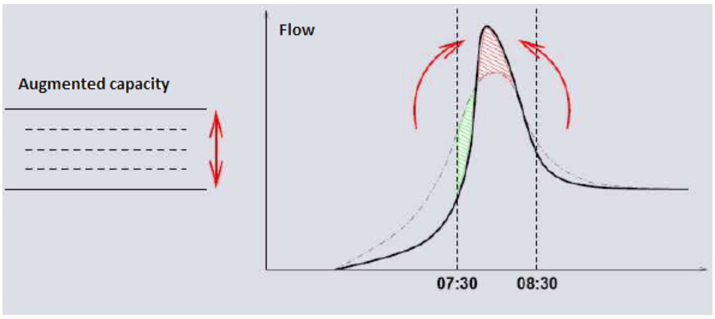


Figure 3.2.25: Profile peak hour demand (morning) due to an increase in capacity. *Source: de Palma and Marchal (1996).*

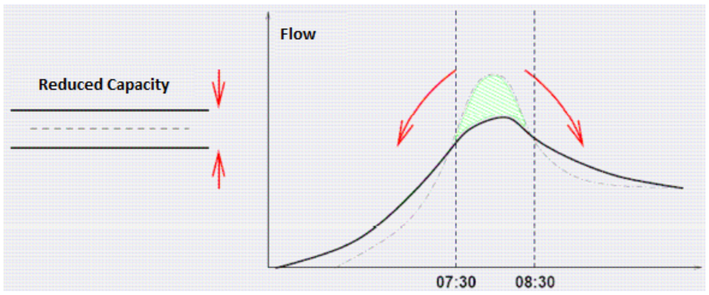


Figure 3.2.26: Profile peak hour demand (morning) after a decrease in capacity. *Source: de Palma and Marchal (1996).*

3.2.12 MEZZO

In 2004, MEZZO was developed by Wilco Burghout at the Royal Institute of Technology (Stockholm) as the mesoscopic component of a hybrid mesoscopic-microscopic simulation model (Burghout (2004)).

Its structure is similar to the link-node queue-server model of DYNASMART, in that it uses speed/density relations for determining vehicle link travel times. Additionally, stochastic node-servers reproduce delays caused by interactions at nodes.

In contrast to DYNASMART, the simulation is event-based, as in the Dynameq model. Moreover, explicit mechanisms ensure correct modeling of start-up shock-waves, in the case of queue dissipation. This results in more realistic behaviour of queue formation and dissipation over both time and space, which are essential for correctly estimating congestion and travel times in the network.

The model is based on a time continuous hypothesis, with different stochastic queue servers for each lane and for each turning movements. It can also use any type of speed/density functions. As a result, traffic dynamics are realistic enough that the model can be integrated into a hybrid model. Experimental results show that MEZZO performs almost as well as hybrid or microscopic models (Burghout et al. (2005)).

The computational performance of MEZZO for small networks (over 250 links) is good. Significant improvements in time calculation are expected through the simplification of: speed/density functions, queue servers, event aggregation, and others.

Dynamic Network Loading

In MEZZO, the traffic network is represented by a graph that consists of nodes and links. The links represent the roadway between the nodes and are considered unidirectional. The nodes are the points where multiple traffic streams join or diverge, such as intersections and traffic origins or destinations. Nodes usually have multiple incoming and outgoing links and are the source of the main conflicts in the traffic stream. The lanes on a road are not represented separately.

Figure 3.2.27 shows an object model of the MEZZO node-link structure.

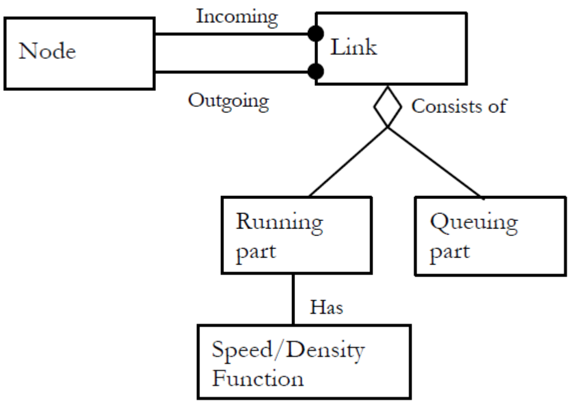


Figure 3.2.27: Object model of the MEZZO network structure. *Source: Burghout (2004).*

The movements of the vehicles are governed by the traversal of links and the crossing of intersections. They are captured by two models: the link model and the node model.

Link Model A MEZZO link is divided into two parts: the running part and the queue part. The queue part begins at the downstream node and grows towards the upstream node when the incoming flow exceeds the outgoing flow on the link. The running part of the link is the part that contains vehicles that are on their way to the downstream node but are not delayed by the downstream capacity limit. So, the boundary between the running part and the queue part is dynamic. It varies depending on the variations in the inflow and outflow in the link.

When a vehicle enters a link, it moves into the running part with a speed that functions as the density on that part. This speed is used to calculate the earliest time a vehicle can reach the downstream node. The speed-density relationship employed in Mezzo is shown in Equation 3.2.18.

$$V(k) = \begin{cases} V_{free} & if\, k < k_{min} \\ V_{min} + (V_{free} - V_{min}) \left(1 - \left(\frac{k - k_{min}}{k_{max} - k_{min}}\right)^{a}\right)^{b} & if\, k \in [k_{min}, k_{max}] \\ V_{min} & if\, k > k_{max} \end{cases} \tag{3.2.18}$$

where:

$V(k)$ is the speed assigned to the vehicle function of the density.

k is the current density on the running part of the link.

V_{free} is the free flow speed.

V_{min} is the minimum speed.

k_{min} is the minimum density where speed is still a function of density.

k_{max} is the maximum density where speed is still a function of density.

a, b are model parameters.

MEZZO allows different sets of parameters to be specified for different link types to capture the different performance characteristics of the various elements of the road network.

After the calculation, the vehicles on the link are ordered according to their earliest exit time. This defines which vehicles are in the queue part at any point in time. The queue part at any simulation time t is defined to contain those vehicles that have an earliest exit time smaller than t. It includes all vehicles that would have exited the link if not for some delay caused at the downstream node.

In MEZZO, the density is calculated in the running part only. This is to ensure that there is no double counting of the delay a vehicle experiences.

Node Model At the downstream end of the links, the node connects to other links. Each of these connections represent a turning movement. These turning movements are limited to vehicles that move toward the corresponding downstream links. MEZZO represents this capacity with queue-servers. The vehicles that are part of the queue segment are processed one at a time by these servers and transferred to the next link if there is available space for them on that link.

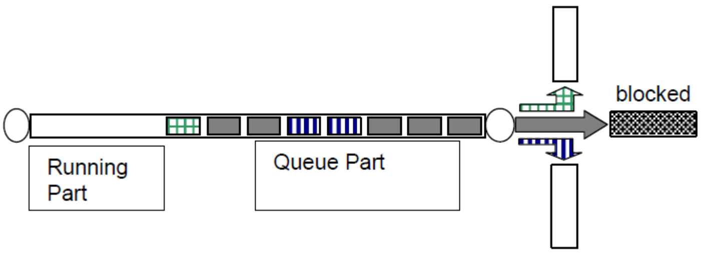


Figure 3.2.28: Queue servers at turning movements serving vehicles. *Source: Burghout (2004).*

Each turning movement has one or more stochastic turning servers that transfer vehicles to the next link. Having one server for each lane with a turning movement instead of one for all of them ensures that headway is distributed more correctly at the outgoing link. The structure of the MEZZO node model allows any stochastic process for the servers, but a truncated normal distribution is usually used.

The turning servers process only the vehicles whose route requires them to make the turning. Figure 3.2.28 shows three turning movements: straight, left and right turn. The server for the vehicles moving straight cannot process more vehicles until the destination link has become unblocked.

The fact that the queue of one turning movement can block access to the other turning movements needs to be represented. Each turning movement has a queue look-back limit. This is the maximum number of vehicles at the front of the queue that a server can "look back" on to find a vehicle that is heading for its specific turning movement.

In Figure 3.2.29 we show an example. For instance, it may be unlikely that the last vehicle in the queue (checkered) could pass the whole queue of vehicles heading straight before making the left turn. On the other hand, it may or may not be possible for the vehicles heading for the right turn (striped) to pass, depending on the exact configuration of the approach. If the look back limit for the left turn is 6 vehicles, then the left turning vehicle in the back of the queue (green) cannot exit until it has advanced to at least position 6 in the queue.

The queue servers in the MEZZO node model provide adequate mechanisms for representing traffic dynamics, such as queue build-up and dissipation. The

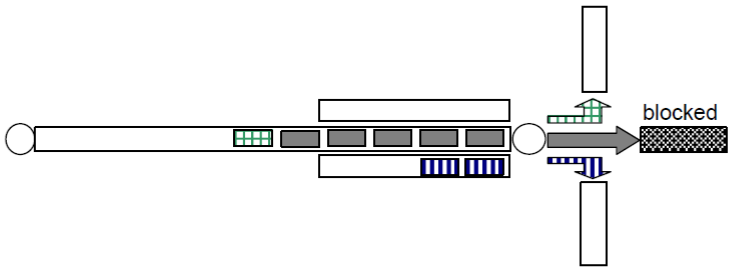


Figure 3.2.29: Turning pockets and blocking of turning movements by other turning movements in MEZZO. *Source: Burghout (2004).*

model explicitly represents the start-up shockwave of a dissipating queue. In order to capture the shockwave that travels upstream when a queue begins to dissolve, the speed of the start-up shockwave is calculated using the density upstream and downstream of the node following the Equation 3.2.19, which is known from traffic literature (May (1990)).

$$\varpi_{AB} = (q_A - q_B) \,/\, (k_A - k_B) \tag{3.2.19}$$

where:

ϖ_{AB} is the shockwave speed.

q_A, q_B are the upstream and downstream flows of the node.

k_A, k_B are the upstream and downstream densities of the node.

So, using this speed, the earliest exit time of all vehicles in the queue-part is updated by calculating when the shockwave reaches them and how long it would then take them to drive to the exit. This solves a common problem of queue-server mesoscopic models, where the space of a vehicle that exits downstream immediately becomes available upstream.

Flow Assignment

The route choice in MEZZO can be performed by two different mechanisms:

- A pre-trip route choice, which selects the habitual path of the driver.

- A route-switching mechanism, which allows drivers to divert en-route from their current path.

Pre-trip route choice In MEZZO, the route choice model is based on a set of known paths and historical link travel times for all links in the network. These travel times are time-dependent. When MEZZO creates a vehicle at an origin, the driver makes a pre-trip route choice according to a multinomial Logit function and based on the set of known routes that connect its origin to its destination according to a certain probability.

The Equation 3.2.20 shows the multinomial Logit probability that a driver will choose certain alternative i.

$$P_i\left(t\right) = \frac{e^{U_i(t)}}{\sum\limits_{j \in S} e^{U_j(t)}} \tag{3.2.20}$$

where:

$P_i\left(t\right)$ is the probability of alternative i being chosen given departure time t.

$U_i\left(t\right)$ is the utility of alternative i given departure time t that is a function of the time-dependent travel time.

S is the set of alternative routes between the given origin and destination.

The pre-trip route choice needs a set of candidate paths for each OD pair and time-dependent historical travel times for the links. The generation of a path set that includes all realistic paths is not trivial. MEZZO uses the following iterative DTA process, shown in Figure 3.2.30.

Starting with the first iterative loop, the shortest path algorithm describes the network and time-dependent travel times for all links in the network. The shortest path algorithm finds the shortest paths for all origins and destinations for each departure time period. All paths that are not in the route database are added.

Then, Loop 2 becomes active. Drivers will choose their routes according to the historical travel times. One of the simulation outputs are the new travel times experienced by drivers as a result of their route choice and interactions with other drivers. These new travel times are averaged with the "historical" travel times that were inputs to the simulation following the Equation 3.2.21.

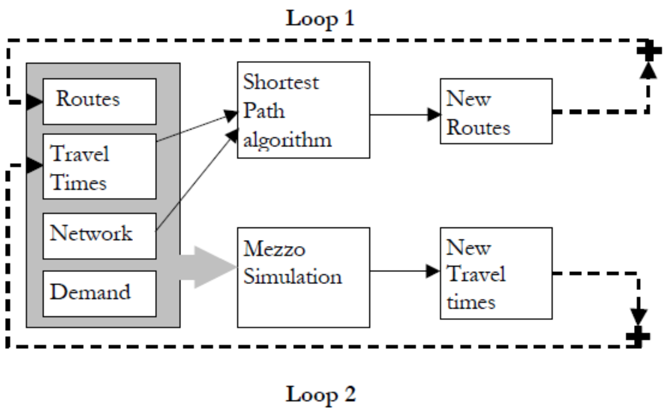


Figure 3.2.30: MEZZO generation of historical travel times and route database. *Source: Burghout (2004).*

$$tt_i^{n+1}(t) = \alpha TT_i^n(t) + (1-\alpha)\, tt_i^{n+1}(t) \tag{3.2.21}$$

where:

$tt_i^{n+1}(t)$ is the historical travel time on link i for iteration $n+1$ when entering at time t.

$tt_i^n(t)$ is the historical travel time on link i for iteration n when entering at time t.

$TT_i^n(t)$ is the simulated travel time on link i for iteration n when entering at time t.

α is the moving average parameter $\in [0,1]$.

Burghout explains that this method of single exponential smoothing ensures stability in the iterated process, which may have oscillatory behaviour if the new travel times simply replace the old ones.

The updated travel times are then used in the next iteration of Loop 1. In this case, the shortest path algorithm may find new shortest paths that are added to the route set. And then a new iteration of Loop 2 is performed. This process is repeated until Loop 1 does not find any new routes and Loop 2 produces travel times that are sufficiently close to the historical travel times it used as input.

This DTA procedure does not guarantee convergence to equilibrium, but it has been shown to converge to a local equilibrium. The path set generation procedure produces realistic paths.

En-route switching The route switching mechanism in MEZZO models the reaction of drivers in order to broadcast information about an incident. The information consists of the link in which the incident happened, the time it started, the expected duration and the expected delay on that link. When such information is broadcast, a certain percentage of the drivers receives the information and determines if the information concerns a link on their current route. If this is the case, the driver adds the expected delay to the expected remaining travel time and compares it to the shortest time alternative route from his current link to the destination. If the alternative is estimated to be significantly better than the current route, the driver can switch the route en-route. Other cases in which this would happen include receiving information about a road blockage or when the driver experiences excessive delay.

3.2.13 VISTA

(Visual Interactive System for Transportation Algorithms)

Another alternative DTA model emulates vehicle dynamics using first and second order models of traffic flow dynamics. VISTA, developed by Ziliaskopoulos and Waller in 2000 at Northwestern University, uses the familiar Cell Transmission Model of Daganzo. VISTA has been successfully used in transportation projects in the U.S. and Europe.

The principal characteristics of the VISTA system include:

- A traffic flow simulator called RouteSim that is based on the Cell Transmission Model;
- A time-dependent shortest path algorithm;
- The path assignment module;
- A dynamic matrix estimation algorithm.

The basic DTA models implemented in the VISTA software are:

- Dynamic Traffic Equilibrium with Drivers Departing at a Fixed Time,
- Dynamic Traffic Equilibrium with Drivers Arriving at a given Window and/or Departing at a given time,
- Dynamic Traffic Equilibrium with Buses and Trucks,
- Dynamic System Optimum,
- Stochastic Dynamic Traffic Equilibrium,
- Dynamic Person Equilibrium (Intermodal/Multimodal)

The general iterative process that computes DTA is outlined in Figure 3.2.31.

Dynamic Network Loading

The VISTA simulator is RouteSim, developed by Ziliaskopoulos and Lee in 1996. RouteSim is a path-based simulator that propagates traffic according to the Cell Transmission Model (CTM) (Daganzo (1994)). The basic idea of the CTM is to simulate movements of small groups of vehicles as they enter and leave sections of each link. Links are divided into cells that are equal in length to the distance traveled in one time step by a vehicle moving at free flow speed. As such, if no congestion exists, all vehicles in a cell will move to the next cell forward in one time step; however, the number of vehicles that move forward is limited by the amount of space available in the next cell, and the maximum flow permitted across the cell boundary. If the number of vehicles attempting to move forward exceeds the space or flow constraints, some vehicles will not be able to move forward, and a queue will form.

In the CTM, vehicle position is tracked only at a cell level, and vehicle speeds are estimated based on transmission time across cell boundaries. While this may be less detailed than other models, the cell length and time step can be reduced for a higher degree of detail. So, this model does not require explicit calculation of speeds, and thus does not rely on the use of speed-density functions to propagate traffic; however, the principles of the cell transmission model are consistent with the hydrodynamic theory of traffic flow. Further, the model can capture many realities of the network, such as traffic signals, by using time-dependent cell capacities and saturation flow rates.

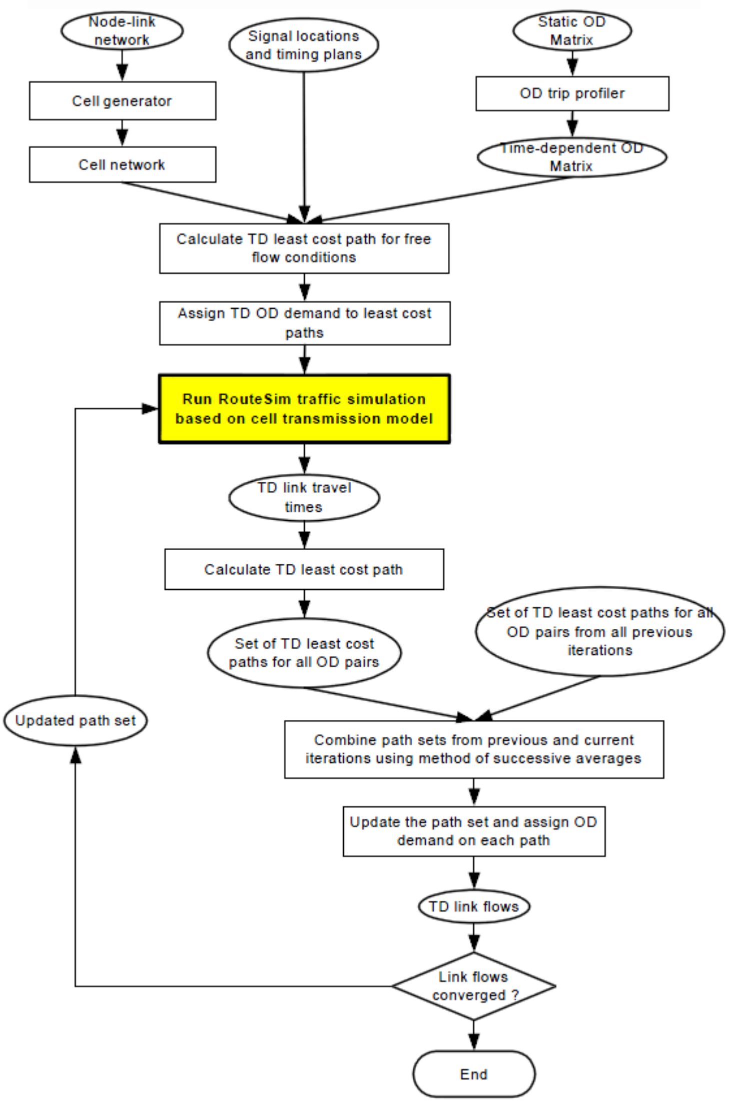

Node-link network
Signal locations and timing plans
Static OD Matrix
Cell generator
OD trip profiler
Cell network
Time-dependent OD Matrix
Calculate TD least cost path for free flow conditions
Assign TD OD demand to least cost paths
Run RouteSim traffic simulation based on cell transmission model
TD link travel times
Calculate TD least cost path
Set of TD least cost paths for all OD pairs from all previous iterations
Set of TD least cost paths for all OD pairs
Updated path set
Combine path sets from previous and current iterations using method of successive averages
Update the path set and assign OD demand on each path
TD link flows
Link flows converged ?
End

Figure 3.2.31: VISTA Dynamic traffic equilibrium with drivers departing at a fixed time. *Source: Mouskos et al. (2003)*

The simulator also includes a car-following model in order to perform microscopic traffic simulation in some parts of the network. Thus, it allows greater detail in capturing traffic movements. VISTA can include traffic signals, ramp meters and priority signals. In addition, any class of vehicle can be modeled by the simulator. And a great variety of traffic detectors can also be included in the model to simulate the deployment of ITS. RouteSim can be used in real time.

In a preprocessing step, RouteSim divides the network links into a number of cells based on their length and free flow speed and transmits vehicles between cells according to the cell density, the downstream cell density, the jam density, and the saturation flow rate. Since this involves only simple comparisons and not complex floating-point calculations, RouteSim can handle networks of thousands of links while maintaining acceptable CPU time and memory requirements. In addition, the simulator was designed to update vehicle movements at varying time intervals, depending on how frequently queue evolution needs to be monitored. Near intersections, incidents, construction zones and other problematic points, the simulator updates the queues every two seconds. For long uniform freeway segments at multiples of that time period, it updates at up to twenty seconds. This results in enormous CPU saving without substantially sacrificing the accuracy of computations.

Flow Assignment

VISTA provides the possibility of using static and dynamic routing. Its route generation procedure uses the results produced by the VISTA simulator in order to develop time-dependent costs. These times are used in constructing traveler routes. VISTA costs are not limited to travel time; they can also include distances, modes of travel, tolls and other factors. The preferences of travelers can also be incorporated into the model. The assignment module recognizes multiple classes of vehicles, constraints in the roads for some types of vehicles, controls, etc.

One of the principal characteristics of the DTA in VISTA is a time-dependent shortest path algorithm that considers traffic conditions when finding the best path to a destination. It is based on the traveling behavior of the user (either departure time or arrival time). The VISTA system uses an iterative solution to assign the time-dependent paths for each OD pair, where path demands are distributed and redistributed among paths until equilibrium is reached.

The assignment algorithm includes an inner demand update loop and an outer path generation loop. In each inner loop, the path demands are loaded into the network and a simulation (RouteSim) is run to obtain path costs and path cost derivatives. For each OD and for each departure time, the demands on the existing path set are updated in such a way that the path costs for all existing paths approach equilibrium. This is based on an approximation of path cost. The updated demands are then loaded into the network and the simulation is repeated. The inner loop is repeated until equilibrium is reached for all OD pairs for all departure time combinations, or until a maximum number of iterations is reached. Next, the time-dependent shortest paths are calculated. If equilibrium is reached, and all of the shortest paths are already included in the path set, then the algorithm terminates with an equilibrium solution. Otherwise, the shortest paths are added to the path set and an inner loop cycle is repeated with the updated path set. This algorithm uses traffic simulation (RouteSim) to ensure that traffic is propagated according to maximum flow and jam density rules.

3.3 Summary and Contributions

In order to accomplish the thesis objectives listed in Chapter 1, we first needed to investigate the relevant simulation-based DTA models present in the literature. In this chapter, we have presented the conducted literature review of the different DTA models based on simulation.

First, we have distinguished three groups of DTA models based on the level of detail with which each of them represents the studied system. From low to high fidelity: macroscopic, mesoscopic and microscopic simulation models. In this chapter, we have briefly summarized the characteristics of each group and we have mentioned the main representatives.

Second, we have carefully reviewed thirteen relevant DTA models based on simulation presented in the previous classification. For each of the examined models, we have analyzed not only the approach used to reassign the flow but also the dynamic network loading component.

To the best of our knowledge, there did not exist in the literature a state of the art about simulation-based DTA models with the high level of detail that we have presented in this chapter. Thus, the first contribution of this thesis is this complete literature review of DTA models based on simulation.

Chapter 4

Proposed DTA

4.1 Introduction

The objective of the research presented in this dissertation is to develop a DTA model based on mesoscopic simulation. As we commented in the previous Chapter 1, one of the reasons for developing a new DTA model is to overcome some of the limitations observed in literature proposals and current practical implementations of flow assignment methods. Other reasons are related to increasing interest in using time-dependent traffic models in urban networks and incorporating them into advanced active traffic management and information systems. In these cases, the key is using mesoscopic simulation to perform the dynamic network loading (DNL) in the DTA scheme. This type of simulation allows a balance in the number of parameters, process efficiency, and the reproduction of certain traffic behaviors that are needed when the proposed DNL is embedded into a DTA scheme.

As we commented on the previous Chapter 1, another reason for developing a new DTA model is that none of the existing models are suitable for easy integration with other traffic procedures like estimating time-dependent OD matrices. Full, smooth and computationally efficient integration of traffic models is absolutely necessary for advanced applications.

This chapter is organized as follows. In Section 4.2, we provide the fundamentals of the new proposed DTA model. Then, Section 4.3 is dedicated to presenting the new DTA model. Following this, we use the rest of the chapter to describe the different components of the model, paying special attention to the components that do not have their own specific Chapter in this thesis: the K-shortest path algorithm (Section 4.3.2) and the time-dependent shortest path algorithm (Section 4.3.5).

4.2 Fundamentals

As discussed in Sections 2.3 and 2.4, a simulation-based variational inequalities formulation of DTA offers a more general approach, despite some drawbacks, and it therefore has been selected for this thesis.

The DTA model developed in this thesis is based on the dynamic extension of Wardrop's Principle, referred to before as Dynamic User Equilibrium (DUE). In order to achieve dynamic equilibrium, we solve the variational inequalities formulation by employing a preventive approach based on an iterative solution algorithm, which is a modification of the Method of Successive Averages proposed by Powell and Sheffi (1982) for a variational inequalities problem.

DTA models for predicting user equilibrium flows on traffic networks are often solved by an algorithm that iterates between two main components until the convergence criteria is satisfied. These two components are referred to as DNL and path flow reassignment. The DNL procedure takes the allocation of travel demands to network paths at each time step as given, loads these by emulating the dynamics of its propagation through the network and computes time dependent path costs (travel time dependent), from that network loading. The path flow reassignment step then uses the time dependent costs obtained in the DNL to adjust the allocation of traffic to paths that will be used in the next DNL.

DTA models based on simulation use a traffic simulator to reproduce complex traffic flow dynamics. In this case, we develop a mesoscopic simulator which is embedded in the proposed DTA framework in order to capture realistic traffic dynamics. A multiclass multilane DNL model based on a mesoscopic scheme is proposed. This model considers a continuous-time link based approach with complete demand discretization.

The proposed DTA scheme includes a time-dependent shortest path component in order to add new paths throughout the procedure when necessary. Shortest path calculation in DTA is a critical issue. The usual approach computes the shortest paths in a kind of one-shot procedure. In this case, link costs change from time interval to time interval, but the shortest paths for a given time interval are calculated as if all links in those paths were reached during that interval and therefore their costs were constant. However, in reality, links are reached at different time intervals and therefore their costs change, depending on when the link is traversed. Hence, a proper approach for a shortest path calculation in a DTA procedure must be a proper time-dependent shortest path procedure.

The process starts with a path set determined by a static shortest path algorithm, which uses path costs in a free flow traffic situation. After the first iteration, the path set can change if a new time dependent shortest path exists as a result of the new cost situation. Thus, the proposed path flow reassignment process must take into account this new path in the next path demand distribution.

Taking into account that the aim of a DTA based on the time-dependent Wardrop principle is to achieve DUE, a measure of how close the solution is from DUE is necessary. A relative gap refined by a departure time interval is the gap measure employed in this thesis.

4.3 The Proposed DTA Model

A description of the proposed DTA model is given in the following.

The proposed DTA model requires the followings inputs: the demand that has to be loaded into the network and some network characteristics like its geometry or control plans. The demand is divided into discrete departure time periods and is represented by a set of time-dependent origin-destination matrices that contains one matrix for each departure time interval. Each matrix contains the number of trips from each origin to each destination of the network at the corresponding departure time interval.

At the initialization step, the algorithm requires a set of shortest paths from each origin to each destination. Here, it is not necessary to calculate time-dependent shortest path for each OD pair for each departure time interval, because this calculation is based on the free flow travel times of the links, and obviously all these travel times are the same for all the time intervals. In this way, the initialization step uses a static shortest path algorithm to calculate the same set of paths for each departure time interval corresponding to each OD pair.

Also, it is important to note here that if we start the flow reassignment process with only one possible path for each OD pair for assigning all the corresponding flows, the possibility of generating false congestion is very high. If this occurs, all the main links of the network can present congestion, and consequently very high costs. In this case, in the next step of the process, we will find a shortest path that will not use these congested links. Thus, we could arrive at a solution very far from equilibrium, and will therefore need more iterations in order to converge. To avoid this phenomenon, the initialization step prepares a small set of paths for each OD pair instead of only one path by imposing fewer iterations on the DTA algorithm in order to achieve equilibrium.

In summary, in the initialization step a K-static shortest path algorithm is executed in order to obtain the best set of paths for each OD pair. The inputs to the K-static shortest path algorithm are the free flow travel costs of each link of the network, which are easy to calculate. We assume that the vehicles circulate, if it is possible, at the maximum allowed speed in each of the links. So the free flow travel cost (in this case, travel time) is calculated for each link by dividing the length of the link by the maximum allowed speed in this link. Also, we need to know the number of initial M paths that depends on the network's characteristics. The most convenient way to select this number would be to test different options during the network calibration process. Obviously, the output of the K-static shortest path algorithm (with K=M) is one set of the best M paths for each origin-destination pair (the same for all the departure time intervals).

After the initialization step, and while the maximum number of DTA iterations (established by the user before starting the process) is not achieved, the proposed algorithm iterates between two main components: the flow reassignment and DNL.

Firstly, we try to assign the demand of the different OD pairs (od) at each departure time interval (t) to the corresponding paths of the initial set of M "best paths" (P_{odt}). For each OD pair and for each departure time interval, we have the number of trips q_{odt} for the corresponding time dependent OD matrix.

The first iteration of the flow reassignment component is performed differently from the assignment of the remaining iterations in the procedure. In this first iteration, the flow assignment is inversely proportional to the path cost. For all OD pairs od, all departure time intervals t, and all paths $p \in P_{odt}$, the flow is assigned by following the Equation 4.3.1.

$$f^k_{odpt} = \frac{1/c_{odpt}}{\sum\limits_{j \in P_{odt}} (1/c_{odjt})} \cdot q_{odt} \tag{4.3.1}$$

where:

f^k_{odpt} is the flow assigned at iteration k to path p from origin o to destination d departing at time interval t.

c_{odpt} is the cost of path p from origin o to destination d departing at time interval t.

In the initial iteration, the input to the flow reassignment method is: the set of initial M paths for each OD pair for each departure time interval, and the time-dependent OD matrices with the trips between origins and destinations for each departure time interval. The output is the specific flow for each path of the initial set of paths for each OD pair for each departure time interval.

After that, the role of DNL is to determine how the reassignment flows in the current iteration propagate along the corresponding paths and generate time dependent traffic intensities in network sections, link travel times, path travel times, etc. In this thesis, we develop a new mesoscopic traffic simulation model to solve the DNL.

The mesoscopic traffic simulation provides the new costs of all links of the network. A time dependent flow propagation implies that link costs can change over time and therefore each link can have a different cost for each time interval.

The DTA procedure uses this information:

- To update the cost of the paths used in the previously performed DNL, in order to know the real current cost of the paths, not the cost assumed before the loading procedure.
- To calculate the new shortest path for each OD pair for each departure time interval. Because link costs depend on the arrival time of vehicles to the link, we have returned to the case discussed above, and the time-dependent shortest path procedure is needed for calculating the new best path. The algorithm also needs information about the network geometry to provide the solution. The output is one shortest path and its cost for each OD pair for each departure time interval.

At this time we must measure how close the proposed flow assignment is to the goal, the DUE. In other words, it is time to evaluate the relative gap function. This convergence criterion needs the cost of the used paths of the current iteration and the cost of the best path for each OD pair for each departure time interval. With this information, we can measure the difference between the total travel costs experienced and those that would have been experienced if the travel costs of all vehicles were equal to those of the current shortest path.

If the relative gap is acceptable (around 5%, Tong and Wong (2000)) then the DTA algorithm ends. Otherwise, if the relative error is not acceptable, the flow assignment is too far from equilibrium and we need to improve it. It is time to begin the second iteration.

If the DTA algorithm reaches the pre-established maximum number of iterations, then it ends. In this case, the corresponding result is the last calculated flow assignment, although it is not an equilibrium solution. If the performed iterations are less than the maximum number of iterations, the algorithm continues by executing a new flow reassignment (a standard iteration, because it is not the first).

In this case, we propose a modification of the Method of Successive Averages to determine how the fraction of the demand for a time interval is split for determining the new time dependent path flow. This can be performed using the travel times experienced on these paths in the previous DNL iteration.

In addition to the current path set for each OD pair for each time interval, we need the new time-dependent shortest path for each OD pair for each departure time in the previous DTA interval, which was calculated from the travel cost obtained in the previous DNL. The proposed flow reassignment procedure acts differently, depending on whether or not the new shortest paths belong to the current set of paths for the same interval. In the second case, we add this new path to the set.

Hereafter, the proposed DTA algorithm proceeds identically to the first iteration until either the convergence criteria or the maximum number of iterations is achieved.

The structure of the proposed DTA model is illustrated in Figure 4.3.1.

Throughout the following sections, we specify each component of the presented DTA scheme.

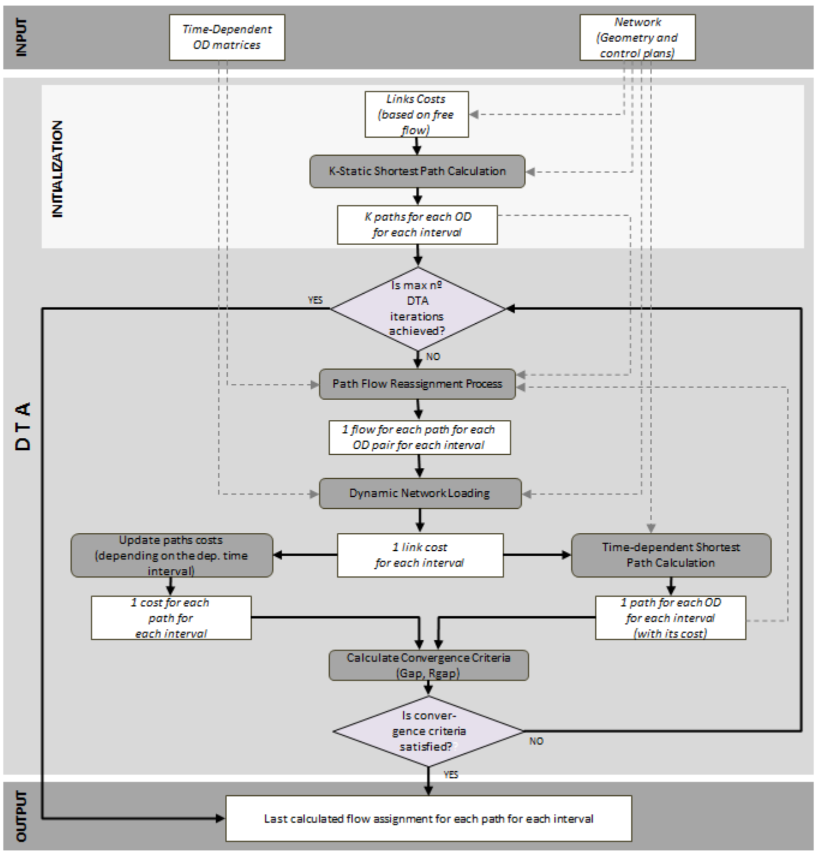

INPUT
Time-Dependent OD matrices
Network (Geometry and control plans)
INITIALIZATION
Links Costs (based on free flow)
K-Static Shortest Path Calculation
K paths for each OD for each interval
DTA
Is max nº DTA iterations achieved?
YES
NO
Path Flow Reassignment Process
1 flow for each path for each OD pair for each interval
Dynamic Network Loading
Update paths costs (depending on the dep. time interval)
1 link cost for each interval
Time-dependent Shortest Path Calculation
1 cost for each path for each interval
1 path for each OD for each interval (with its cost)
Calculate Convergence Criteria (Gap, Rgap)
Is convergence criteria satisfied?
NO
YES
OUTPUT
Last calculated flow assignment for each path for each interval

Figure 4.3.1: Structure of the proposed DTA model.

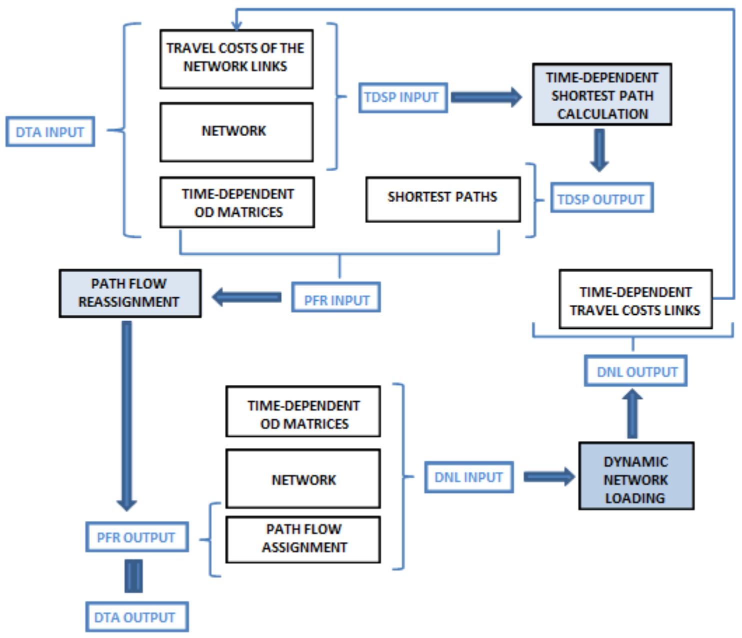


Figure 4.3.2: Outline specification of inputs and outputs for the proposed DTA process.

4.3.1 Inputs and Outputs

First of all, we describe in detail the input and output data that appears throughout the proposed DTA scheme. In Figure 4.3.2, we show the flow of data throughout the main components of the procedure: DNL, flow reassignment, and the time-dependent shortest path algorithm. As we can observe in this Figure 4.3.2, the output of one component is part of the input of the next component, iterating until a convergence criteria is satisfied, i.e., the procedure feeds on itself, with the exception of the starting input specified in the following.

4.3.1.1 Time-Dependent OD Matrices

The demand is defined by a time-sliced OD matrix for each class of vehicle considered by the user. The matrices are time-sliced in the sense that constant demand may be specified for given time intervals. We assume the time-dependent OD trip demands are known a priori for the entire time horizon of interest.

The time-dependent OD matrices specify the number of trips desired from each origin to each destination at each time interval. A typical length of these intervals ranges between 10-15 minutes, depending on the available data. Time-dependent OD matrices capture trip rates and travel patterns. OD demand can potentially vary according to changes in traveler activity patterns, which may vary by day of the week, season, weather conditions or special events.

The objective of DTA models is to describe the assignment of origin-destination flows on the different paths connecting every OD pair corresponding to the cpreviously mentioned DUE state. One criterion for the quality of the results of a DTA model is the coherence between the operational goal of the model, the model itself and the data used, in particular the dynamic OD matrices.

A critical component of such an exercise is the set of time-dependent matrices of OD flows that capture network travel demand. OD matrices, be they static or dynamic, are not directly observable, so the current practice consists of adjusting an initial matrix from indirect measurements. OD estimation methods are typically employed for this purpose.

Link flow counts, which are provided by an existing layout of traffic counting stations, are the most easily available dynamic data sources and the least costly. A wide range of models has been developed to solve the complex problem of estimating a dynamic OD matrix from link counts. The classical OD estimation methods solve a system of linear equations that map the unknown OD flows to observed sensor count measurements.

The emergence of new Information and Communication Technology (ICT) sensors provide new types of real-time data, such as samples of travel times between sensors. They allow new formulations for space-state models that solve the dynamic estimation of OD trip tables. In 2012, Barceló et al. proposed a recursive linear Kalman-Filter for state variable estimation that takes advantage of travel times and traffic counts collected by tracking Blue-tooth equipped vehicles and conventional detection technologies (Barceló et al. (2012)).

4.3.1.2 Traffic Network

The network considered for the DTA model presented here is similar to that used for static network models, with some additional information for links and nodes. The links have the additional attribute "lanes", since the vehicles are accounted for by lane and the nodes require traffic control information, such as traffic signals. The network is composed of the following elements:

Node An intersection of links. Each node has the possibility of including a traffic light control in order to manage vehicles passing from links to this node.

Centroids Special nodes that represent origins or destinations of network traffic.

Link A unidirectional connection defined by two nodes (origin and destination). A link is characterized by its length, the number of lanes and its maximum speed allowed.

Lane A longitudinal division of a link for a stream of traffic. It inherits the attributes of the link.

Turn A movement between two links (incoming and outgoing) at a node. Given a certain pair of links, the turn is not defined between all the lanes of the two links. The turn is characterized by the lanes of the incoming link allowed for going to the lanes of the outgoing link.

4.3.2 K-Static Shortest Path Calculation

4.3.2.1 K-Static Shortest Path Algorithms

A shortest path problem is for finding a path with minimum travel cost from one or more origins to one or more destinations through a connected network. In our case, it is important to know the second or third shortest path between two nodes. For this, k-shortest path algorithms are generally used.

Although a k shortest path algorithm can provide several alternative paths, it is inherently limited by heavy overlapping among derived paths, which may lead to erroneous travel information for users. This means that a significant proportion of links on the first path are overlapped by the second and third path calculated from this method, so the drivers on those links may suffer severe congestion if they act on the travel information. In our case, it represents spatial similarity between generated paths and may not be representative of the heterogeneity found in the set of all paths. In order to obtain a path set that represents more the variety of choice available, overlapping alternatives should be excluded.

On the other hand, more shortest path algorithms exist for building the optimal path based on the node they reach. However, when considering a network with several turn prohibitions, traditional network optimization techniques have difficulty dealing with it. Banned and penalized turns may be not described appropriately in the standard node/link method of defining a network with intersections represented by nodes only. Among all approaches proposed for solving the problem, the most widely used method is network expansion for describing turn penalties, for adding extra nodes and links to original network, and for modifying the network in order to easily implement the conventional shortest path algorithm. The principal advantage is that it can describe the turn prohibition perfectly. A limitation, however, is the computational performance as network size increases, making it unfeasible for very large networks.

4.3.2.2 Lim and Kim Algorithm

The method used in this dissertation is a link-based shortest path algorithm proposed by Lim and Kim (2005), which provides efficient alternative paths for route guidance and which also considers overlapping among paths. This algorithm builds new paths based on both the degree of overlapping between each path and travel cost. It then stops building when the overlapping ratio exceeds its criterion. This algorithm generates the shortest path based on the link-end cost instead of node cost. Thus, it can reflect all turns (such as left-turn, right-turn, P-turn, U-turn and turn prohibition). It constructs a path between origin and destination by link connection, so network expansion is not required.

Lim and Kim proposed a standard link-based shortest path algorithm that expands on the link-based shortest path optimality condition with turn penalty, shown in Equation 4.3.2.

$$LEC\left(o,i\right)+TP\left[link\left(o,i\right),link\left(i,j\right)\right]+LC\left(i,j\right)\leq LEC\left(i,j\right),\ \forall o,i,j\in N \tag{4.3.2}$$

where:

$G = (N, A)$ is the network

N is the set of nodes in network G.

A the set of links of the network G.

$LC\,(i,j)$ is the non-negative link cost required to travel from node i to node j .

$LEC\,(o,i)$ is the link-end cost, or minimum path cost from origin to node i through link (o,i). It refers to the directed link leading from node o to node i.

$TP\,[link\,(o,i)\,, link\,(i,j)]$ is the turn penalty which implies the additional cost at node i from link (o,i) to link (i,j) when turning prohibitions exist.

In this algorithm, instead of the preceding nodes in conventional shortest path algorithms, preceding links are used to memorize the track of the shortest path. The preceding link $PL\,(i,j)$ is the link immediately before link (i,j) on the shortest path.

Let R be the set of all labeled links, R^o the set of unlabeled links and O the set of all connected nodes with origin. The steps of the link-based shortest path algorithm are as listed:

Step 1

> **Label** the link (h,i) , connecting origin node h with node $i \in O$.
> **Enter** link (h,i) into set R, i.e., $R = \{link(h,i)\}$.
> **Set** $LEC\,(h,i) = LC\,(h,i)$ and $LEC\,(h,j) = \infty\,\forall j \neq i$
>
> $PL\,(h,i) = \emptyset$

Step 2

> **Find** an unlabeled link.
> **If** $LEC\,(i,j) + TP[link\,(i,j)\,, link\,(j,k)] + LC\,(j,k) \le LEC\,(j,k)\,Then,$
> $LEC\,(j,k) = LEC\,(i,j) + TP\,[link\,(i,j)\,, link\,(j,k)] + LC\,(j,k).$
>
> $PL\,(j,k) = link\,(i,j)$

Step 3

> **Label** the link (i,j)

Add the link (i, j) to the set R, and delete it from the set R^o.

Step 4

If $R^o = \emptyset$ then STOP, otherwise go to **Step 2**.

This shortest path algorithm generates several different paths by adopting both the link-based shortest path technique presented above and the link penalty method for finding dissimilar ones.

Let:

p_n^{rs} is the n-path for origin-destination pair rs.

l_n^{rs} is the length of n-path for origin-destination pair rs.

P^{rs} is the path set for origin-destination pair rs, $P^{rs} = \{p_n^{rs}, l_n^{rs}\}$.

$ol_{k/n}^{rs}$ is the overlapping length of n-path to the length of k-path for origin-destination pair rs, $\forall P_k^{rs} \in P^{rs}, k = n-1, n-2, ..., 1$.

$Op_{k/n}^{rs}$ is the degree of overlap between the length of n-path and the length of k-path for origin-destination pair rs. $Op_{k/n}^{rs} = \frac{ol_{k/n}^{rs}}{l_n^{rs}}$ $\forall P_k^{rs} \in P^{rs}, k = n-1, n-2, ..., 1$.

Op is the maximum degree of overlap.

Oz is the link penalty; $Oz = \left[\frac{1}{Op}\right]^{\alpha}$

α is the positive parameter.

With these variables, the shortest path algorithm proposed by Lim and Kim (2005) is described as:

Step 0 Initialization

Set Op and $n = 1$.

Step 1

With the link-based shortest path algorithm, find the shortest path.
Add l_n^{rs} and p_n^{rs} to path set; $P^{rs} = \{P_n^{rs}, l_n^{rs}\}$

Step 2 Link cost update (Link penalty)

New link costs of the n-th path = link costs of the n-th path * Oz

Step 3 Path search and Overlapping ratio calculate

$n = n + 1$

With the link-based shortest path algorithm, find the n-th shortest path; $\{P_n^{rs}, l_n^{rs}\}$

Calculate the degree of overlap: $Op_{k/n}^{rs} = \frac{ol_{k/n}^{rs}}{l_n^{rs}}, \forall P_k^{rs} \in P^{rs}, k = n-1, n-2, ..., 1$

Step 4 Convergence test

If $Op_{k/n}^{rs} > Op$ then STOP.

Otherwise, add $\{P_n^{rs}, l_n^{rs}\}$ to P^{rs} and proceed Step 2.

In Step 2 of the algorithm, the link costs pertaining to the shortest path are modified by the multiplying link penalty Oz, which is a degree function of path overlap (Op). If $Op = 1.0$, only one path is generated because all paths are allowed to overlap each other. If Op is below 1.0, more than one path is generated. In our case, we consider $Op = \frac{1}{2}$ and $\alpha = 5.0$.

In Figure 4.3.3, we show the output of the k-shortest path algorithm applied in an urban scenario for different degree values of path overlapping parameter Op and α.

4.3.3 Flow Assignment

In order to achieve the objective of this thesis, it is fundamental to implement a flow reassignment method that converges efficiently to DUE. First of all, it is important to note that not all flow reassignment algorithms achieve the expected equilibrium. Only the methods that implicitly assume that the network traffic conditions are predictable (preventive methods) are able to achieve this (Friesz et al. (1993)). While reactive algorithms would be those assuming that network conditions are not predictable and therefore users base their decisions en-route on real-time information about current traffic conditions. Due to the objective of this thesis, we propose a preventive flow reassignment algorithm which bases its decisions on its historical experience.

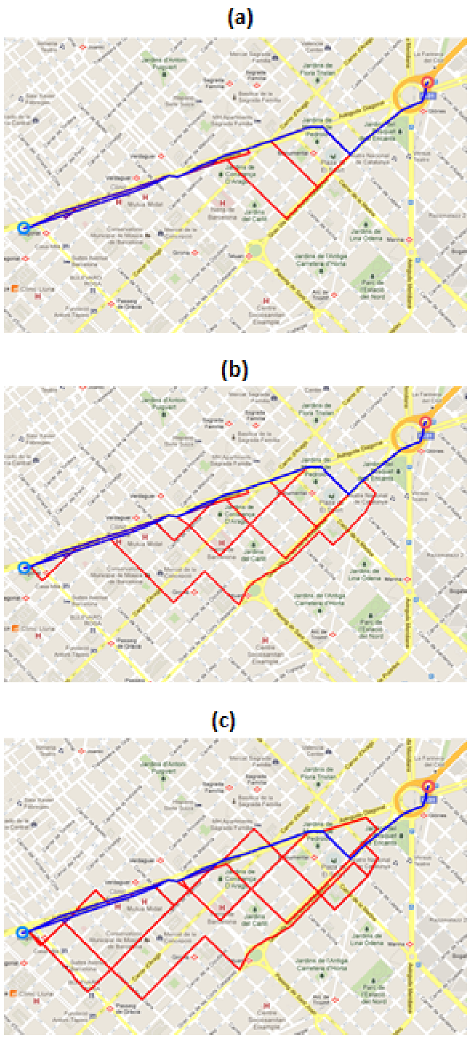


Figure 4.3.3: Example of the solutions achieved with the k-shortest path algorithm by Lim and Kim. (a) $Op = 0.95, \alpha = 2$. (b) $Op = 0.95, \alpha = 5$. (c) $Op = 0.8, \alpha = 5$.

Among the different preventive approaches existing in the literature, the iterative flow reassignment algorithm proposed here is based on a modification of the popular Method of Successive Averages (MSA). This is one of the most widely used methods for the path flow reassignment component in a DTA scheme, because it is one of the most simple and efficient. The method was used for the first time in transportation modeling by Powell and Sheffi (1982).

In the context of transportation, MSA consists of removing a fraction of the flow from each of the used paths in the current iteration and adding this amount to the flow of the current time-dependent shortest path for each OD pair and for each departure time interval.

The aim of the proposed modification is try to overcome the limitations observed during the state-of-the-art study of DTA with MSA. We focus our efforts on resolving two drawbacks:

- The expensive requirement of memory of a typical implementation of the MSA algorithm.
- The indiscriminate diversion flow from each used path to the new best path, which is common in the standard MSA.

In order to improve the currently available options for solving these two limitations, we develop a new MSA that combines some of the literature modifications with the addition of new ones. In order to overcome the first problem, we consider the general scheme formulated by Mahut et al. (2003), which is based on the idea of limiting the number of paths for each origin-destination pair for each departure time interval. To overcome the second limitation, we propose the use of a diversion factor in order to perform the reassignment by taking into account the cost of the alternative paths. This new factor is based on a logit distribution of demand flow according to actual travel costs of the alternative paths. The method considers the costs based on the actual link travel times obtained in the DNL of the previous iteration of the global procedure.

In summary, an adaptation of the MSA method that combines these two specific solutions has been developed. The former consists of limiting the maximum number of available paths for each OD pair for each departure time interval, in order to reduce the computational storage needed in the original MSA. The second uses a diversion factor based on actual travel times in the reassignment process, in order to reassign flow more realistically among the alternative paths.

Detailed specifications of the proposed flow reassignment method are the objective of Chapter 6.

4.3.4 Dynamic Network Loading

In the DTA model presented in this thesis, the DNL problem is solved using a new traffic simulation model based on a mesoscopic scheme that considers continuous-time link-based approach with complete demand discretization.

Considering disaggregated treatment of each individual vehicle allows the use of different vehicles classes in the problem, so the proposed network loading is a multiclass traffic simulation. This model considers as many vehicle classes as deemed necessary. Each vehicle class is characterized by two attributes: mean reaction time and the mean effective length of the vehicles in the class. In order to avoid that all vehicles belonging to a class are clones, the user can also define a deviation that will use the average values of a class to generate the reaction time and effective length of each vehicle in this class.

At the level of network links, our model only needs its length and the maximum allowed speed. The hypothesis is that, when a vehicle enters a link, it considers its speed as the maximum allowed speed in this link. Thus, the vehicle tries to achieve this speed and it will do so in the case of free flow conditions.

Link flow propagation is based on FIFO assumptions. Thus, if a vehicle enters a link later than another vehicle, and they come out in the same lane, they must leave it in the same entry order, i.e., one should not be able to overtake the other within this link.

Links spatially overlap in the physical representation of the network. Moreover, the proposed model simplifies the intersections by considering nodes as non-physical, abstract attributes that are responsible for managing the traffic entering and exiting the links. Thus, we cannot calculate the time that a vehicle needs to cross a node, because we have neither geometric nor speed information. Therefore, the node crossing time is imputed to the links among which the vehicle circulates. However, it is important to consider a fictitious node transfer time for slightly penalizing a vehicle's travel time when it passes from one link to a consecutive link. This would be done in order to reproduce the reduced speed of a vehicle making a turn, or of a driver being cautious when changing a link.

Each network node must have an associated turning set defined. We define a turn between two lanes of two links: the source link of the turn and the destination link of the turn. The former should be a link situated upstream from the node, and the second downstream. The turn manages the vehicle movement from its output lane origin link to the input lane of the destination link.

The node manages the vehicles that are waiting to cross, and it is responsible for informing the corresponding vehicle when the situation that blocked it changes: the traffic light phase that controls its turn changes to green, the vehicle that occupied the space of the node arrives to its next link, or the first cell of the input lane of its destination link is liberated.

Because one of the goals is to reproduce the traversal movements of vehicles when changing lanes, which considerably increases link congestion, this model allows longitudinal discretization of the links in lanes. Therefore, it is a multilane traffic simulation model.

The proposed model considers lanes in a simplified way: all lanes of a link have the same length (the link length) and the same maximum speed (the link speed). The number of lanes on each link of the network is also a required attribute.

The proposed DNL only reproduces temporal delays caused in traffic due to the traversal movements of vehicles performing a mandatory lane change within this link. This type of lane change is produced when the vehicle must perform this movement to continue on its assigned path without violating turn restrictions in the network nodes. In order to reproduce traffic patterns that occur when vehicles change lanes, it is necessary to consider the lane in which the vehicle entered the link and the lane in which it left. In this version of the DNL model, the proposed methodology for calculating these lanes takes into account traffic conditions in different link lanes at the moment. Moreover, the hypothesis is to minimize the number of lane changes.

The DNL problem is considered from a discrete demand point of view, so it is formulated by defining a function for each vehicle. From a specific link position, this provides the time at which the vehicle in question reaches that position $\left(t_{veh}^{a}\left(x\right)\right)$ and when it leaves $\left(t_{veh}^{d}\left(x\right)\right)$. These times are defined at the link level (not a lane level), although lane conflicts should be taken into account and lane attributes should be used to define this link level's functions.

The objective of the DNL is to calculate traffic variables for each link from the time dependent flow assignment at each of the network paths. This calculation pivots on the knowledge of the input and output times of the vehicle at each of the links of the network. So, the proposed model considers two positions (the initial and the final) at each lane of a link, in which the functions $t^a_{veh}(x)$ and $t^d_{veh}(x)$ should be evaluated.

From the above considerations, the development of the DNL model presented in Chapter 5 always follows the same methodology. Given a certain link of the network, the model calculates the following times for each vehicle:

- Arrival time of that vehicle to the initial link position.
- Departure time of that vehicle from the initial link position.
- Arrival time of that vehicle to the final link position.
- Departure time of that vehicle from the final link position.

The model is solved using an algorithm that is a discrete-event procedure. In a discrete event-based simulation, the different models that compose the procedure are designed in such a way that their outputs will remain valid for as long as the inputs to the calculation do not change. When a change in information occurs in the network, an event is generated at a specific point in time.

During the development of this mesoscopic network loading model, different events have been designed and implemented. Some of these events are related with the different times when the vehicle arrives to or departures from certain specific link positions (initial and end). Moreover, other events are directly related with the traffic simulation (start or end), and with the entry and exit of a vehicle in the network. Finally, some complementary events were needed to facilitate the implementation of the process and to improve the computation times. The implemented events are as listed:

- Start Simulation
- Enter Vehicle
- Vehicle Arrives at Link Origin
- Vehicle Leaves Link Origin
- Vehicle Check Departure From Link Origin

- Vehicle Arrives at Link End
- Vehicle Leaves Link End
- Vehicle Tries To Leave Link End
- Check Node
- Change Interval
- Exit Vehicle
- End Simulation

The process time for each event must be calculated at the same moment the event is generated. The calculation of this time explicitly follows the functions that define the traffic dynamics in the developed mesoscopic simulator.

As we mentioned before, details of how the proposed dynamic network model was developed are explained in Chapter 5.

4.3.5 Time-dependent Shortest Path Calculation

4.3.5.1 Time-Dependent Shortest Path Algorithms

Although the shortest path problem is one of the best studied combinatorial optimization problems in the literature, the dynamic graph variants received much less attention over the years. A graph is dynamic when some of the graph entities change with time. The most usual time-dependent changes are in the edge costs. Several combinatorial optimization problems in graphs may be defined in dynamic graphs, and this usually alters the problem's definition.

To our knowledge, the first citation to deal with the shortest path problem in dynamic graphs with time-dependent cost changes is Cooke and Halsey (1966). The authors proposed a recursive approach for finding the shortest path between two nodes of the network starting at certain time. It is shown that if travel costs take on integer positive values, then the procedure terminates with the shortest path from all nodes to a given destination.

Later, in Dreyfus (1969), Dijkstra's algorithm is extended to the dynamic case. In this work the FIFO property is not mentioned. This property is necessary to prove that Dijkstra's algorithm terminates with a correct shortest path tree on time-dependent networks. Let $c_{ij}(t)$ be the traveling time on the link (i, j) starting from i at time t. The FIFO property states that for each pair of time instants t_1, t_2 with $t_1 < t_2$: $\forall (i, j) \in A, c_{ij}(t_1) + t_1 \leq c_{ij}(t_2) + t_2$. Later, Kaufman and Smith (1993) formally proved that this Dijkstra generalization is valid only if the FIFO condition is satisfied.

Depending on how time is treated, dynamic shortest path problems can be divided into two types: discrete and continuous. In discrete dynamic networks, time is modeled as a set of integers. In continuous dynamic networks, time is treated as real numbers. For example, Orda and Rom (1990) proposed a continuous time-dependent one-to-all shortest path algorithm which allows delays at the links of the network. While, Ziliaskopoulos and Mahmassani (1993) introduced an all-to-one shortest path algorithm with time-dependent link costs using a discretization of the horizon of interest for small time intervals. It is important to note that these authors proposed solving the problem simultaneously for all values of the departure times.

Chabini (1998) proposed a Decreasing Order of Time Algorithm (DOT), based on the functional equations defined in Cooke and Halsey (1966), to compute the shortest paths from all origins to one destination for all departure times.

Dynamic problems involving other particularities have been investigated during the last 10 years. Dean (2004) discussed general properties and algorithms. Ahuja et al. (2003) studied the one-to-all shortest path algorithm for three problems: the minimum-time walk problem, the minimum excess time walk problem and the minim-cost walk problem. Dynamic problems involving turn penalties and prohibitions, multimodal networks and intersection movements were investigated by Wardell and Ziliaskopoulos (2000).

4.3.5.2 A Label Correcting Algorithm

The previously presented FIFO property is also called the non-overtaking property, because it basically says that if V_1 leaves i at time t_1 and V_2 at time t_2 $(t_1 < t_2)$, V_2 cannot arrive at j before V_1 using the link (i, j). It is important to note here that the developed DTA is based on a mesoscopic simulation model which does not allow overtaking, i.e, we propose a link flow propagation that satisfies FIFO conditions. Thus, the selection of a shortest path algorithm consistent with this is a logical option. Additionally, under the FIFO assumption, time-dependent shortest path problems exhibit many nice structural properties that enable the development of efficient polynomial-time solution algorithms. Thus, in this thesis we decided to deal with the time-dependent shortest path only for FIFO networks.

The used method is an adaptation of the algorithm proposed by Ziliaskopoulos and Mahmassani (1993). This may be considered a temporal extension of the label correcting algorithms used to compute static shortest path algorithms. The obtained results are the shortest paths from all origins to one destination for all departure times with time-dependent link costs using a discretization of the horizon of interest in small time intervals.

Let a graph $G = (N, A)$, with a finite nodes set N and a finite set of oriented links A. The studied time period T is discretized in a set of time intervals, $T = \{t_1 = (t_0, t_0 + \triangle t), t_2 = (t_0 + \triangle t, t_0 + 2\triangle t), .., t_S = (t_0 + (S-1)\triangle t, t_0 + S\triangle t)\}$, where t_0 is the earliest possible departure time from any origin node, $\triangle t$ is a small time interval during which no changes in traffic conditions or travel costs are perceived, and S is a constant such that the intervals from t_0 to $t_0 + S\triangle t$ cover the entire period of interest T. A cost function $c_{ij}(t)$ represents the time required at time t to reach node j from node i. This function is assumed to be a non-negative function defined in each discrete time interval. $\pi_i(t)$ is defined as the shortest traversal time of the directed path which connects node i to the destination d at time t. Also, $b_i(t)$ is the successor node of i at time t on the temporal shortest paths to destination d, and A_j^- is the set of arcs in the backward star of node j. The problem lies in determining, for each departure time t, the shortest paths from each node i to the destination d.

In the algorithm used, the node labels are maintained in a list Q. This algorithm performs the following steps:

```
Step 1. Initialization:
    For $i = 1 (i \neq d)$ to $n$ do:
        For $t = t_1$ to $t_S$ do:
            $\pi_i(t) = \infty$
    For $t = t_1$ to $t_S$ do:
        $\pi_d(t) = 0$
    $Q = \{d\}$
Step 2. Label update:
    **If** $Q = \emptyset$, go to Step 4;
    **Otherwise,** choose the first node $j$ of the queue $Q$, $Q = Q - \{j\}$;
    **For** $i = 1|(i,j) \in A_j^-$ to $n$ do:
        **For** $t = t_1$ to $t_S$ do:
            **if** $\pi_i(t) > c_{ij}(t) + \pi_j(t + c_{ij}(t))$, **then**
                $\pi_i(t) = c_{ij}(t) + \pi_j(t + c_{ij}(t))$
                **and** $b_i(t) = j$
        **If** at least one of the labels $\pi_i(t)$ was modified, then $Q = Q \cup \{i\}$
**Step** 3. **Repeat** Step 2.
**Step** 4. **Stop**.
```

At the end of the computations, the vector π_i associated to each node of the network contains the labels for each time period.

4.3.6 Convergence Criteria

In the presented DTA model two convergence criteria can be used to know if the procedure achieves the DUE or not:

- a relative gap ($Rgap$)
- a relative gap refined by interval ($Rgap_t$)

The relative gap for any feasible solution of DUE, proposed by Janson (1991), can be determined by the Equation 4.3.3.

$$RGap^k = \frac{\sum\limits_{o,d,t,p \in P_{odt}^{k-1}} f_{odtp}^{k-1} \left(c_{odtp}^k - c_{ody_{odt}t}^k \right)}{\sum\limits_{o,d,t} q_{odt} c_{ody_{odt}t}^k} \tag{4.3.3}$$

where:

f_{odpt}^{k-1}	Flow assigned on path p at $(k-1)\,th$ iteration.

c^k_{odpt}	Path cost based on travel time experienced on path p during the last DNL performed after the flow assignment of the $(k-1)\,th$ iteration.
$c^k_{ody_{odt}t}$	Path cost of the minimum path y_{odt} determined with a time-dependent shortest path algorithm based on travel time experienced on path p during the last DNL performed after the flow assignment of the $(k-1)\,th$ iteration.

Mahut et al. (2003) defined another gap measure inspired from Janson's proposal. This is a relative gap refined by departure time interval ($refined\,RGap$), and it is the difference between the total travel cost experienced and the total travel cost that would have been experienced if all vehicles had the travel cost (over each interval) equal to that of the current shortest path. The formulation is shown in Equation 4.3.4.

$$RGap^k_t = \frac{\sum\limits_{o,d,p\in P^{k-1}_{odt}} f^{k-1}_{odtp}\left(c^k_{odtp} - c^k_{ody_{odt}t}\right)}{\sum\limits_{o,d} q_{odt} c^k_{ody_{odt}t}} \tag{4.3.4}$$

A refined relative gap equal to zero for all the departure time intervals would indicate a perfect DUE flow.

4.4 Summary

In this chapter we have presented a new simulation-based DTA model through a variational inequalities approach. The proposed iterative process solves the dynamic extension of Wardrop's Principle: the DUE. The developed method has two fundamental components: the DNL and the flow reassignment method.

The DNL problem must be able to reproduce the network flow propagation, taking into account time and a variable traffic demand on each path of the network. In this chapter, we have introduced a multiclass multilane DNL model based on a mesoscopic scheme that considers a continuous-time link-based approach with complete demand discretization. Chapter 5 will explore this component more deeply.

Similarly, in the present chapter we have only introduced our flow reassignment component, but detailed specifications of the proposed flow reassignment method are the objective of Chapter 6.

Therefore, in this chapter we have explained in detail only the other components of the proposed DTA scheme, paying special attention to the shortest path algorithms used during the procedure: a K-static shortest path algorithm and a time-dependent shortest path algorithm. The proposed DTA scheme includes a time-dependent shortest path component in order to add new paths throughout the procedure when necessary. Moreover, the process can start with an initial path set determined through a static shortest path algorithm, which uses path costs in free-flow traffic conditions. After the first iteration, the path set can change if a new time dependent shortest path exists as a result of the new costs. Thus, the proposed path flow reassignment process must take into account this new path in the next path demand distribution.

In addition, we have specified the convergence criteria that the DTA method uses to detect the achieved convergence. The relative gap is one of the gap measures employed in this thesis for qualifying the reassignment procedure solution.

The main contributions of the developed DTA model will be presented in Chapter 7. In that chapter, a set of computational experiences will be carried out and the results that support these main contributions are displayed in detail.

Chapter 5

Dynamic Network Loading

5.1 Introduction

For years, models that reproduce the dynamic behavior of traffic have been studied from the perspective of traffic flow theory. They are applied to freeway models that are a chain of freeway sections which may have on-off ramps but no other extensions to the network. Simplified assumptions may be necessary to extend the models to other networks. Practical needs and natural curiosity have prompted research on dynamic models for traffic networks, especially urban networks. This has put the DTA models in the forefront. These models can be used to evaluate traffic flow both to simulate traffic management strategies and their impact on user behavior.

In many cases, these models were merely an extension of the concepts contained in static models. But the extension of static models to take into account dynamics is by no means straightforward, since dynamic supply modeling requires a completely new definition and formulation of the problem. To properly perform a DTA, it is necessary to be able to describe dynamics of traffic flow on the entire network. And this is the problem addressed by DNL: to develop a model that can reproduce network flow propagation while taking into account time and variable traffic demand on each path of the network.

This thesis proposes a DNL model to be embedded into a DTA framework. A DNL model is any model that maps time-dependent path demand flows onto time-dependent link flow.

As we mentioned before, DNL models are one of the most important components of DTA models, which in turn are a function of origin-destination time-dependent flows and time-dependent path flows. In general, we can say that DNL models depend on DTA models, since the path flows depend on the link costs, whilst the link costs are results of the path flows.

Section 5.2 summarizes the relevant literature regarding DNL models. We present the main link flow propagation schemes proposed in the literature: macroscopic, mesoscopic and microscopic models. We also analyze the existing solutions for the problems of node behavior and lane changes. Section 5.3 presents a new multiclass multilane DNL model based on a mesoscopic scheme that considers a continuous-time link-based approach with microscopic demand discretization. In Section 5.4, we summarize the model specifications and the design of the event-based algorithm to solve it. Finally, Section 5.5 describes the intensive computational experiences conducted in order to demonstrate the proposed multilane, multiclass model.

5.2 Literature Review

At justified in the previous classification of DTA approaches (see Chapter 2), the simulation-based approach is the best suited option for developing a DNL component that will be used in a DTA scheme.

All these DNL models have an intrinsic mechanism of flow propagation. This problem has been studied with a great number of different approaches that are sometimes referred to as:

- Macroscopic simulation traffic models
- Mesoscopic simulation traffic models
- Microscopic simulation traffic models

In order to define a DNL model, in addition to the above mentioned flow propagation models, we should address other issues. One of them is the management of traffic behavior at the nodes. If we consider two consecutive network links joined by a node, the intersection plays a key role in traffic flow propagation through links, since it is the downstream node of the first link and the upstream node of the second one. The DNL should manage both flows: those departing from the first link and those entering the second link.

Another important issue is the longitudinal discretization of links in lanes, and consequently lane changes. In a real network, we often need to use a specific lane in order to perform certain movements at intersections in orderto continue on a specific route (obligatory lane changes). The traversal movements produced by a vehicle changing its lanes can considerably increase link congestion, so we need to take this into account in our DNL model.

5.2.1 Flow Propagation Dynamics on the Links

In recent decades, interest in time-dependent traffic models has increased because of the need to better reproduce traffic dynamics in order to evaluate the continuously evolving ITS.

Despite the rich variety of dynamic network models, there are not many differences between the proposed approaches. Sometimes there are differences in temporal, spatial or demand discretization which can make it seem that there are differences between some simulation procedures based on the same model. For this reason, Astarita (2002) proposed a classification of the DNL mechanisms based on whether or not each of the proposed procedures discretizes time, space and demand.

The Astarita classification identifies the following models for the DNL procedure. These are intended particularly for link flow propagation:

- Microsimulation models
- Continuous in time link models
- Discrete in time link models
- Models following a packet approach
- Macroscopic simulation models

Astarita represented the proposed models in three-dimensional space, where the axes x, y and z are time, space and demand respectively. Thus, the value zero represents the continuous models which do not make any type of discretization;

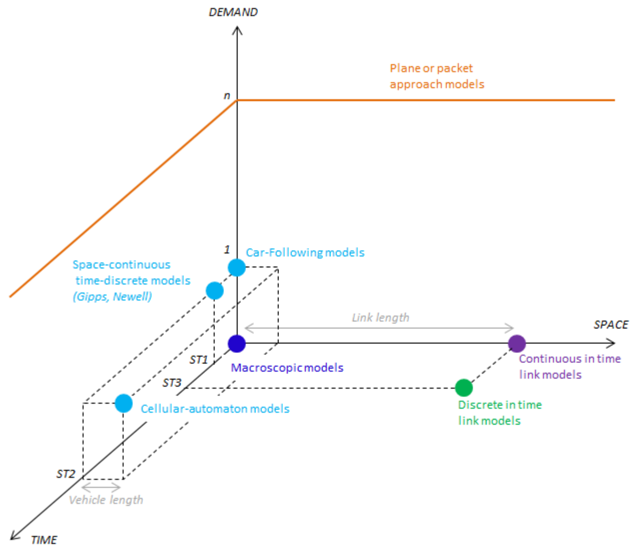

DEMAND
Plane or packet approach models
n
1
Car-Following models
Space-continuous time-discrete models (Gipps, Newell)
Link length
SPACE
ST1
ST3
Macroscopic models
Continuous in time link models
Discrete in time link models
Cellular-automaton models
ST2
Vehicle length
TIME

Figure 5.2.1: Our DNL classification inspired by Astarita's representation.

while advancing on one of the axes towards infinity represents a discretization of the variable of that axis (time, space or demand). For example, continuous-time models will be located on the space-demand plane $(y - z)$.

Figure 5.2.1 shows our DNL classification inspired by the representation proposed by Astarita.

In the literature, the DNL models based on the discretization of demand can be further broken down into a representation of space, which may be either continuous or discrete. Discrete-flow traffic models based on the cellular-automaton (CA) paradigm are defined in relatively fine discrete space (equal to the length of a single vehicle), while those based on car-following (CFM) logic use a continuous notion of space. Further, if we observe the time approach, the first models (CA) use a time-discrete approach, while the CFM use a time-continuous one. Also, it is important to note that some continuous-space models (like the proposals of Newell (2002) or Gipps (1981)) use a time-discrete approach, which differentiates them from CFM. All these models explicitly represent the individual lanes of each link in the network.

Car-following models are based on the individual movements of each of the vehicles which form demand, so these models are located at a point whose coordinate on the demand axis is equal to one. Following the same argument, these models treat time continuously, so its coordinate on the temporal axis must be 0. And finally space is continuously treated since the vehicles can physically take up any location in the network (i.e., inside links or nodes) during the loading process. For this reason, the spatial axis coordinate is zero. So, as we can see in Figure 5.2.1, the car-following models are located on the point $(0, 0, 1)$ at the presented classification inspired by Astarita's classification.

As we mention above, other space-continuous microsimulation models presented in the literature (Newell (2002), Gipps (1981), ...) consider a time-discrete approach. In this case, the models use a certain simulation step ($ST1$), so the coordinate at the temporal axis must be $ST1$, i.e., a positive number different from zero. They are located on the point $(ST1, 0, 1)$ at the presented classification.

Finally, microscopic models based on the cellular automaton paradigm use a discrete notion of space equal to the length of a vehicle. They also use a certain simulation step ($ST2$), so they are located on the point $(ST2, vehicle\, length, 1)$ at the presented classification based on Astarita.

Dynamic network models based on continuous time and traffic demand are generally well defined in a continuous space or they consider a link-discretization of space. The macrosimulation models belong to the first case (thus they are located at the point$(0, 0, 0)$), while in the second case the continuous-time link based models are located on the spatial axis at the point $(0, link\, length, 0)$.

In the group of macrosimulation-based models, the movements of the vehicles are modeled through functions of density and flow, based on fluid dynamics and respecting the law of flow conservation. The state variable is described for the entire length of each link in the network. The link flow and the speed are functions of this variable. Furthermore, macroscopic traffic models assume homogeneous traffic characteristics, thus many such models have been developed in order to be used in simulating freeways.

Some of the continuous-time link-based models have been solved numerically using time discretization ($ST3$). This has resulted in models directly related to the discrete-time link-based models, which were located in Astarita's classification at the point $(ST3, length\, link, 0)$.

Finally, packet approach methods are located in the space-time plane with a coordinate on the demand axis that depends on the size of the platoon. Within this group we can differentiate between an approach that considers the vehicle platoon concentrated at a single point in the network (on the links), and another approach that considers a continuous platoon, which means that vehicles are uniformly distributed in time and space.

Due to its limitations in realistic traffic modeling, each of the different presented approaches have varying degrees of practical application. The literature says that the most realistic models are those that use a car-following logic; these are the microscopic simulation models. These models are based on a detailed representation of the network. They use a relatively high number of traffic parameters, and also a great number of heuristics to solve different blocks of the propagation procedure. This makes large networks more difficult to calibrate, which is critical in the real-time applications of DNL and traffic assignment.

In the following, we review the above-mentioned traffic flow models, which are used to reproduce flow propagation along the links, i.e., the flow dynamics in the network links. We summarize the models by classifying them into three groups, depending on the flow demand discretization. So, from continuous consideration of demand to total discretization (single vehicles), we differentiate macroscopic, mesoscopic and microscopic models.

5.2.1.1 Macroscopic Traffic Flow Models

Macroscopic traffic flow models assume that the aggregate behavior of drivers depends on the traffic conditions in their environments. They deal with traffic flow in terms of aggregate variables. Usually, the models are derived from the analogy between vehicular and continuous media flows. In this section, we discuss continuum macroscopic flow models. Hereby, we will consider models describing the dynamics of macroscopic variables using partial differential equations.

Most macroscopic traffic flow models describe the dynamics of the spatial density $k = k(x,t)$, the speed $v = v(x,t)$ and the flow $q = q(x,t)$. Let (x,t) represents a location on a road segment, and a time instance respectively. Let density k(x,t) represents the number of vehicles present on a segment of road between (x, x+Δx) at time instance t, and flow q(x,t) represents the number of vehicles pass location x during time interval (t,t+Δt). Let velocity $v(x,t)$ the space-mean speed.. Some macroscopic traffic flow models also contain partial differential equations of velocity variance $\Theta = \Theta(x,t)$, or traffic pressure $P = P(x,t) = k\Theta$.

Let us assume that the dependent traffic flow variables are differentiable functions of time and space. Then, the relationship 5.2.1 holds exactly and, due to this, only two of the presented variables are independent.

$$q = k \cdot v \tag{5.2.1}$$

Thus, we may write the fundamental conservation law of fluid dynamics as Equation 5.2.2 shows.

$$\partial_t k + \partial_x q = 0 \tag{5.2.2}$$

Equations 5.2.1 and 5.2.2 constitute a system of two independent equations and three unknown variables. Consequently, to get a complete description of traffic dynamics, a third independent model equation is needed. Later in this section, several model specifications are considered.

In 1955, Lighthill and Whitham provided a pioneering first-order macroscopic dynamic model of traffic flow (Lighthill and Whitham (1955)). Their approach assumed that the expected velocity v can be described as a function of the density $k : v(x,t) = v(k(x,t))$. The resulting non-linear first-order partial differential Equation 5.2.3 was introduced.

$$\partial_t k + \partial_x (k \cdot v(k)) = 0 \tag{5.2.3}$$

Different variants of the model can be characterized by the relationship $v = v(k)$. The original proposition of the LWR model used a well-known relationship between speed and density, due to Greenshields et al. (1935). This is a characterized by the Equation 5.2.4.

$$v = v_0 \left(1 - \frac{k}{k_j}\right) \tag{5.2.4}$$

where:

v_0 is the free flow speed.

k_j is the jam flow density (the density of the traffic system when the velocity is zero).

In this case, the complete model is given by Equation 5.2.5.

$$\partial_t k + (v_0) \cdot \partial_x k - \left(\frac{v_0}{k_j}\right) \cdot \partial_x k^2 = 0 \tag{5.2.5}$$

Another relationship that has been used in conjunction with the LWR theory is the following piecewise linear relationship between flow and density, shown in Equation 5.2.6.

$$q = \begin{cases} k \cdot v_0 & for\, k \leq k^* \\ \alpha \cdot (k_{jam} - k) & for\, k \geq k^* \end{cases} \tag{5.2.6}$$

where $\alpha = \frac{k^* \cdot v_0}{k_j - k^*}$ and k^* is called the critical density and it corresponds to the maximum flow.

In 1994, Daganzo proposed another simplified model, known as the Cell Transmission Model, based on a three-linear-segment flow density relationship (Daganzo (1994)). The third segment defines a maximum flow constraint, somewhere below the maximum flow defined for the two-segment model. This model requires the discretization of space into cells. It can be applied to general networks with the addition of appropriated node models.

Since the assumption is that speed is only a function of density, the LWR theory predicts that a vehicle reacts instantaneously to changes in density. This is equivalent to saying that acceleration is unbounded, i.e., these changes in velocity may be instantaneous. This is considered one of the main deficiencies of this model.

Second-order models were developed as an attempt to improve the accuracy of first-order models and to circumvent their qualitative deficiencies, although at the expense of higher complexity that renders analytical solutions virtually impossible. The most popular second-order model was suggested by Payne (1971). Payne's model considers car-following and takes into account the reaction time of drivers (T), which leads to a dynamic mean speed equation instead of the static one used in first-order models. The proposed model overcomes the basic problem of instantaneous changes in the velocity of the LWR theory mentioned above. Equation 5.2.7 is Payne's momentum equation.

$$\partial_t v + v \cdot \partial_x v = \frac{1}{T}(v(k) - v) - \left(\frac{c_o^2}{k}\right) \partial_x k \tag{5.2.7}$$

where:

c_o is the wave speed of the linearized flow density relationship.

The model contains convection, relaxation and anticipation terms:

- Convection $(v \cdot \partial_x v)$ describes changes in the mean velocity due to inflowing and outflowing vehicles.
- Relaxation $\left(\frac{1}{T}(v(k) - v)\right)$ describes the tendency of traffic flow to relax to an equilibrium velocity. It limits how quickly the flow may adapt to changes in density.
- Anticipation $\left(\left(\frac{c_o^2}{k}\right) \partial_x k\right)$ describes the anticipation of the driver in spatially changing traffic conditions downstream.

A more general model form of Equation 5.2.7 is given by Equation 5.2.8.

$$\partial_t v + v \cdot \partial_x v = \frac{1}{T}(v(k) - v) - \left(\frac{1}{k}\right) \partial_x P + \left(\frac{\eta}{k}\right) \partial_x^2 v \qquad (5.2.8)$$

where η is the kinematic traffic viscosity. The total time derivative describes the rate of velocity changes experienced by a moving observer who observes the traffic flow while moving along the stream at the same velocity v. Note that for the Payne model 5.2.7, we have $P = k \cdot c_0^2$ and $\eta = 0$. Introducing traffic viscosity results in approximate smooth solutions of Payne's model. In addition, the numerical treatment of these higher order models is simplified.

Kühne and Rödiger (1991) and Kerner and Konhäuser (1994) chose $P = c_0^2$ and $\eta = \eta_0$. The assumption of constant velocity variance is not realistic. Rather, in equilibrium, velocity variance decreases with increasing traffic density, leading to Equation 5.2.9.

$$\partial_t v + v \cdot \partial_x v = \frac{1}{T}(v(k) - v) - \left(\frac{c_o^2}{k}\right) \partial_x k + \left(\frac{\eta_0}{k}\right) \partial_x^2 v \qquad (5.2.9)$$

In 1996, Helbing extended the Payne models by introducing an additional partial differential equation for velocity variance Θ (Helbing (1996)). His macroscopic model is derive from gas-kinetic equations and consists of the conservation of vehicles Equation 5.2.2, velocity dynamics and the Equation 5.2.10 describing the dynamics of the variance Θ.

$$\partial_t \Theta + v \cdot \partial_x \Theta = -2\left(\frac{P}{k}\right) \cdot \partial_x v + \left(\frac{2}{T}\right)(\Theta^e - \Theta) - \left(\frac{1}{k}\right) \cdot \partial_x J \qquad (5.2.10)$$

where the flux of velocity variance $J = J(x,t) = k(x,t)\Gamma(x,t)$ is defined by the product of the density and the skewness of the velocity distribution. Rather than being experimentally determined, the equilibrium velocity v^e and variance Θ^e are determined by considering the interaction process between vehicles in the stream. The resulting expressions are functions of density$k$, velocity v and velocity variance Θ, namely:

$$v^e(k,v,\Theta) = v_0 - T(1-p(k))P \quad and \quad \Theta^e(k,v,\Theta) = C - T(1-p(r))J \quad (5.2.11)$$

where $p(k)$ denotes the immediate overtaking probability while C is the covariance between the velocity and the desired velocity. The model equations are 'closed' by specifying expressions for p, C and J. In 1996, Helbing also proposed techniques for incorporating the fact that vehicles occupy a non null amount of roadway space.

5.2.1.2 Mesoscopic Traffic Flow Models

Mesoscopic models combine the properties of both microscopic and macroscopic simulation models that describe traffic flow in medium detail. Vehicle and driver behavior are not described individually, but rather in more aggregate terms. However, behavior rules are described at an individual level. Different models describe velocity distributions at specific locations and instants of time. The dynamics of these distributions are generally governed by various processes describing the individual behavior of drivers. Three well known examples of mesoscopic flow models are present in the literature: headway distribution models, cluster models and gas-kinetic continuum models.

Headway Distribution Models
Headway distribution models are mesoscopic in the sense that they describe the distribution of the headways of individual vehicles while neither explicitly considering nor tracing each vehicle separately. A time headway is defined by the difference in passage times of two successive vehicles. In general, it

is assumed that these time-headways are identically distributed, independent, random variates.

In 1968, Buckley presented a headway distribution model based on a semi-Poisson distribution (Buckley (1968)). Another example of headway distribution models was presented by Branston (1976), the generalized queuing model.

One variation of the headway distribution model is the mixed headway distribution model that distinguishes between leading and following vehicles: the time headways of leading and following drivers are taken from different probability distributions.

These mesoscopic models have been criticized for neglecting the role of traffic dynamics. Also, these models assume that all vehicles are essentially the same. That is, the probability distribution functions are independent of traveler type, vehicle type, travel purpose, etc. In order to remedy this, in 1998 Hoogendoorn and Bovy developed a headway distribution model for, respectively, multiclass traffic flow and multiclass multilane traffic flow (Hoogendoorn and Bovy (1998)). Using a new estimation technique, they analyzed multiclass traffic in the Netherlands on both two-lane rural roads as well as two-lane motorways.

Cluster Models

A cluster is a group of vehicles that share a specific property. Cluster models are characterized by the central role of these clusters of vehicles. In the literature, these models consider some different aspects of clusters. Usually the number of vehicles in a cluster (size of the cluster) and the speed of a cluster are of dominating importance. The size of a cluster is not static, the number of vehicles in it can grow and decay during the process. However, the within cluster traffic condition (as for example headways or speed differences) are usually not considered explicitly, i.e., the clusters are considered homogeneous in this sense.

Gas-kinetic Continuum Models

Gas-kinetic traffic flow models describe the dynamics of the speed distribution functions of vehicles in the traffic flow instead of describing the traffic dynamics of individual vehicles. These models are governed by terms that reflect various dynamic processes describing individual driver behavior, such as acceleration, interaction between vehicles and lane-changing.

In 1971, Prigogine and Herman were the first to describe the dynamics of traffic flow by using a gas-kinetic approach (Herman et al. (1971)). Their model describes the dynamics of the reduced phase-space density (PSD) $\tilde{\rho}(x, v, t)$. PSD is a mesoscopic generalization of macroscopic traffic density $k(x, t)$, and it reflects the speed distribution function of a single vehicle. Prigogine and Herman assumed that dynamic changes in the reduced PSD are caused by convection, acceleration towards the desired speed, and deceleration due to interaction between drivers, which yields the following partial differential Equation 5.2.12.

$$\partial_t \tilde{\rho} + v \partial_x \tilde{\rho} = \left(\partial_t \tilde{p}\right)_{acc} + \left(\partial_t \tilde{p}\right)_{int} \tag{5.2.12}$$

where

$\left(\partial_t \tilde{p}\right)_{acc}$ reflects changes caused by acceleration towards the desired speed,

$\left(\partial_t \tilde{p}\right)_{int}$ denotes changes caused by interactions between vehicles, i.e., deceleration due to fast vehicles catching up with slower vehicles.

In Equation 5.2.12, the acceleration term $\left(\partial_t \tilde{p}\right)_{acc}$is defined.

$$\left(\partial_t \tilde{\rho}\right)_{acc} = -\partial_v \left(\tilde{\rho} \cdot \left(\tilde{V}^0 \left(v \left| x, t\right.\right) - v\right) / \tau\right)$$

where τ denotes the acceleration time and $\tilde{V}^0 \left(v \left| x, t\right.\right)$ is the desired speed distribution.

The interaction term$\left(\partial_t \tilde{p}\right)_{int}$ in Eq. 5.2.12 equals the so-called collision equation.

$$\left(\partial_t \tilde{p}\right)_{int} = \left(1 - \pi\right) \tilde{\rho}\left(x, v, t\right) \int \left(w - v\right) \tilde{p}\left(x, w, t\right) dw \tag{5.2.13}$$

The interaction term of the original work of Prigogine and Herman was criticized and improved on by Paveri-Fontana (1975)). He considered a hypothetical scenario where a free flow vehicle catches up with a slow moving queue. The author showed that the Prigogine and Herman formalism infers that when a vehicle passes a platoon, it passes each single car in the queue independently, while a real-life situation falls between these two extremes.

Paveri-Fontana also showed that the term reflecting the acceleration process yields a desired velocity distribution that is dependent on the local number of vehicles. This is in contradiction to the well-accepted hypothesis that the personality of drivers is inconsequential to changing traffic conditions. To remedy this deficiency, Paveri-Fontana considered a generalization of the reduced PSD $\rho(x, v, v^0, t)$ with an independent variable describing the desired speed v^0, which considers the joint distribution of speed and desired speed rather than describing the dynamics of speed distributions. He proposes an equation similar to Equation 5.2.12.

$$(\partial_t \rho)_{acc} = -\partial_v \left(\rho \cdot \left(v^0 - v\right)/\tau\right)$$

and:

$$\begin{aligned}(\partial_t \rho)_{int} = &-\rho\left(x, v, v^0, t\right) \int_{w<v} (1-\pi) \left|w - v\right| \tilde{\rho}\left(x, w, t\right) dw + \\ &+\tilde{\rho}\left(x, v, t\right) \int_{w>v} (1-\pi) \left|w - v\right| \rho\left(x, w, v^0, t\right) dw\end{aligned} \tag{5.2.14}$$

where the reduced PSD equals:

$$\tilde{\rho}\left(x, v, t\right) \overset{def}{=} \int \rho\left(x, v, v^0, t\right) dv^0 \tag{5.2.15}$$

Gas-kinetic traffic flow models have been criticized by Newell (1995), among others. One point of concern was the inability to adequately describe traffic flow operations under non-free-flow traffic conditions. Moreover, compared to macroscopic traffic flow models, mesoscopic gas kinetic flow models have a relatively large number of unknown parameters and model relations (e.g., velocity distribution functions) that need to be estimated from traffic observations. In addition, the relatively large number of independent variables (i.e., time, location, velocity, and desired velocity) necessitates numerical solution approaches using a four-dimensional lattice, which hampers application of simple numerical approximation schemes and increases computational complexity. However, Helbing (1997), Hoogendoorn and Bovy (1998), Hoogendoorn and Bovy (2001a), and Hoogendoorn (1999) have renewed the interest in gas-kinetic models by applying them to continuum macroscopic traffic models.

Helbing's approach (Helbing (1997)) is similar to the approach of Paveri-Fontana (1975), although lane-changing is explicitly considered using the dynamics of the Multiclass Phase-Space-Density (MUC-PSD) $\rho_j\left(x, v, v^0, t\right)$, where j indicates the corresponding roadway lane. In 1998, Hoogendoorn and Bovy presented another extension of a gas-kinetic model for multiclass traffic flow using Multilane

PSD $\rho_u\left(x, v, v^0, t\right)$, where u indicates the user class u (cars, buses, trucks, etc.) (Hoogendoorn and Bovy (2001b)).

In 1999, Hoogendoorn derived gas-kinetic models for both multilane and multiclass traffic flow, considering a platoon-based description of the traffic stream. He shows that this approach allows for a realistic macroscopic multilane and multiclass traffic flow model, characterized by asymmetric interaction between the fast and slow vehicle types among roadway lanes. Moreover, the author eliminates the inability of gas-kinetic models to describe non-free-flow traffic by considering traffic as a collection of vehicle platoons, thereby describing the correlation between positions and vehicle velocities. Doing so remedies the flawed vehicular chaos assumption (the assumption that vehicles can be described by a collection of independently moving particles). Additionally, the finite space requirements of the vehicles are explicitly accounted for in his approach. Both Helbing (1997) and Hoogendoorn (1999) established that their models yield expressions for equilibrium velocity and velocity variance, reflecting the expected velocity and the velocity variance of the vehicular flow in a stationary situation, respectively. Hoogendoorn (1999) also shows how the gas-kinetic foundation yields the equilibrium distribution of fast and slow vehicles across the roadway lanes.

In 2001, Hoogendoorn and Bovy consolidated all the previous mentioned gas-kinetic models (Hoogendoorn and Bovy (2001b)). They presented a generic continuum modeling approach for the description of flow operations for a general class of traffic system that models traffic flow independently of the type of traffic (vehicular traffic motorways or rural roads, pedestrian flows). They presented a generalized Phase-Space Density ($g-PSD$). This density concept is a generalization of traffic density with respect to both discrete attributes (such as user-class, roadway lane, destination) and continuous attributes (such as speed, and desired speed). In their approach, the authors also treated the non-continuum processes which describe non-smooth changes in the $g-PSD$, while the continuum processes described smooth changes. Finally, Hoogendoorn and Bovy showed how previous simplified models can be derived from this generic gas-kinetic equations.

5.2.1.3 Microscopic Traffic Flow Models

Microscopic traffic models describe the movement of each vehicle, i.e., they model the actions (such as acceleration, deceleration or lane changes) of each driver as responses to the surrounding traffic. These models are especially suited to the study of heterogeneous traffic streams consisting of different and individual types of driver-vehicle units. This results in the appropriate aggregation of the individual trajectories of all vehicles and, consequently, any macroscopic information.

In order to describe the complete task of driving, microscopic models generally comprise:

- An acceleration strategy towards a desired velocity in the free-flow regime.
- A braking strategy for approaching other vehicles or obstacles.
- A car-driving strategy for maintaining a safe distance when another vehicle is driving behind.

Microscopic traffic models typically assume that human drivers react to stimulus from neighboring vehicles, with the dominant influence caused by directly leading vehicles. In addition, the individual behavior of drivers is characterized in terms of model parameters such as a desired velocity, a preferred gap to the vehicle ahead while following, a bound for the acceleration, a comfortable deceleration, a reaction time, and others. Furthermore, human drivers often exhibit more complex driving patterns such as different kinds of anticipation, limited attention or other processes of adaptation that could be taken into account as well.

Specifically, we can distinguish the following subclasses of microscopic traffic models:

- Time-continuous models which are formulated as ordinary differential equations and, consequently, space and time are treated as continuous variables. Car-following models are the most prominent examples of this approach. In general, these models are deterministic but stochasticity can be added in a natural way.
- Cellular automaton models that use integer variables to describe the dynamic state of the system. In these models time is discretized and the links are divided into cells which can be either occupied by a vehicle or empty. Although these models lack the accuracy of the previous ones, they are able to reproduce some traffic phenomena. Due to their simplicity, they can be implemented very efficiently and are suited to reproduce traffic in large scale networks.

- Iterated coupled maps are between cellular automaton and time-continuous models. In this model class, the update time is considered as an explicit model parameter rather than an auxiliary parameter needed for numerical integration. Consequently, time is discretized while the spatial coordinate is still considered continuous. Popular examples are the Gipps model and the Newell model. However, these models are typically associated with car-following models as well.

In this section we only examine the main models required to justify our proposal for a traffic flow link propagation model.

Car-following Models

Car-following behavior describes how a pair of vehicles interact with each other, more specifically: the way in which the space-time trajectory of one vehicle depends on that of the vehicle in front of it.

In recent years, the importance of such models has increased further, with 'normative' behavioral models forming the basis of the functional definitions of advanced vehicle control and safety systems. Other systems, such as autonomous cruise control, seek to replicate human driving behavior through driver misperception and reaction time. A detailed understanding of this key process is now becoming increasingly important as opportunities for using new techniques and technologies become available.

The study of car-following models has been extensive, with conceptual bases supported by empirical data, but generally limited by the lack of time-series following behavior. In many cases, model stability and the implication of each of the relationships to macroscopic flow characteristics have been investigated. It is highly tempting to try and increase the realism of a chosen model by attempting to incorporate motivational or attitudinal factors that may be able to explain the difference between drivers, although there is little evidence to relate such features to observable dynamic behavior.

In an evaluation of microscopic car-following behavior, Panwai and Dia (2005) described an incentive for traffic engineers and psychologist to work together on a car-following model and to open a new branch of research on it. They enumerated the factors that have been found to influence car-following behavior, some of them psychological factors. Ranney (1999) classified these factors into these two categories:

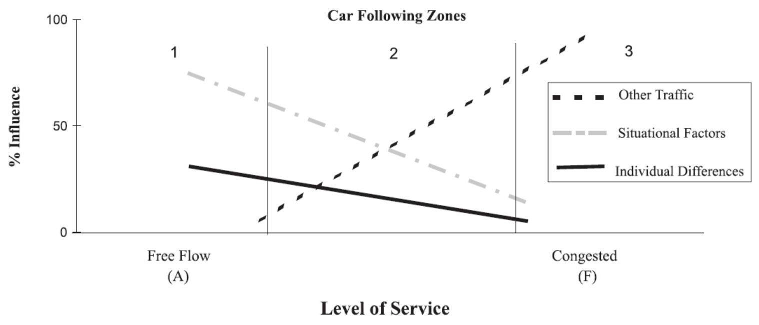


Figure 5.2.2: Relative contribution of different factors in influencing car-following behaviour. *Source: Ranney (1999).*

- Individual differences: age, gender, risk-taking propensity, driving skills, vehicle size and vehicle performance.
- Situational factors:
  - Environment: time of day, day of week, weather and road conditions.
  - Individual: situations of distraction, being in a hurry, trip purpose, driving duration, and impairment due to alcohol, drugs, stress and fatigue.

In Figure 5.2.2 we show a conceptual model of the relative contribution of different categories of factors in influencing car-following behavior presented by Ranney (1999).

According to this model, car-following occurs primarily at intermediate levels of service, represented by car-following zone 2 in Figure 5.2.2. Under free-flow conditions (zone 1), car-following may occur, but only if the driver chooses to drive closely to another vehicle. Under congested conditions (zone 3), drivers have little choice about following the lead vehicle closely. The model also indicates that within zone 2, car-following is determined by a combination of the constraints imposed by other vehicles, individual factors and situational factors, and that the relative contribution of these factors differs as a function of traffic congestion.

Car- following model classifications

Perhaps the earliest car-following model was proposed by Herrey and Herrey (1945), who postulated that a driver would maintain a minimum "safe driving distance" that included the stopping distance, thus obtaining a spacing parameter which was a quadratic function of velocity. Their paper is significant in that it is one of the very few papers (along with Newell (2002)) that describe car-following in terms of vehicle trajectories rather than just velocities, spacings, etc. However, in the literature we can find that the car-following models proposed by Reuschel (1950) and Pipes (1953), independently, are considered the pioneers of microscopic traffic models. Both models were inspired by driving rules, including considerations that the distance between two consecutive vehicles is proportional to the speed.

Since this first car-following experiment, many observations have been made in an effort to understand car-following behavior for the single-lane follow-the-leader situation.

A study by Brackstone and McDonald (1999) classified car-following models into five groups, as follows:

- Gazis-Herman-Rothery (GHR) model
- Safety distance or collision avoidance models
- Linear models
- Psychophysical or action point models
- Fuzzy logic-based models

Other car-following models not covered in the abovementioned review but present in the literature are: desired spacing models , models based on trajectories, and the series of intelligent/human driver models.

This thesis does not attempt to describe all the car-following models presented in the literature. Instead, this thesis gives a short-term description of the models that are clearly influential in our proposal of link flow propagation, which are: collision avoidance models, desired spacing models and Newell's model based on trajectories (Newell (2002)).

Collision Avoidance Model

Also known as the safety distance model. The hypothesis of this class of model is that a driver will place himself at a certain distance from the lead vehicle, such that in the event of an emergency stop by the leader, the follower will come to rest without striking the lead vehicle.

This model was first presented by Pipes (1953), who characterized the motion of vehicles in the traffic stream as following rules, based on the concept of distance headway, suggested in the California Motor Vehicle Code, namely: "A good rule for following another vehicle at a safe distance is to allow yourself at least the length of a car between your vehicle and the vehicle ahead for every ten miles per hour of speed at which you are traveling". Pipes' car-following theory leads to a minimum safe distance headway that increases linearly with speed.

Kometani and Sasaki (1959) also proposed that the base relationship seeks to specify a safe following distance, within which a collision would be unavoidable if the driver of the vehicle in front were to act unpredictably.

The major development of this model was made by Gipps (1981). He constructed a new car-following model with the following properties:

- The model should mimic the behavior of real traffic.
- The parameters in the model should correspond to obvious characteristics of drivers and vehicles, so that most can be assigned values without resorting to elaborate calibration procedures.
- The model should be well behaved when the interval between successive recalculations of speed and position is the same as the reaction time.

The model is derived by setting limits on the performance of driver and vehicle and using these limits to calculate a safe speed with respect to the preceding vehicle. It is assumed that the driver of the following vehicle selects his speed to ensure that he can bring his vehicle to a safe stop should the vehicle ahead come to a sudden stop.

The car-following paradigm investigated by Gipps was based on a simple rule: the follower attempts to maximize his instantaneous speed, subject to two constraints: acceleration and safety.

The acceleration constraint is a statement of the physical limitations of a vehicle in terms of speed and acceleration, as well as of a driver's desire for comfort.

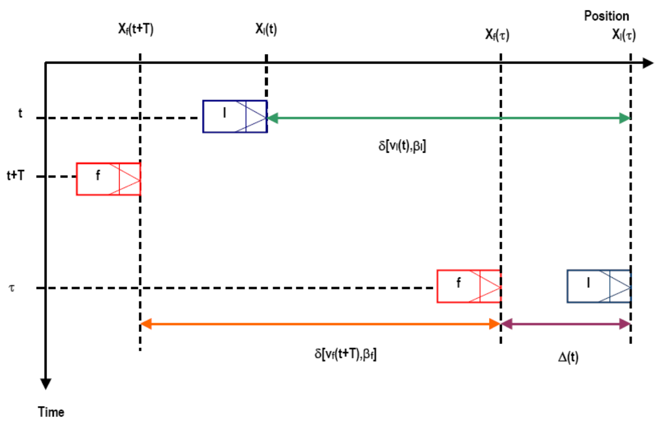


Figure 5.2.3: Safe deceleration to stop diagram. *Source: Mahut (1999) (Adapted from Gerlough and Huber (1976)).*

Essentially, it describes the trajectory of a vehicle that is free to accelerate to its maximum desired speed in the absence of downstream vehicles. On the other hand, the safety constraint is a statement of how the trajectory of a vehicle is affected by the next downstream vehicle.

Mahut (1999) presented a generalization of the car-following model originally proposed by Gipps (1981). It generalizes the safe-stopping (or safety) constraint. Typically, in the formulation of a safety constraint, all variables are referenced to the same time t by estimating the follower's state vector at time $t+reactionTime$ as a function of his state vector (position, velocity and acceleration) at time t. The approach by Mahut allowed the state vectors of the leader and the follower to be referenced to different points in time, i.e., to t and $t + reactionTime$, respectively.

Mahut proposed in Figure 5.2.3 his safe deceleration to stop diagram, where he showed how the modeling hypothesis of safety distance works:

where:

f is the follower vehicle.

l is the leader vehicle.

T is the response time.

$\triangle(t)$ is the minimum distance between the front bumpers of two vehicles at zero speed, referred to as the effective length.

$\delta\left[v_f\left(t+T\right),\beta_f\right]$ is the deceleration distance function: the distance required to decelerate from speed v to a stop with deceleration parameter β.

x_f is the position of the follower.

x_l is the position of the leader.

v_f is the speed of the follower.

v_l is the speed of the leader.

β_f is the desired follower deceleration

β_l is the anticipated deceleration parameter (how the follower anticipates the leader's deceleration).

Hence, to avoid a collision, the trajectory of the follower must satisfy the following inequality constraint, i.e., the safety constraint shown in Equation 5.2.16.

$$x_f\left(t+T\right)+\delta\left[v_f\left(t+T\right),\beta_f\right]+\triangle\left(t\right)\leq x_l\left(t\right)+\delta\left[v_l\left(t\right),\beta_l\right] \qquad (5.2.16)$$

Assuming that the deceleration distance functions are, respectively:

$$\delta\left[v_f\left(t+T\right),b_f\right]=\frac{\left[v_f\left(t+T\right)\right]^2}{2\beta_f},\ \delta\left[v_l\left(t\right),\beta_l\right]=\frac{\left[v_l\left(t\right)\right]^2}{2\beta_l} \qquad (5.2.17)$$

So, the safety constraint becomes the Equation 5.2.18 .

$$v_f\left(t+T\right)\leq\sqrt{2\beta_f\left(x_l\left(t\right)+\frac{\left[v_l\left(t\right)\right]^2}{2\beta_l}-x_f\left(t+T\right)-\triangle\left(t\right)\right)} \qquad (5.2.18)$$

where the absolute values of β_l and β_f are considered.

In addition to the usual quadratic expression (proposed by Gipps) for the deceleration trajectory, a second deceleration function was proposed by Mahut, which yields a logarithmic relationship between speed and stopping distance.

In 2001, Wilson examined Gipps's model and performed a detailed mathematical analysis of the uniform flow solutions and their stability (Wilson (2001)).

In particular, he showed that the uniform flow may only become unstable in unrealistic parameter regimes. It follows that Gipps's model is not able to replicate stop-and-go waves. Further work by Rahme et al. (2002) simulated minor adaptations of Gipps's model in an attempt to produce stop-and-go waves, but found no satisfactory parameter regimes.

Desired spacing models

The desired spacing models were proposed by Parker (1996) and Hidas (1998). These models are based on a desired spacing criterion which is assumed to be a linear function of the speed:

$$D_i = \alpha \cdot v_i + \beta \tag{5.2.19}$$

where:

D_i is the desired spacing.

v_i is the speed of vehicle i.

α, β are constants.

The models are based on the premise that desired spacing is an individual driver characteristic and that drivers have different desired spacing criteria in acceleration and deceleration.

These models eliminate the problems associated with the reaction times used in other models because they describe car-following based on desired spacing between vehicles without attempting to explain the behavioral aspects of car-following.

Hidas developed this car-following model specifically into a microscopic simulation of congested urban traffic flow conditions. It was implemented in ARTEMiS (Analysis of Road Traffic and Evaluation by Micro-Simulation), a traffic simulator developed by the author at the University of New South Wales since 1995. It has been validated for urban interrupted flow conditions, which is one of the reasons why this model is interesting for our research.

In 2005, Hidas modeled vehicle interactions in microscopic simulations of merging and weaving (Hidas (2005)). He combined his desired spacing, car-following model with a new lane-changing model. Thus, Hidas eliminated the shortcomings of the previous proposed models by implementing new concepts based on data collected from video which recorded lane-change observations.

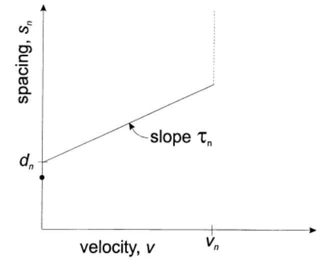


Figure 5.2.4: Relation between spacing and velocity for a single vehicle. *Source: Newell (2002).*

Newell's model

In 2002, a very simple car-following rule was proposed by Newell, wherein if an nth vehicle is following an $(n-1)$th vehicle on a homogeneous highway, the time-space trajectory of the nth vehicle is essentially the same as the $(n-1)$th vehicle except for a translation in space and in time (Newell (2002)).

The spacing $x_{n-1}(t) - x_n(t)$ between vehicles n and $(n-1)$ at time t could change with time, but if the highway is homogeneous, it should remain nearly constant at some value s_n . The s_n could be different for different vehicles and it will also depend on the constant average velocity v.

Newell idealized some empirical relationship between the v (at least over some range of v) and s_n with a linear relation, as illustrated in Figure 5.2.4.

He supposed that the $(n-1)$th vehicle travels at a nearly constant v for some period of time, but then changes velocity and eventually acquires a nearly constant velocity v' . The new trajectory may look like the path shown with a broken line in Figure 5.2.5, but it supposes that the two constant velocity segments are extrapolated until they intersect the solid lines, and that the actual trajectory stays close to the solid lines. The nth vehicle will do likewise, with the junction of the solid lines for the nth vehicle displaced relative to the $(n-1)$th vehicle by a space displacement d_n and time displacement τ_n.

So the piecewise linear trajectory $x_n(t)$ would be simply a translation of the piecewise linear $x_{n-1}(t)$ for a distance d_n and a time τ_n, i.e.,

$$x_n(t+\tau) = x_{n-1}(t) - d_n \qquad (5.2.20)$$

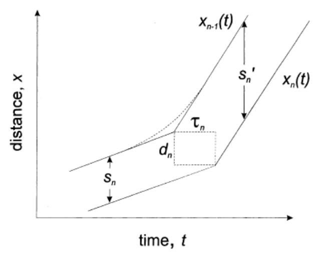


Figure 5.2.5: Piecewise linear approximation to vehicle trajectories. *Source: Newell (2002).*

where $d_n = s_n - v \cdot \tau_n$.

The details of how a following vehicle manages to stay close to a previous relationship are not important, provided that the driver is capable of doing so, which should not be difficult, because if the lead vehicle should increase (decrease) its velocity, the following vehicle does not need to respond immediately. It can wait until the spacing has increased (decreased) to a value comparable to its value for the new velocity. Indeed, the slope τ_n in Figure 5.2.5 derives from what the driver considers to be a safe driving distance, which implies an ability for him to respond comfortably to anything the $(n-1)$th vehicle may do.

Specifically, our interest was to analyze the algorithm proposed by Yeo et al. (2008) inside the Next Generation Simulation (NGSIM) Project, sponsored by the U.S. Federal Highway Administration. This car-following model is a new behavioral algorithm for the over-saturated freeway flow which used Newell's model as a starting point.

This algorithm combines a car-following model with a lane-changing algorithm. The car-following model was developed to simulate over-saturated flow, and they introduce some additional algorithms to control lane changes and their effects on the car-following models. This modeling framework allowed simulation of all possible driving conditions.

The different components of the proposed algorithm and the relationships between them are summarized in the diagram shown in Figure 5.2.6:

In order to simplify, we only discuss the basic car-following model used for Skabardonis in his complete algorithm. This car-following model is a simplified

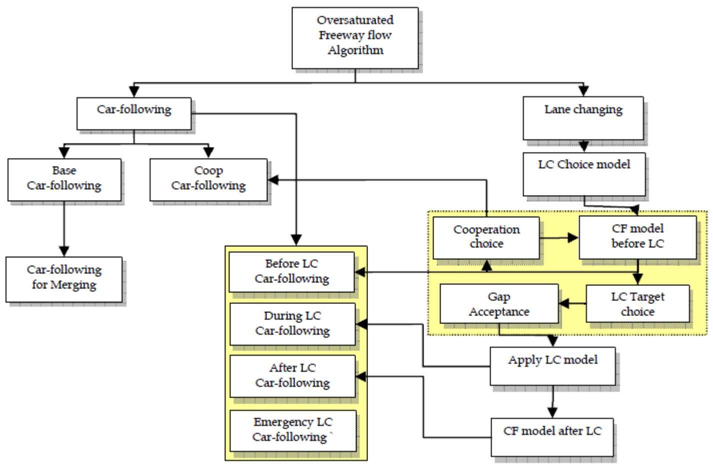


Figure 5.2.6: Skabardonis scheme proposed inside the NGSIM Project. *Source: Yeo et al. (2008).*

Newell model, to which they added security restrictions in order to avoid collisions during the simulation, and also to represent the physical limits of vehicles (such as maximum acceleration ratios and deceleration).

Thus, they defined it through Equation 5.2.21.

$$x_n(t+\triangle t) = \max\left\{x_n^U(t+\triangle t), x_n^L(t+\triangle t)\right\} \tag{5.2.21}$$

To calculate the distance traveled during a simulation step, they could define their upper bound using the minimum value of the distance generated by: the basic rule of this type of car-following model, the acceleration capabilities of the vehicle, the desired maximum speed and the maximum safe distance. Thus, they defined it with Equation 5.2.22.

$$x_n^U(t+\triangle t) = \min\left\{\begin{array}{c} x_{n-1}(t+\triangle t-\tau_n) - l_{n-1} - g_n^{jam}, \\ x_n(t) + v_n(t)\cdot\triangle t + a_n^U\cdot\triangle t^2, \\ x_n(t) + v_n^f\cdot\triangle t, x_n(t) + \triangle x_n^s(t+\triangle t) \end{array}\right\} \tag{5.2.22}$$

And the lower bound is determined by the deceleration capability and by restricting the vehicle to maintain the current position without going backwards. So, from the Equation 5.2.23.

$$x_n^L(t+\triangle t) = \max\left\{x_n(t) + v_n(t)\cdot\triangle t + a_n^L\cdot\triangle t^2, x_n(t)\right\} \tag{5.2.23}$$

where:

$$\triangle x_n^s(t+\triangle t) = \triangle t\cdot$$
$$\left(a_n^L\cdot\tau_n + \sqrt{\left(a_n^L\cdot\tau_n\right)^2 - 2a_n^L\cdot\left(x_{n-1}(t) - x_n(t) - \left(l_{n-1} + g_n^{jam}\right) + d_{n-1}(t)\right)}\right)$$

$$d_{n-1}(t) = -\frac{\left(v_{n-1}(t)\right)^2}{2a_{n-1}^L}$$

where:

$\triangle t$ is the simulation step.

τ_n is the wave travel time for the nth vehicle.

l_{n-1} is the length of the $(n-1)$th vehicle.

g_n^{jam} is the distance between $(n-1)$th vehicle and nth when both are stopped.

$v_n^f(t)$ is the free flow speed of the nth vehicle at time t.

a_n^U is the maximum acceleration of the nth vehicle.

a_n^L is the maximum deceleration of the nth vehicle.

$v_n(t)$ is the speed of the nth vehicle at time t.

Moreover,

$$x_n^L(t+\triangle t) \le x_n(t+\triangle t) \le x_n^U(t+\triangle t) \tag{5.2.24}$$

The model proposed by Skabardonis is specially interesting for our research, because it can overcome Gipps limitations in relationship to the reproduction of stop-and-go situations.

Particle Hopping Models

Another development of vehicular traffic flow theory is particle hopping models. In these, a road is represented as a string of cells, which are either empty or occupied by exactly one particle. Movement takes place by hopping between cells. If all particles are updated simultaneously, then, formally, the particle hopping model are also cellular automata (CA). The technical difference between car-following and CA models for traffic flow is that in the latter space and time are discrete, whereas they are continuous in the mathematical treatment of car-following models (with some time-discrete exceptions that we commented on the introduction of this chapter). Simulations of car-following models discretize time but use continuous space. From a theoretical point of view, the methodology of particle hopping models is somewhere between fluid-dynamical and car-following theories, and they help to clarify the connections between these approaches.

In 1956, a CA model for traffic was proposed byGerlough (1956) and was later extended further by Cremer and Ludwig (1986). They implemented fairly sophisticated driving rules and also used single-bit coding, with the goal of making the simulation fast enough to be useful for real-time traffic applications. The bit-coded implementation, though, made it too impractical for many traffic applications.

In 1992, Biham et al. used a model with maximum velocity, applying it to both one- and two-dimensional traffic (Biham et al. (1992)). One-dimensional here refers to roads, etc., and includes multilane traffic. Two-dimensional traffic in the CA context usually means traffic on a two-dimensional grid, such as a traffic model for urban areas. Also in 1992, Nagel and Schreckenberg introduced a model with maximum velocity for one-dimensional traffic, which compared favorably with real world data (Nagel and Schreckenberg (1992)).

In this section we present several CA models that are candidate models for traffic. They all are defined by a certain number of sites. Each site can be either empty or occupied by exactly one particle. Also, in all models, particles can only move in one direction. The number of particles is conserved except at the boundaries. For traffic, particles model cars.

The stochastic traffic cellular automaton

In 1992, a stochastic CA for one dimensional vehicular traffic was proposed by Nagel and Schreckenberg (1992). The model is referred to as the Stochastic Traffic Cellular Automaton (STCA) and is defined as follows. Each particle (car) can have an integer velocity between 0 and v_{max} . The complete configuration at time step t is stored and the configuration at time step $t+1$ is computed from that, using a parallel or synchronous update. All cars execute in parallel the following steps:

1. Let g (gap) equal the number of empty sites ahead.

2. If $v > g$ (too fast), then slow down to $v := g$ (rule 1); otherwise if $v < g$ (enough headway) and $v < v_{max}$, then accelerate by one: $v := v + 1$ (rule 2).

3. Randomization: If after the above steps the velocity is larger than zero ($v > 0$), then, with probability p, reduce v by one (rule 3).

4. Particle propagation: each particle moves v sites ahead (rule 4).

The randomization incorporates three different properties of human driving into one computational operation: fluctuations at maximum speed, overreactions at braking, and retarded (noisy) acceleration.

When the maximum velocity of this model is set to one ($v_{max} = 1$), then the model becomes much simpler; each particle executes the following in parallel: if a site ahead is free, move with probability $1 - p$ to that site. Since the STCA shows different behavior for $v_{max} \geq 2$ than for $v_{max} = 1$, the literature distinguishes them as STCA/1 and STCA/2, respectively.

The deterministic limit of the STCA (Cellular Automata - 184)

One can take the deterministic limit of the STCA by setting the randomization probability p equal to zero, which is the same as simply skipping the randomization step. It turns out that when using a maximum velocity $v_{max} = 1$, this is equivalent to the cellular automaton Rule 184[1] in Wolfram's code[2]; so this model is known as CA-184. Many CA models for traffic are based on this model. Biham et al. (1992) introduced it for traffic flow, with $v_{max} = 1$. Nagel and Herrmann (1993)used it with v_{max} larger than one. It is also the basis of the two-dimensional CA models for traffic.

It has been reported that the general behavior of the model does not change due to this simplification. The only practical difference obtained by setting $v_{max} = 1$ is that the acceleration is now unbounded as well. Another important feature of CA-184 is that its fluid-dynamical limit is known. This limit is obtained by making the cells and time steps increasingly smaller while increasing the number of particles, such that the average density of particles remains constant.

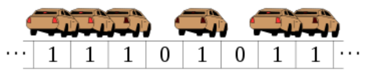
It may be conjectured that CA-184 is equivalent to the LWR model proposed by Newell in 1993. The existence of only two different wave speeds in Newell's model (one for each of the two flow regimes) is consistent with the behavior of CA-184. Furthermore, both models implicitly allow instantaneous acceleration and deceleration.

The asymmetric stochastic exclusion process

Probably the most investigated particle hopping model is the Asymmetric Stochastic Exclusion Process (ASEP). Its behavior is defined as follows.

1. Pick one particle randomly (rule 1).

[1]Rule 184 is a one-dimensional binary cellular automaton rule. In this model, particles (representing vehicles) move in a single direction, stopping and starting depending on the cars in front of them. The number of particles remains unchanged throughout the simulation. If we interpret each 1 cell in Rule 184 as containing a particle, these particles behave in many ways similarly to automobiles in a single lane of traffic: they move forwards at a constant speed if there is open space in front of them, and otherwise they stop.



[2]Wolfram code is a naming system often used for one-dimensional cellular automaton rules, introduced by Stephen Wolfram in 1983. The code is based on the observation that a table specifying the new state of each cell in the automaton, as a function of the states in its neighborhood, may be interpreted as a k-digit number in the S-ary positional number system, where S is the number of states that each cell in the automaton may have, $k = S^{2n+1}$ is the number of neighborhood configurations, and n is the radius of the neighborhood. Thus, the Wolfram code for a particular rule is a number in the range from 0 to $S^{S^{2m+1}}-1$, converted from S-ary to decimal notation.

2. If the site to the right is free, move the particle to that site (Rule 2).

The ASEP is closely related to CA-184 and STCA (i.e., both with maximum velocity set to one). The actual difference lies only in the manner in which sites are updated. CA-184 and STCA update all sites synchronously, whereas ASEP uses a random serial sequence. In order to compare itself with the other synchronously updated models, one has to note that the ASEP notably updates each particle an average of once after N single-particle updates. A time step in the ASEP is therefore completed after N single-particle updates.

5.2.2 Node Behavior

One of the major issues we need to address when performing a DNL is how traffic behaves in nodes, where the upstream demand flow may exceed the downstream supply.

Node management handles the movements of vehicles from one link to the next on their route. In this section, we describe some of the methodologies used in the literature to describe these movements, although it is notable that this is one of the less reported aspects of DNL models.

Here, we describe some of these methodologies, beginning with those based on queuing theory.

In order to estimate waiting times and queue lengths of isolated bottlenecks, FIFO (first in, first out) servers have been used successfully for years. However, non-stationary arrivals create difficulties for these applications, particularly when demand exceeds system capacity over a certain period of time. In 1995, Daganzo showed that the FIFO rule for non-isolated intersections will provide inadequate results if it does not incorporate the schockwave phenomenon, even if it is a simple network with two sequential server-defined entries and exits.

We should add some additional restrictions to queuing models in order to meet some basic physical rules for traffic flows. For example, we need to assign a physical length for each vehicle to calculate the bound capacity of each queue. However, we are not incorporating the shockwave phenomenon with these additional restrictions, because the queuing models are not able to describe how vehicles propagate in space.

In 1998, Ben Akiva proposed a DNL based on links divided into two parts: moving and queuing (Ben-Akiva et al. (1998)). The moving part is the portion of the link where vehicles can move at a certain speed, in accordance with traffic flow theory. The queuing part of the link represents the area where vehicles have stopped to form a queue. Ben Akiva proposed a deterministic queuing model for this second part. (see Section 3.2.5).

In 1999, Akçelik proposed a queuing model for signalized intersections (Akcelik et al. (1999)). With this model, he simulated queue propagation depending on shockwave propagation. Furthermore, it is important to note that in his proposal, Akçelik takes into account the phases of the intersection control to calculate wave movement.

However, there are also other interesting methodologies that are not based on queueing theory but nevertheless warrant attention.

In 2001, Mahmassani proposed a mechanism for transferring traffic flow in the nodes from link to link (Mahmassani (2001)). The method includes the number of vehicles that remains in the queue, and the number of vehicles added or subtracted from each link at each simulation step. In 1994, Mahmassani proposed a wide range of traffic control measures at intersections in order to reflect the entry/exit capacity constraints that are the basis of this transferring node mechanism.

In 2003, Mahut proposed a management mechanism at each of the intersection nodes in the network (Mahut et al. (2003)). This method is based on the turn definitions for each allowed movement from one link to another. Each turn is defined by an access code and a saturation flow rate per lane. The advantage of this methodology is that it does not require the definition of a certain geometry such as lane width, turning angles or intersection dimensions.

5.2.3 Lane Changing Methods

The second problem to take into account when developing a DNL model is analogous to the first, but refers to the lane discretization of the links. In real networks, movements at intersections usually occur only in specific lanes. Therefore, vehicles must change to the required lane to follow the route they have been assigned. These lane changes may result in a demand flow in a lane that exceeds the supply that this link can assume. To properly deal with these phenomena, it is necessary to establish methods for explicitly taking into account lane changes.

Aside from the abovementioned proposal by Skabardonis, who combined a car-following model with a model of lane changing behavior, we analyze some other methodologies used in the literature to solve this traffic problem.

In the model proposed by Mahut et al. (2003), vehicle trajectories in the links are modeled implicitly, rather than explicitly. Specifically, each driver chooses the lane for getting in and out of a link. The driver makes this decision just before arriving to the link and cannot change once inside. The main argument that justifies this approach is that, in order to model the flow congestion patterns of certain paths, it is sufficient to model the mandatory lane changes. These lane changes are carried out in order to enter an exit from the link through the permitted turns.

The simulation software AIMSUN (see Section 3.2.1), from its mesoscopic viewpoint proposes a lane change model governed by this set of rules:

- Input and output lanes are chosen before the vehicle enters the link, and then cannot be reconsidered (as in the Mahut proposal).
- The FIFO rule is applied according to the departure lane, i.e., once a vehicle has entered the link it cannot overtake any vehicle going to the same departure lane.
- The lane change is assumed instantaneously, which means that it does not consider any rule on gap acceptance.
- (From the above two rules: overtakings are allowed only between two vehicles with different destination lanes.)

5.3 Proposed Dynamic Network Loading Model

5.3.1 Justification of the Proposed DNL Model

Because the objective is to develop a new DNL model as one of the main components of the proposed iterative DTA procedure, it is necessary to have an efficient DNL process with the shortest possible computational times, even in the case of medium- or large-sized traffic networks. It is also very important to try to reach a balance in the number of parameters without losing a suitable description of the traffic flow dynamics. On the one hand, one handicap to calibration could be: the more parameter values that depend on the network, the more difficult it will be to find the most suitable values for reproducing reality. On the other hand, failure to use any input parameter will probably make it more difficult to reproduce traffic behaviors that should not be overlooked and that are related to network characteristics like geographical location or other factors.

Taking into account the above requirements for selecting a DNL approach in a DTA model, we can rule out models based on macroscopic simulation. These models, which are very suitable in terms of execution times, do not reproduce traffic behaviors that are needed for a dynamic network model embedded into a DTA scheme.

Furthermore, microscopic simulation models are also routinely discarded because they usually require very high computational time and a large number of input parameters, which in practice are difficult to calibrate.

Thus, the most suitable option is a model that combines the best properties of both microscopic and macroscopic models, i.e., a mesoscopic model. Our goal is to develop models that can reproduce reality like microscopic models, but without so many details. In this way, we can achieve better execution times and remove excessive input parameters that are difficult to calibrate.

The proposed DNL considers time in a continuous form instead of the microscopic traffic simulation method of discretizing time in previously defined steps. Our proposed model is different from microsimulation models in that space is considered as discrete rather than continuous, except for the case of cellular automaton models, which work with cells. Thus, in our case, there is no explicit control of what happens with vehicles inside the links. The model focuses only to specific points of the link that are essential for correctly defining network loadingcorrectly. However, the proposed model keeps the key feature of microscopic models: demand discretization considered for each vehicle.

To clarify the position of the proposed model compares to existing ones, we will refer to the modified Astarita classification (5.2.1) scheme for analyzing time, space and demand discretizations. In our proposed model:

- Time is considered continuous, therefore the time coordinate is zero.
- Space is considered discrete. As already mentioned, we lose continuous control of the vehicle's location in the network and we consider only link control. Therefore, the space coordinate is a link length.

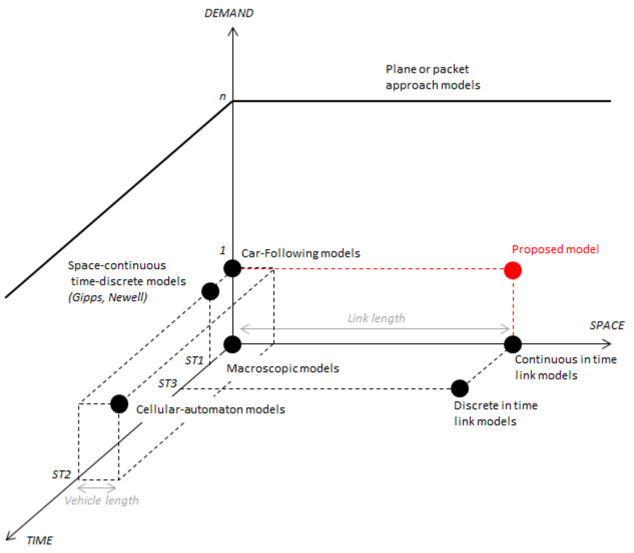

DEMAND
Plane or packet approach models
n
1
Car-Following models
Proposed model
Space-continuous time-discrete models (Gipps, Newell)
Link length
SPACE
ST1
Macroscopic models
Continuous in time link models
ST3
Discrete in time link models
Cellular-automaton models
ST2
Vehicle length
TIME

Figure 5.3.1: Proposed model located at proposed scheme classification.

- Demand is considered discrete and fully disaggregated vehicle to vehicle. This is crucial for modeling situations which into account different vehicle classes. Therefore the demand coordinate is one.

Thus, we locate the proposed model at the point $(0, arc\,length, 1)$ in the previously presented scheme classification, as shown in Figure 5.3.1.

5.3.2 Proposed Dynamic Network Loading Model

As pointed out before, the proposed DNL link simulation is continuous in time with disaggregated demand. Link flow propagation is solved without evaluating flow at intermediate points along the links.

Like some microscopic models based on car-following theory or cellular-automaton, vehicles in the proposed model move by trying to maximize their speeds in the presence of certain constraints. These constraints ensure that vehicle trajectories satisfy position, speed and acceleration bounds while trying to avoid vehicular collisions.

In order to simplify understanding of the models, we will first consider traffic propagation in a single link in the network, which will also have a single lane. In this way we will study the model that vehicles follow while entering and moving in and out of this link. We will only take into account the longitudinal movements in the link (by assuming that links have only one lane) and without regard to traffic conflicts produced by the interaction of several links or by node management.

Then the model will be extended to the case of two links with only one lane each and which are joined by a node. In this way, we will study new situations that did not exist in the initial case. On the one hand, it should explain node management, i.e., when and why a node can generate congestion in its upstream link or cannot get the desired flow to its downstream link. On the other hand, we must investigate how link flow affects its next downstream link, and vice versa.

Finally, we will introduce into the proposed model necessary changes for taking into account lanes within each link in the network. The forward dynamics will be the same as those proposed for the case of links without lanes. First, we will analyze the conflicts that arise in single-link flow propagation when vehicles change lanes. Then we will try to reproduce the congestion caused by these traversal movements. After that, we will extend the model to two links joined by a node, in order to study lane changes produced by node management. Finally, we will consider lanes and the turns allowed.

5.3.2.1 Proposed Models without Lanes

Assumptions

Before beginning to study the presented model, we will specify some necessary assumptions made during its development, both in terms of attributes that the model needs, and the simplifications needed to formulate it comfortably.

First, we consider as many vehicle classes as the user deems necessary. Each vehicle class should be characterized by two attributes: the mean reaction time and the mean effective length of the vehicles in the class. The vehicle reaction time is the time required by the driver to observe, decide, react and execute actions which change itsroad behavior. The effective length of a vehicle is equal to a vehicle's length plus the distance between this vehicle and its preceding vehicle when they are stopped in a queue. In order to avoid that all vehicles belonging to a class are clones, the user can also define a deviation for using the average values of a class to generate the reaction time T_{veh} and the effective length L_{veh} of each vehicle in this class.

Disaggregation of each individual vehicle allows use of different vehicle classes in the problem. Therefore, the proposed network loading is a multiclass traffic simulation. This model considers as many vehicle classes as the user of the model deems necessary. Each vehicle class is characterized by two attributes: the mean reaction time and the mean effective length of the vehicles in the class.

For network links, the first developed model needs only two attributes: the link length and the maximum allowed speed in the link. When a vehicle enters a link, we consider its speed to be the maximum allowed in this link. Hence, we suppose that this vehicle is able to achieve this speed and that it will do so in free-flow traffic conditions. We cannot define the maximum allowed speed for each vehicle because that could cause problems in managing link flow propagation, which is under FIFO assumptions. Thus, if a vehicle enters a link later than another vehicle, and they exit from the same lane, overtaking is not possible in this link and they must leave it in the same entry order. In addition, if we consider the maximum speed limitation for each vehicle, we could also have problems with the shortest path calculations for the first DTA iteration. In this case, we consider that link cost is in free-flow traffic conditions, so it is equal to the time it takes to traverse this link at maximum speed (and therefore we need maximum speed as a link attribute, not a vehicle attribute).

We consider units of space and time in meters and seconds, respectively. Thus, the speed unit is meter/second.

Taking into account that we consider the problem from a discrete demand point of view, the flow propagation process is described vehicle to vehicle. Usually, we formulate it from one of the following two relationships: time-dependent position $(x(t))$ or space-dependent time $(t(x))$. In this case we formulate the flow propagation model from the latter.

So, we formulate the problem by defining a function for each vehicle which returns the time when that vehicle is in a specific link position x: $(t_{veh}(x))$.

It is important to note that it is not sufficient to know the time when a vehicle occupies a certain link position. What is really interesting is to control the times when the vehicle reaches and leaves that position. Otherwise, that function

$(t_{veh}(x))$ may not be uniquely defined, i.e., the vehicle can be in position x at different times. So, we define two functions: one that provides the time when a vehicle arrives at a certain position $(t^a_{veh}(x))$, and one that provides the time when it departs from this position $\left(t^d_{veh}(x)\right)$. In that way, both functions will be uniquely defined.

The aim of a DNL procedure is to calculate time-dependent traffic variables (flow, density, speed) for each link, using the time-dependent flow assignment in each of the network paths. Calculating these link variables requires knowledge of the input and output times of the vehicle for each of the network links. Thus, at each instant of time we will know the link where each vehicle is, and therefore the link state (e.g., the number of the vehicles within it). Also, we can easily calculate the time the vehicle takes to cross the link, allowing us to know the average circulation speed.

Therefore, in the proposed model we consider only two positions at each link where we evaluate the functions $t^a_{veh}(x)\,,\, t^d_{veh}(x)$. Obviously the two fixed positions are the link's initial position (x_s) and the final position (x_t).

With respect to these positions, we should make some extra specifications. First, a vehicle is considered to be in a certain position when the bumper of the vehicle reaches this position. Second, we need to specify the exact point in the link for considering the terms "initial and final positions".

For practical purposes we need an initial, "separate" space in each of the links. This initial space is a cell whose length is the weighted average of the effective lengths of all the vehicle classes involved in the problem. Among other reasons, we need to differentiate the beginning of the link to control the inflow. We also use this cell to take appropriate measures when reaching maximum flow capacity, so that flow propagation remains realistic.

Also, this initial cell allows us to define the link's initial position (x_s). The initial link position is not the physical position in which the link starts. Rather, it is the end of the first separated cell in the link. Thus, when we calculate the time when the vehicle arrives at the initial link position, we take into account the time when the vehicle physically enters that link.

Furthermore, the final link position (x_t) coincides with the position in which the link physically ends. Thus, when we calculate the departure time from the final position, we calculate the time when the vehicle leaves the link.

From the above considerations, the models presented below always follow the same methodology. Given a certain link in the network, we need to calculate

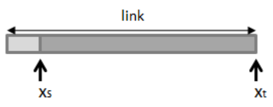


Figure 5.3.2: Link without lanes scheme with differentiated initial and final positions.

the following times for each vehicle:

- Arrival time of that vehicle to the initial link position $(t^a_{veh}(x_s))$.
- Departure time of that vehicle from the initial link position $\left(t^d_{veh}(x_s)\right)$.
- Arrival time of that vehicle to the final link position $(t^a_{veh}(x_t))$.
- Departure time of that vehicle from the final link position $\left(t^d_{veh}(x_t)\right)$.

Single Link without Lanes Model

The first developed model only takes into account the longitudinal movements of the vehicle within a link. This is why we consider a single link isolated. Also, to prevent traversal movements proper to the lane changes, we should not consider the existence of lanes in the link.

Consider a certain link in the network with length L_l and maximum allowed speed V_l, and vehicle *veh*. We define below time calculation functions for this first model.

Vehicle arrival time at the initial link position

When we calculate the time when a vehicle arrives at the initial position of a link $(t^a_{veh}(x_s))$, we obtain information about the time when the vehicle has entered this link and has occupied the initial cell, differentiated of the rest of the link as we explained before.

Since in this first case we work on an isolated link, there is no possibility that the upstream node controls the vehicle's arrival to the link. Thus, we start from a situation in which we randomly generate the time when a vehicle is located at the entrance of the link and has arranged to enter it (t^g_{veh}).

With the aim of calculating the vehicle arrival time at the initial link position, we must know the status of the first cell of this link. It is for this reason that the cell's own link should always be able to know the status of its first cell. This state can be busy or free cell, taking into account whether or not another vehicle exists in that space, which would mean that this vehicle has entered the link, has occupied the first cell and has not yet been able to advance inside this link.

If the first cell is busy, the vehicle must wait until the cell state changes in order to calculate its arrival time at the initial link position. The link itself must manage the vehicles that are waiting to get into it. To this end, we save into the link an ordered list which stores the vehicles that are waiting for space to enter the first cell of this link, preserving arrival order.

When the first cell changes its state from busy to free, the link informs the first vehicle that has tried to enter, and then we allow this vehicle to occupy the first cell of this link. Thus, the arrival time at the initial link position is equal to the time in which the first cell is released, plus the time that the vehicle needs to cross this first cell at the maximum allowed speed in this link, plus the driver reaction time. So, in this case, the arrival time at the initial link position is expressed by Equation 5.3.3.

$$t^{a}_{veh}\left(x_s\right) = t_{iniCellFree} + t_{crossIniCell} + T_{veh} \tag{5.3.1}$$

where:

$t_{iniCellFree}$ is the time when the first cell changes its state from busy to free.

$t_{crossIniCell}$ is the time that a vehicle takes to cross the link's first cell length $(L_{iniCell})$ at the maximum allowed speed in the link (V_l).

$$t_{crossIniCell} = \frac{L_{iniCell}}{V_l} \tag{5.3.2}$$

Furthermore, we change the state of the first cell, which becomes busy.

Otherwise, if the first cell is free, we allow the vehicle to occupy it, i.e., we can calculate the arrival time of the vehicle at the initial link position, that is, the time when the vehicle reaches the link and the time that it takes to cross the first cell. Thus, in this case, this time is calculated by Equation 5.3.3.

$$t^{a}_{veh}\left(x_s\right) = t^{g}_{veh} + t_{crossIniCell} \tag{5.3.3}$$

Furthermore, we change the state of the first cell, which becomes busy.

The arrival time of the vehicle at the initial link position is calculated by Equation 5.3.14.

$$t^{a}_{veh}\left(x_s\right) = \max\left\{t^{g}_{veh} + t_{crossIniCell}, t_{iniCellFree} + t_{crossIniCell} + T_{veh}\right\} \quad (5.3.4)$$

Vehicle departure time from the initial link position

Once the vehicle is in the first cell of the link, we try to calculate its departure time from that position $\left(t^{d}_{veh}\left(x_s\right)\right)$. To do this, we need to know whether there is free space in the link (without counting the first cell) to advance or not. For this reason, it is essential to keep link capacity control.

Each link must have an attribute that is responsible for controlling its capacity at each instant of time. Thus, we should always know the free space in the link (in meters), since the vehicle itself is occupying it. When the vehicle wants to move into the link, it should ask the link if its free space is enough to enter or not. Since the presented model allows the use of vehicles of different classes, the link compares the effective length of this vehicle with the available space in the link.

If there is enough space for this vehicle, its departure time from the initial link position coincides with the time when it is asked if it could move forward. In addition, you must subtract vehicle length from free space in this link and the vehicle becomes part of all vehicles that are in the link. So, in this case, this time is calculated by Equation 5.3.5.

$$t^{d}_{veh}\left(x_s\right) = t^{a}_{veh}\left(x_s\right) \quad (5.3.5)$$

If there is no space for that vehicle in this link, it should wait until the situation changes. In this case, we are still not able to calculate the time $t^{d}_{veh}\left(x_s\right)$. The link itself will manage the vehicle that is waiting in that first cell. When the available space in the link changes, we inspect again whether or not the vehicle has space, and so on until space is obtained. In this case vehicle departure time from the initial link position is equal to the time when the link has enough space for this vehicle, plus the driver reaction time. Thus, we express this time by Equation 5.3.6.

$$t^{d}_{veh}\left(x_s\right) = t_{freeLink} + T_{veh} \quad (5.3.6)$$

where:

$t_{freeLink}$ is the time when the link has enough free space to accommodate the vehicle.

Definitively, vehicle departure time from the initial link position is calculated by Equation 5.3.7.

$$t_{veh}^{d}(x_s) = \max\left\{t_{veh}^{a}(x_s), t_{freeLink} + T_{veh}\right\} \tag{5.3.7}$$

Vehicle arrival time at the final link position

Once the vehicle has moved into the link, our objective is to calculate the time when it arrives at the final link position ($t_{veh}^{a}(x_t)$). In the case of dealing with a fluid traffic situation, the arrival time of the vehicle at this position is equal to the time when the vehicle has left the initial position of this link, plus the time that it takes to cross the link (minus the first separated cell) at the maximum speed allowed on it. So, the Equation 5.3.8 shows the calculation.

$$t_{veh}^{a}(x_t) = t_{veh}^{d}(x_s) + t_{crossLink} \tag{5.3.8}$$

where:

$t_{crossLink}$ is the time that a vehicle takes to cross link l without taking into account the first cell

$$t_{crossLink} = \frac{L_l - L_{iniCell}}{V_l} \tag{5.3.9}$$

However, we may be working in a congested traffic situation or in a situation in which vehicles cannot circulate at the maximum allowed speed on the corresponding link. In order to address this problem we need to introduce the concept of vehicle leader (l_{veh}).

When a vehicle enters the link and occupies its first cell, we need to record who its leader is. Since we are considering the case of an isolated link without lanes, its leader is the last vehicle that has entered this link before it. Considering the FIFO rule mentioned above, the leader must always leave the link preceding the lead vehicle. Thus in the case where there is congestion in the link, we use leader information to calculate the leading vehicle arrival time at the final position. When the vehicle leader leaves the final position, this place can be occupied with the vehicle that follows the leader. In this way, the vehicle arrival time at the final link position is equal to the time that the leader has left this position, plus the time that the vehicle takes to cross the leader's effective length, i.e., the time that the vehicle takes to hold the space that the leader occupied. Also we need to take into account the reaction time of the driver of this vehicle. So in this case, this time is calculated by Equation 5.3.10 .

$$t_{veh}^{a}\left(x_t\right) = t_{l_{veh}}^{d}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh} \tag{5.3.10}$$

In any case, to calculate the time when the vehicle arrives at the final link position, we should take the maximum of the calculated times for the two considered traffic situations: fluid or congested. Thus, on the one hand, we follow the FIFO rule to ensure that the vehicle never overtakes its leader; and on the other hand, we ensure that the vehicle will have enough time to cross the link at the maximum allowed speed in this link. Therefore, Equation 5.3.11 shows this maximum value.

$$t_{veh}^{a}\left(x_t\right) = \max\left\{t_{veh}^{d}\left(x_s\right) + t_{crossLink}, t_{l_{veh}}^{d}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh}\right\} \tag{5.3.11}$$

If the vehicle does not have a leader, we consider that the second factor of this maximum is zero.

About calculation of the vehicle's arrival time at the final link position, we observe that we cannot calculate this time when the vehicle has a leader, but we do not know the departure time of the leader from the final position. If this occurs, the vehicle must wait until that time is known. The leader is the one who manages this process, the vehicle which is led by it and that is waiting to know its departure time. When this time exists, the leader informs the follower so that it can calculate its time $t_{veh}^{a}\left(x_t\right)$.

Vehicle departure time from the final link position

Finally, the vehicle must know the time when it will leave the final link position $\left(t_{veh}^{d}\left(x_t\right)\right)$. Here, it is the downstream node which manages whether or not the vehicle can leave the link. Since in this first proposed model it is considered an isolated single link, we consider that the vehicle could always depart from this position at the same moment that it arrives. Thus, in this case, this time is calculated by Equation 5.3.12.

$$t_{veh}^{d}\left(x_t\right) = t_{veh}^{a}\left(x_t\right) \tag{5.3.12}$$

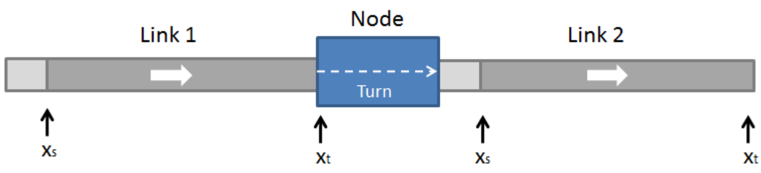


Figure 5.3.3: Link-node-link scheme with differentiated initial and final positions.

Link-Node-Link without Lanes Model

The second proposed model is an extension of the previous case in which we consider two consecutive links in the network connected by an intersection. This intersection is the downstream node of the first link and the upstream node of the second one. In this way we take into account real traffic situations that we cannot analyze or solve in the first proposed model. With respect to the lanes, we continue without regarding their existence, in order to separate the problems that arise when vehicles move in a traversal way for lane changes. This special case is explicitly discussed below.

The proposed model works with a network simplification that considers nodes as abstract attributesthat are responsible for managing the in and out traffic of the links, but without physical characteristics. For this reason, we cannot calculate the time that a vehicle needs to cross a node, because we have neither geometric information nor speed limits inside it.

In the physical representation of the proposed network, the links practically overlap in spatial terms. Thus, the node crossing time is imputed to the links that the vehicle circulates between. However, we know the importance of considering a fictitious node transfer time that we could use to appropriately penalize vehicle travel time when it passes from a link to its consecutive link. In this way, we want to reproduce the speed reduction that vehicles experience when they turn, or the caution of drivers when they are changing their circulation link. This node transfer time must be small compared with the total travel time experienced in the links in which the vehicle is circulating. In addition, as is discussed below, we could even ignore this time in searching for a solution to a problematic proposed situation.

In this scenario we calculate the vehicle departure time from the initial link position $\left(t^{d}_{veh}\left(x_s\right)\right)$ and the vehicle arrival time at the final link position $\left(t^{a}_{veh}\left(x_t\right)\right)$. This identical to the first case we explained earlier: the single isolated link scenario. However, we need to change the calculation of the arrival time at the initial position $\left(t^{a}_{veh}\left(x_s\right)\right)$ and the departure time from the final position $\left(t^{d}_{veh}\left(x_t\right)\right)$, since these are themselves directly affected by the flow propagation managed by nodes located upstream and downstream from the link, respectively.

To analyze the influence of the node flow management in each of these calculations, we specify below how the intersection operates in the proposed model.

When the vehicle reaches the final link position, it should ask its downstream node if it can follow its route, i.e., if we can calculate its departure time from the final position that it is occupying. And therefore we can calculated its arrival time at the initial position of the next link in its assigned route.

To answer this question, each node in the network must define its turn set (among other attributes). As there are now no lanes in the link, turns are defined at the link level, particularly for two of them: the origin link and the destination link . The first one should be a link that is located upstream from the node, and the second link must be located downstream from it.

The path assigned to a vehicle must be validated, i.e., the turn defined from one link to the next link in the assigned route should always be allowed. Even so, when a vehicle wants to know if it can pass to the next link, the first thing we check is whether or not the turn is defined for the vehicle that needs to follow its path.

Then, we must check if this turn has traffic lights. If traffic lights exist, we must verify if they allow the vehicle to cross the node, that is, if it is in a green phase or not. Otherwise, we must verify that the vehicle has node priority and that no conflict exists between the trajectory of that vehicle and the other vehicles crossing at the same time.

Finally, we need to check if the first cell of this turn's destination link is free or busy. Because simplification of the network nodes (in which the crossing node time is a temporal penalty that could even be zero) allows the vehicle to leave the final position only if it has free space in its next link. This is discovered when it consults whether or not it can cross the node.

Obviously, the node flow management can generate congestion in the links located upstream. At the same time, this management could cause the input flow of the downstream links to be much lower than their capacity. In short, the node is responsible for traffic management in the network.

For practical purposes, vehicle departure time from the final link position $\left(t_{veh}^{d}\left(x_{t}\right)\right)$ is equal to the arrival time at this position if the node allows the vehicle to follow its assigned route. Otherwise, the vehicle waits to pass. It may be waiting because:

- the turn has its traffic light in a red phase,
- the node is occupied by another vehicle whose trajectory is in conflict with the trajectory of the studied vehicle,
- the first cell of the destination link of this turn is busy.

The node manages the vehicles that, for these reasons, are waiting. It is also responsible for informing the corresponding vehicle when the situation that blocked its pass changes. The delay caused by the management of the node is added to the travel time of the link in which the vehicle was waiting to pass. Equation 5.3.13 calculates this time.

$$t_{veh}^{d}\left(x_{t}\right)=\left\{\begin{array}{ll} t_{veh}^{a}\left(x_{t}\right) & if\,the\,vehicle\,can\,pass\,the\,node\,inmediatelly \\ t_{canPass}+T_{veh} & if\,the\,vehicle\,must\,wait\,to\,pass \end{array}\right. \tag{5.3.13}$$

where:

$t_{canPass}$ is the time when the node allows the vehicle to pass.

If we now pay attention to the proposed calculation of the arrival time at the initial link position $(t_{veh}^{a}\left(x_{s}\right))$, we can observe that the proposal is only useful for the first links of all the paths in the network. These links do not have upstream nodes, and therefore arriving vehicles are generated from a random procedure that tries to reproduce vehicle arrivals from different locations, which themselves are generated by demand.

For all other links, the proposed approach simplifies this time calculation. The vehicle may not leave a link until it is sure that it can enter its next link. So, to obtain the arrival time at the next downstream link, we add the following to the departure time from the final position of the upstream link:the time that the vehicle takes to cross the node and the time to cross the first cell of the next link. As explained above, the temporal penalty of the crossing vehicle may become negligible. . So, we calculate this time with Equation 5.3.14 .

$$t_{veh}^{a}\left(x_{s}\right)=t_{veh}^{d}\left(x_{t}(previousLink)\right)+t_{crossNode}+t_{crossIniCell} \tag{5.3.14}$$

where:

$t_{crossNode}$ is the penalty time for transferring the node.

Definitively, this second model allows us to introduce intersections in the proposed scheme. These are represented as nodes in the graph (network). Incorporating intersections created conflicts that did not appear in the case of single isolated links, and which also we could not solve. These conflicts can be resolved by interpreting events at real intersections and translating them to node management rules.

5.3.2.2 Proposed Models with Lanes

Assumptions

In addition to the assumptions used to develop models of links without lanes, we must appropriately specify some extra changes in order to take into account lanes in the network links.

The first consideration is about network links. We need to expand the previous definition of the link length attribute and to consider it at the lane level, i.e., the attribute lane length (L_c). In this thesis we simplified all lanes into a link with the same length. For example, in the case of a curved link with several lanes, the inner and outer lanes in fact have different lengths; but we have applied a unique length to both lanes: the assimilated link length for each lane.

About the other attribute defined at the link level, we do not need to extend the maximum allowed speed (V_l) to the lane level, because all lanes inside a link have the same maximum speed: the speed of the link.

Aside from these attributes, we need to define the number of lanes in each of the links in the network (n_c).

The problem is considered in the same way as the previous case (without lanes). We begin by defining two functions of time calculation for each vehicle. Specific positions on the link return the time when the vehicle arrives and leaves that position $\left(t^a_{veh}\left(x\right), t^d_{veh}\left(x\right)\right)$. The definition of these times is still at the link rather than the lane level, although we should use lane conflicts and lane attributes to define the functions at this link level, as we discuss below.

With respect to other considerations of the initial and final link positions and their treatment, we now consider them to be in exactly the same form, taking into account that the vehicle will be in a certain lane of this link. In the proposed

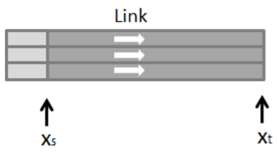


Figure 5.3.4: Link with lanes scheme with differentiated initial and final positions.

model, we justify the use of two differentiated positions, where we will evaluate our calculation functions: the initial link position (x_s) and the final link position (x_t).

As for the previous case, we need to separate a space at the beginning of each of the network links. Moreover, we now need to differentiate a first cell in each of the lanes of each link in the network. The size of these cells is the weighted average of the effective lengths of all the vehicle classes involved in problem.

In this case, we consider the initial link position (x_s) to be the end of the first cell of the lane where the vehicle enters the link. Furthermore, in regard to the final position (x_t), the position in which the link ends physically, in this case in which the lane by which the vehicle leaves the link ends.

We introduced changes in the model to explicitly treat the lanes within the links in order to reproduce the temporal delays that are caused by traversal movements of vehicles changing lanes within the link. At the end of this section, we detail what type of lane change is taken into account in the developed model and the assumptions made about it.

From the above considerations, the development of the models presented below always follows the same method. Given a certain link in the network, we need to calculate the following times for each vehicle:

- Arrival time of that vehicle at the initial link position $(t_{veh}^{a}(x_s))$.
- Departure time of that vehicle from the initial link position $\left(t_{veh}^{d}(x_s)\right)$.
- Arrival time of that vehicle at the final link position $(t_{veh}^{a}(x_t))$.
- Departure time of that vehicle from the final link position $\left(t_{veh}^{d}(x_t)\right)$.

Lane Change

In real life, when a driver wants change lanes, he or she should pay attention to the free space available in the adjacent lane before entering it. Because a driver cannot accept a gap that is smaller than the space needed for the vehicle, an unlimited number of gaps can be refused when the free space between two consecutive cars is less than the vehicle length. Under these conditions, a vehicle can delay its lane change indefinitely while blocking the lane until the lane change starts.

On the other hand, if a vehicle occupies a certain amount of space that another neighboring vehicle wants to occupy, one of the two vehicles should decelerate or accelerate to provide sufficient longitudinal separation between these two vehicles in order to the change lanes. Because of the considered assumptions, a vehicle travels at the maximum allowed speed in this link or it experiences delay. Because in the proposed model it is not possible to accelerate, in this case our of the two vehicles will decelerate.

Finally, we need to take into account that each lane change reduces the capacity of the link while the vehicle performs the lane change, because the vehicle doubles its space by occupying both lanes at the same time.

These commented situations about lane changes have drastic effects on the traffic flow of those lanes, and obviously can cause congestion.

The lane changes made by vehicles can be classified into two types:

- Mandatory: the vehicle must perform the lane change to continue its assigned path without violating the turn restrictions of the network nodes. Given two consecutive links of a given route, not all the lanes of the first link have a turn defined for the second one. Thus, in the case that the vehicle occupies a lane that prohibits access to its next link, the vehicle must perform a lane change before reaching the final position of the link in order to then perform the corresponding turn.

- Optional: the vehicle changes its lane in an attempt to increase its speed, passing to a lane that it perceives as faster.

The DNL model of lanes in the links only reproduces travel time delays in the links as a result of mandatory lane changes.

In order to reproduce traffic patterns occurring when vehicles change lanes, it is necessary to consider the lane through which the vehicle enters the link (c_{in}), and the lane through which it leaves this link (c_{out}). They are calculated at different times in the process. The input lane is calculated in the vehicle route's upstream link. When the vehicle arrives at the final position of the upstream link, we must calculate the lane that the vehicle will use to enter the next link in its route. We calculate the exit lane of a link when the vehicle enters the link, considering its input lane and the next link in its route.

Different methodologies exist for calculating input and output lanes. But in this version of the model, we propose calculating both by taking into account traffic conditions in different link lanes when we calculate input and output lanes. Moreover, this hypothesis minimizes the number of lane changes performed.

- Input Lane $\left(c_{in)}\right)$: When a vehicle reaches the final position of certain link and it is ready to leave this by the exit lane c_{out}, we must calculate the lane to enter the next link in the assigned route before consulting the node about whether or not it can pass. So, we just have to consider which lanes of the next link are accessible from the vehicle's current lane. Among all the options, we calculate by choosing the least busy laneat the time.
- Output Lane (c_{out}): If a vehicle enters a link through the lane c_{in} and this same lane is allowed to turn to the next link in its assigned route, then the vehicle does not change its lane and $c_{out} = c_{in}$. If, however, the turn between the input lane and the next link in its path does not exist, the vehicle must change lanes before it arrives at the final position of its current link. The procedure consists of determining which lanes are capable of performing the turn and to selecting the least busy among them.

Single Link with Lanes Model

As in the approach taken to formulate the previous model (which does not take into account the lanes inside the links), we will first develop a model that considers a single isolated link scenario, regardless of upstream and downstream nodes and the remaining links in the network. Thus, we will analyze the conflicts traffic propagation when the vehicles change lanes in a single link, and we will develop a model that reproduces the congestion caused by these traversal movements of the vehicles crossing.

Let there be a certain link in the network with length L_l, maximum allowed speed V_l and number of lanes n_c, and let there be a vehicle veh. We define below time calculation functions for the model that considers a single isolated link with lanes inside it.

Vehicle arrival time at the initial link position

In order to know the time when a vehicle arrives at a link by a certain lane and it occupies the first cell of this input lane, we must calculate the vehicle arrival time at the initial link position ($t^a_{veh}(x_s)$).

Since this is the case of an isolated link, we do not consider the existence of an upstream node that controls the vehicle arrival to the link through this lane. It is for this reason that we suppose a beginning situation where the vehicle arrival time at the initial link position is generated randomly (t^g_{veh}).

To calculate the arrival time of the vehicle at the initial link position through the entering lane c_{in}, we must know the situation of the first cell in this lane at the moment when the vehicle tries to enter it. It is for this reason that the selfsame link must know the state of the first cells of each of its lanes. As explained before, this state can be free or busy, taking into account if there is another vehicle that occupies that space or not.

If the first cell of the entering lane is busy, the vehicle must wait until the cell state changes in order to calculate its arrival time at the initial link position through this lane. The link itself must manage the vehicles that are waiting to get into it by that lane. To this end, we give the link one ordered list for each lane that the link has, thus storing the vehicles that are waiting for space to enter the first cell of this link through each of the lanes. This preserves the arrival order.

When the first cell of a certain lane changes its state from busy to free, the link informs the first vehicle that tried to enter this lane, and then we allow this vehicle to occupy the first cell of this lane. Thus, the arrival time at the initial link position through this lane is equal to the time when the first cell of the lane is released, plus the time needed to cross this first cell at the maximum allowed speed in this link, plus the driver reaction time. So, in this case, Equation 5.3.15 shows the calculation.

$$t^a_{veh}(x_s) = t_{iniCellFree} + t_{crossIniCell} + T_{veh} \tag{5.3.15}$$

Furthermore, we change the state of the first cell of this lane, which becomes busy.

If the first cell of a certain lane is free, we allow a vehicle to occupy it, i.e., we can calculate the arrival time of the vehicle at the initial link position; that is, the time when the vehicle reaches the link through this lane, plus the time that it takes to cross the first cell of this lane. Thus, in this case, this time is calculated by Equation 5.3.16.

$$t_{veh}^{a}\left(x_s\right) = t_{veh}^{g} + t_{crossIniCell} \tag{5.3.16}$$

Furthermore, we change the state of the first cell of this lane,which becomes busy.

Ultimately, the arrival time of the vehicle at the initial link position is calculated by Equation 5.3.17.

$$t_{veh}^{a}\left(x_s\right) = \max\left\{t_{veh}^{g} + t_{crossIniCell}, t_{iniCellFree} + t_{crossIniCell} + T_{veh}\right\} \tag{5.3.17}$$

Vehicle departure time from the initial link position

Once the vehicle is in the first cell of the entering lane in its corresponding link, we try to calculate its departure time from that position $\left(t_{veh}^{d}\left(x_s\right)\right)$. With this objective, we should know if there is free space in this lane of the link (without taking into account the first cell), so that the vehicle can advance. In the previous case, we solve the problem with explicit control over the capacity of the link. In this case, we consider the longitudinal discretization of the links in lanes; so the process is more complicated because the control of space should be performed on the lanes instead of on the links.

Each lane of the link must have an attribute that is responsible for controlling its free space (in meters)and that does not take into account the first cell because the vehicle is occupying it. When a vehicle wants to move into the link, we need to know if there is enough free space in the entering lane (that is, the vehicle's current lane). We simply compare vehicle effective length with the available space in the lane.

If there is enough space for this vehicle in this lane, its departure time from the initial link position coincides with the time when it asks if it can move forward. In addition, we must subtract vehicle length from free space in that lane, and the vehicle becomes one of the vehicles that are on the link. So, in this case, the time is expressed by the Equation 5.3.18.

$$t_{veh}^{d}\left(x_s\right) = t_{veh}^{a}\left(x_s\right) \tag{5.3.18}$$

If instead there is not enough space in this lane for that vehicle to advance in it, it should wait until the lane situation changes. In this case, we cannot still calculate its departure time from the initial link position $\left(t_{veh}^{d}\left(x_{s}\right)\right)$. The link itself will manage the vehicle that is waiting in that first cell of the entering lane of this link. When the available space in this lane changes, we repeat the free space check until it is obtained. In this case, the vehicle departure time from the initial link position is equal to the time when the lane has enough space for this vehicle, plus the vehicle driver reaction time. Thus, we calculate this time by Equation 5.3.19.

$$t_{veh}^{d}\left(x_{s}\right)=t_{freeLink}+T_{veh} \tag{5.3.19}$$

where:

$t_{freeLink}$ is the time when the entering lane of the link has enough free space to accommodate the vehicle.

When the vehicle moves into the link, then it is able to perform its lane change.

Definitively, vehicle departure time from the initial link position is calculated by Equation 5.3.20.

$$t_{veh}^{d}\left(x_{s}\right)=\max\left\{t_{veh}^{a}\left(x_{s}\right),t_{freeLink}+T_{veh}\right\} \tag{5.3.20}$$

Vehicle arrival time at the final link position

Once the vehicle has moved into the link through its corresponding input lane (c_{in}), our objective is to calculate the time when it arrives at the final link position $(t_{veh}^{a}\left(x_{t}\right))$ and occupies its output lane (c_{out}). In order to calculate this time, we need to consider that the vehicle may have to perform a lane change if its input and output lanes are different. It is important to note that the proposed model only allows the vehicle to perform its lane change if it is inside its input lane and not if it is already in the first cell.

We want to know whether or not a vehicle that needs to perform a lane change is able to. In order to solve this, we apply the following policy: we allow the lane change if all the lanes that the vehicle must cross to arrive to its output lane have enough available space. If so, the vehicle can transversely cross the link. To simplify, we consider that the vehicle performs the spatial change instantaneously. Thus, the vehicle only physically counts on its input and output lanes before and after the lane change, respectively. At the temporal level, we apply certain delays to the vehicle. These delays will depend on the density of crossed lanes and on the number of lanes that the vehicle needs to cross.

Also, it is important to note that we do not allow a vehicle to change its lane if the leader vehicle by its output lane ($l_{veh}^{c_{out}}$) has not made its own change. When a vehicle occupies the first cell of its input lane in a link, it must record who its leader in the exit lane is. This refers to the last vehicle that has entered the link and shares the output lane with the follower vehicle. Since the model must satisfy the FIFO rule, as in the previous model without lane differentiation, the vehicle leader by output lane should always leave the link preceding the vehicle being led.

If the lane change has been made (or if the vehicle does not need a lane change because it is already in a lane that allows it to exit the link) and there is thus sufficient space to move into the exit lane of this link, then the vehicle must leave the list of vehicles in the input lane, and it must free the space that it was occupying. Also, the vehicle passes to occupy an equal space of its effective length in the output lane, and it will be in the list of vehicles of this lane. Now, the vehicle objective is to calculate its arrival time at the final link position ($t_{veh}^{a}(x_t)$).

If the lane change cannot be performed because one of the lanes that the vehicle must cross is busy, then the vehicle must wait in its input lane until the situation changes. The vehicle will wait until all the lanes that it needs to cross, output lane included, have enough available space. The delay caused by this simplification of the model should be slightly greater than the real delay produced if the vehicle changes from one lane to the next, and so on until reaching the output lane. This is because it is more difficult to find a situation in which all the lanes have the available space at the same time than to have free space only in two lanes. On the other hand, because changes are instantaneous, the vehicles cannot physically occupy crossing lanes. This generates less congestion. Both simplifications mitigate each other by obtaining congestion caused by lane changes comparable to those reproduced in the microscopic simulator, as we can observe in Section 6.4.3.

If the lane change of a vehicle cannot be performed because the leader vehicle by output lane has not changed, then the vehicle must wait in its input lane and proceed as if it cannot pass because there is not available space in the link.

It is important to note that we cannot calculate arrival time at the final link position until the vehicle is not in its output lane. If we can calculate this time, we need to differentiate some traffic situations. If we consider the case of fluid traffic, the arrival time at the final position of the link through its output lane is equal to the time that the vehicle leaves from the initial link position, plus the time that the vehicle spends crossing the link (without the first cell) at the maximum speed allowed in this link, plus the time caused by the lane change (if done). Equation 5.3.26 shows the calculation.

$$t^a_{veh}\left(x_t\right) = t^d_{veh}\left(x_s\right) + t_{crossLink} + t_{LCH} \tag{5.3.21}$$

where:

t_{LCH} is the added time due to the lane change.

$$t_{LCH} = p_{LCH*} number\, of\, crossed\, lanes$$

where p_{LCH} is the time penalty for changing lane. The value of this constant depends on the characteristics of the studied network, so we focus on this important calibration parameter in the following results sections.

If the considered traffic situation is a case where the vehicles do not circulate in a free flow way in the output lane of the studied vehicle, then we need to use the information of the leader by output lane in order to calculate the vehicle arrival time at the final link position. When the vehicle leader leaves the final position, this can be occupied by the led vehicle. Thus, the arrival time at the final link position is equal to the time that the leader by output lane has left this position plus the time that the vehicle takes to cross the effective length of the leader vehicle in order to occupy its space. Also, we must take into account the reaction time of the driver of the vehicle.

In this way, we obtain the Equation 5.3.22.

$$t^a_{veh}\left(x_t\right) = t^d_{l^{c_{out}}_{veh}}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh} \tag{5.3.22}$$

Note that we cannot calculate the time $t^a_{veh}\left(x_t\right)$ if we do not know the departure time from the final link position of the leader vehicle by output lane. If this occurs, the vehicle must wait until that time is known. The leader vehicle by output lane manages this process and informs the led vehicle of the leader's departure time.

Definitively, to calculate the vehicle arrival time at the final link position in its exit lane, we must take the maximum of the defined times for all the possible situations listed. Thus, the model ensures that the vehicle does not overtake its leader by output lane, in accordance with the FIFO rule. Also, it ensures that the vehicle has enough time to cross the link without exceeding the speed limit. On the other hand, the model guarantees that the vehicle arrives at the final link position occupying its output lane, since it is not possible to calculate this time until the vehicle has occupied this lane.

So, this time is expressed by the Equation 5.3.23.

$$t^{a}_{veh}\left(x_t\right) = \max\left\{t^{d}_{veh}\left(x_s\right) + t_{crossLink} + t_{LCH}, t^{d}_{l^{c_{out}}_{veh}}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh}\right\} \tag{5.3.23}$$

If the vehicles do not have a leader by output lane, we consider that the second expression is zero.

Vehicle departure time from the final link position

Once the vehicle reaches the final position of this link, we must calculate the time when it will leave this position $\left(t^{d}_{veh}\left(x_t\right)\right)$. In this case, it is the downstream node which manages whether or not the vehicle can leave this link. Since in this model it is considered an isolated single link, we consider that the vehicle could always depart from a position at the same moment it arrives. Thus,

$$t^{d}_{veh}\left(x_t\right) = t^{a}_{veh}\left(x_t\right) \tag{5.3.24}$$

Link-Node-Link with Lanes Model

We present the last proposed model that is an extension of the case which has just been developed. We move on to consider two consecutive network links joined by a node. That intersection plays a key role in the traffic flow propagation on the links, since it is the downstream node of the first link and the upstream node of the second link. It should be responsible for managing both the exit of a vehicle from the first link as well as the entry of a vehicle to the second link. Thus, we take into account situations that were not possible to analyze or solve in the previous case of an isolated single link. Also, we consider traversal movements produced by vehicles when they change their lanes, and we

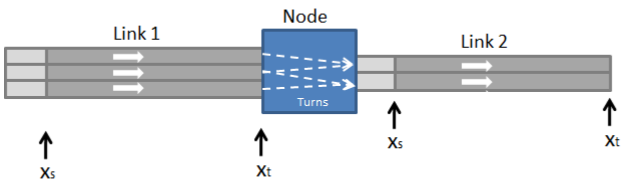


Figure 5.3.5: Link-Node-Link with lanes scheme with differentiated initial and final positions.

analyze the influence of the longitudinal discretization of the link in the model's node management.

In this case, we calculate the vehicle departure time from the initial link position $\left(t^{d}_{veh}\left(x_s\right)\right)$ and the arrival time at the final link position $\left(t^{a}_{veh}\left(x_t\right)\right)$. This is identical to the first case we explained about the single isolated link with lanes. However, we change the calculation of the arrival time at the initial position $\left(t^{a}_{veh}\left(x_s\right)\right)$ and the departure time from the final position $\left(t^{d}_{veh}\left(x_t\right)\right)$, since they are directly affected by the flow propagation managed, respectively, by the upstream and downstream nodes. In order to analyze the influence of node flow management at the lane level, we specify below how the intersection operates on the proposed model.

When a vehicle reaches the final link position through its output lane, it should ask its downstream node if it can follow its assigned route, i.e., if we can calculate the departure time from the final link position through its output lane and, therefore, if we can calculate its arrival time at the initial position of the next link in its assigned route.

To answer this question, each node in the network must have defined its own turn set. We define a turn between two lanes of two links: the source link in the turn and the destination link in the turn. The first one should be a link located upstream from the node, and the second one, located downstream from it. The turn is responsible for managing the vehicle movement from its output lane of the origin link to its entering lane of the destination link.

In order to know whether or not a vehicle can perform the corresponding turn, the first check that we need to perform is to verify if this turn exists, i.e., if the movement from the output lane of the origin link to the input lane of the destination link exists. Then, we must check if this turn has a traffic light and, in the case that it exists, if it is in a green phase. Otherwise, if a traffic light does not exist for this turn, we need to ensure that the vehicle has priority on this node, i.e., there will be no conflict between the trajectory of that vehicle and other vehicles that may be crossing at the same time.

Finally, we need to check if the first cell in the input lane of the destination link of this turn is free or busy. Because we simplified the nodes in the network (whereby crossing node time is a temporal penalty that could even be zero), we only allow the vehicle to leave its final position if the next link in its input lane has free space.

Therefore, if the node allows that vehicle to follow its assigned route, then the vehicle departure time from the final link position is equal to the arrival time at this position. Otherwise, the vehicle remains waiting to pass. It may be waiting because:

- the turn has its traffic light in a red phase,
- the node is occupied by another vehicle, whose trajectory is in conflict with the trajectory of the studied vehicle,
- the first cell of the destination link's input lane for this turn is busy

The node manages the vehicles that are waiting for these reasons, and it is responsible for informing the corresponding vehicle when the situation that blocked its pass changes: the traffic light phase that controls its turn changes to green, the vehicle that occupied the space arrives at its next link, or the first cell of the input lane is released.

The delay caused by the management of the node is added to the travel time of the link in which the vehicle is waiting to pass. By taking advantage of the fact that the information can be disaggregated for each lane of the link, we save the information about the link travel times, depending on the destination links in the turns. In this manner, we can give travel times experienced by the vehicles in certain links, depending on the next link in its assigned route. These detailed times are critical for calculating time-dependent shortest paths during the DTA iterations.

So, this time is calculated by the Equation 5.3.25.

$$t^{d}_{veh}\left(x_t\right) = \max\left\{t^{a}_{veh}\left(x_t\right), t_{canPass} + T_{veh}\right\} \tag{5.3.25}$$

where:

$t_{canPass}$ is the time when the node allows a vehicle to pass because one of the blocking situations changes.

If we now pay attention to the proposed calculation of the arrival time at initial link position ($t_{veh}^{a}\left(x_s\right)$) in a certain input lane, we can observe that the proposal is only useful for the first links of all the paths in the network. These links do not have upstream nodes and therefore arriving vehicles are generated from a random procedure,which tries to reproduce the vehicle arrivals in the network from different demand generation locations.

For the remaining links in the network, the time calculation is simple due to the proposed approach. The vehicle may not leave a link until it is sure that it has space in the first cell of the input lane of the next link of its assigned route. So, to obtain the vehicle arrival time at its downstream link, we add –to the departure time from the final position of the upstream link– the time that the vehicle takes to cross the node and the first cell of the next link. So, Equation 5.3.26 expresses it as:

$$t_{veh}^{a}\left(x_s\right) = t_{veh}^{d}\left(x_t(previousLink)\right) + t_{crossNode} + t_{crossIniCell} \tag{5.3.26}$$

where:

$t_{crossNode}$ is the penalty time for transferring the node.

In conclusion, in this section we have developed a multiclass multilane DNL model. The model has been developed in different stages. In the first one, we try to solve only the multiclass vehicle problem by disregarding the possibility of having lanes inside the network links. In order to solve this problem, we have proposed two different scenarios: the first one considers only one isolated link, and the second introduces the effects of the intersection and its management in the dynamic loading process. To solve the multiclass problem, we have considered the different effective lengths of each class of vehicles in order to control the link capacity during the process.

After solving the multiclass case in the completed scenario (link-node-link), we have extended the model to the multilane case. With the introduction of lanes inside the network links, we have found the unstudied lane change situation, which we want to reproduce because we consider it to be one of the most important causes of traffic congestion. To solve the multiclass multilane model, we have used the same two scenarios and propose some hypotheses about the lane changing process.

Finally, we summarize in Equation 5.3.27 the proposed multiclass multilane DNL model, with the following definitions time calculation functions explained above.

$$\begin{aligned} t^{a}_{veh}\left(x_s\right) &= t^{d}_{veh}\left(x_t(previousLink)\right) + t_{crossNode} + t_{crossIniCell} \\ t^{d}_{veh}\left(x_s\right) &= \max\left\{t^{a}_{veh}\left(x_s\right), t_{freeLink} + T_{veh}\right\} \\ t^{a}_{veh}\left(x_t\right) &= \max\left\{t^{d}_{veh}\left(x_s\right) + t_{crossLink} + t_{LCH}, t^{d}_{l^{c_{out}}_{veh}}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh}\right\} \\ t^{d}_{veh}\left(x_t\right) &= \max\left\{t^{a}_{veh}\left(x_t\right), t_{canPass} + T_{veh}\right\} \end{aligned} \tag{5.3.27}$$

5.4 Model Specifications and Design

In the previous Section 5.3, a mesoscopic traffic flow simulation model is developed to be used as the DNL component of our previously presented DTA method. This simulation model is solved using an algorithm based on discrete events.

In general, there are two primary approaches to building simulation models: time-step and event-based models. These two paradigms are fundamentally different approaches due to how they manage time. In a time-step model, time is the independent variable; while in an event-based model, time is a dependent variable. Thus:

- Time-step models: the state variables of all vehicles are updated at the end of each discrete-time interval, based only on the known state variables at the previous time-step. The conventional approach is to make this time-step equal to a common divisor of all driver response times. Typically, this step is between 0.1 and 1.0 seconds.

- Event-based models: the state variables of the vehicles are updated only at the time which any of its relevant input data changes. An event is generated at a specific point in time (continuous) to reflect a change in the information (stimulus) and act accordingly.

One of the motivations for adopting an event-based simulator in our DNL component is that it can potentially be much more computationally efficient than a time-step model. The computation time is particularly important in the context of a DTA, which usually needs some iterations of the simulation model in the process in order to achieve the DUE.

However, the main motivation is that we must take into account that the other option, the time-step paradigm, considers time as an independent variable. As we explained in Section 5.3, our mesoscopic simulation model considers the flow propagation process vehicle to vehicle, and it is formulated from the relationship: space-dependent time ($t(x)$). So, in this case, it is more suitable to use the event-based paradigm,which coincides with the idea of considering time as a dependent variable.

5.4.1 Specifications

The developed mesoscopic traffic flow simulation model adopts an event-based approach, in which the simulation clocks move between events and there is no fixed time step. Events can be known in advance to occur at a particular time in the simulation or they can be dynamically added to the event list during the simulation.

During the development of our DNL model, we designed and implemented the events related with the different times when the vehicle arrives at or departs from the abovementioned specific link positions (initial and end). Also, we need some other events directly related with the traffic simulation (start or end simulation), and with the entry and exit of a vehicle in the network. Finally, during the development, we need complementary events in order to facilitate the implementation of the process and to improve the computational time results.

Specifically, our mesoscopic simulation model includes the following list of events:

- StartSimulation
- ChangeInterval
- StatsCollector
- EnterVehicle
- VehicleArrivesLinkOrigin

- CheckNode
- VehicleLeavesLinkOrigin
- VehicleCheckDepartureFromLinkOrigin
- VehicleArrivesLinkEnd
- VehicleTriesToLeaveLinkEnd
- VehicleLeavesLinkEnd
- ExitVehicle
- EndSimulation

These events emulate the movements of the vehicles (the entities of the simulation model) through the lanes by using the previously developed model defined in Section 5.3. All these events have an associated time that is used to sort the event list, which is the time when the event has to be process. Moreover, we need some other attribute to sort this list when there are two events with the same associated time. With this aim, each event has an associated priority in order to solve this problem.

This priority attribute can be assigned in two ways: randomly or fixed. The process requires that some events have a fixed priority to ensure that they will be processed before another event. For example, events related to statistics collection are going to be treated before events related to changing the interval. Also, this priority is used to facilitate the auto-management that nodes perform when blocking situations arise. On the other hand, when the process does not require a certain priority, we assign a random priority to the events in order to sort the event list (as arbitrarily as possible) when two events have the same associated time.

The event time of each event must be calculated in the same moment that each event is generated. For some events, the calculation of this time explicitly follows the time definition functions presented in the previous Section 5.3.

In the following, we briefly summarize each event and the calculation of its event times.

StartSimulation

This event is added to the event list just when the simulation starts. It is responsible for adding to the event list events that can be known in advance to occur at a particular time during the simulation. Thus, it adds to the event list as many ChangeInterval events as demand intervals are considered in the problem. Moreover, the StartSimulation event adds events related to statistics collection. The number of StatsCollector events added depends on the length of the statistical collector interval previously defined by the user.

Its event time is the same time as when the mesoscopic simulation process starts, i.e., it is zero.

ChangeInterval

As we commented in Chapter 4, one of the inputs of the DTA problem (and consequently, of the DNL component) is a set of time-dependent OD matrices. It is important to note that each matrix can contain different OD pairs, because the possibility that some demand exists only during certain intervals. So, at each demand interval, it is necessary to know, not only the demand, but also the OD pairs that have demand for this interval.

This event is responsible for preparing the demand that corresponds to each of the intervals of the simulation. After the processing of a ChangeInterval event, an OD pair list with the corresponding trips is prepared. In the particular case of a multi-class demand, we prepare one different “virtual” OD pair list for each class.

Its event time coincides with the time when the demand interval of the simulation starts, which is a required input of the global process.

StatsCollector

This event is responsible for collecting the statistics of the process. The DTA process needs some information about links (travel times) in order to be used in other components of the procedure, for example, in the RGap convergence calculation or in the reassignment flow method. So, we need to perform this collection at the end of each interval.

On the other hand, the simulation must be able to collect statistics of the process from time to time. The decision of the length from one collection to the next is taken by the user and it may be different from the demand interval length.

So, its event time is calculated according to the length of the demand interval (the duration of the OD matrices) and the length of the user-defined recollection interval.

EnterVehicle

The EnterVehicle event is responsible for generating a new vehicle (entity) in the network (system). It is related with the corresponding demand for the interval when the vehicle is being generated, and with the class to which the vehicle belongs. Also, this event must generate the next event, which emulates the arrival of the vehicle to the origin of the first link in the assigned route.

Its event time is calculated by following an exponential distribution with parameters equal to the interval demand / OD trips for each class.

VehicleArrivesLinkOrigin

This event is responsible for performing all the actions that a vehicle needs to do or that it triggers when arriving at the origin of a certain link through a certain input lane. So, this event calculates the departure lane of the vehicle from this link and its leader in the departure lane. Moreover, this event alerts the node, because the vehicle has freed its space in its own node by generating a CheckNode event.

If the vehicle can go into the link, the VehicleArrivesLinkOrigin event generates an event that emulates the departure of this vehicle from the origin of the link.

Finally, this event updates the statistics collected at arrival.

Its event time is calculated by following Equation 5.3.26, which is previously defined in the model presented in Section 5.3.2.2:

$$t_{veh}^{a}\left(x_{s}\right)=t_{veh}^{d}\left(x_{t}(previousLink)\right)+t_{crossNode}+t_{crossIniCell} \quad (5.3.26)$$

CheckNode

This event is responsible for managing movements through the node. As we explained before, some different traffic situations can block the passage of a vehicle from one link to the next in its assigned path. The situations taken into account in the proposed model are traffic lights in a red phase, a vehicle occupying the initial cell of the next link in the path, or a vehicle occupying the node space.

When some of these situations change, the node can manage the new situation by generating a TriesToLeaveLinkEnd event.

Its event time is equal to the time when the situation that blocked the traffic flow dynamics changes.

VehicleLeavesLinkOrigin

This event simulates the actions triggered by the departure of the vehicle from the origin of the link. When the vehicle departs from its input lane, it needs to know if it can arrive at the end of this link in its departure lane. So, it is necessary to take into account if the lane change can be performed. This event is responsible for generating the arrival time at the end of the link. Moreover, this event alerts the node that the vehicle has freed its space in the link by generating a CheckNode event.

In this case, perhaps the event must alert another vehicle that is waiting for this vehicle to progress into the link after it. So, a VehicleCheckDeparture-FromLinkOrigin is generated for this waiting vehicle.

Its event time is calculated following Equation 5.3.20, which is previously defined in the model presented in Section 5.3.2.2:

$$t^{d}_{veh}\left(x_s\right) = \max\left\{t^{a}_{veh}\left(x_s\right), t_{freeLink} + T_{veh}\right\} \quad (5.3.20)$$

VehicleCheckDepartureFromLinkOrigin

This event revises if some vehicle that is waiting to progress inside the link, can perform this movement or not. The vehicle is waiting the departure of this leader (in the departure lane) from the initial link position. If the vehicle can advance, the event generates an event that emulates its departure from the link origin.

Its event time is equal to the time when the situation that blocked the progress of the vehicle through the link changes.

VehicleArrivesLinkEnd

This event is responsible for manage the exit of the vehicle from the link through its departure lane. If the vehicle can continue its assigned route, VehicleArrives-LinkEnd event generates an event that reproduces the departure of the vehicle from the end of the link. If the vehicle cannot pass, the VehicleArrivesLinkEnd event generates an event that emulates its waiting until the blocking situation changes.

Its event time is calculated following Equation 5.3.23, which is previously defined in the model presented in Section 5.3.2.2:

$$t_{veh}^{a}\left(x_t\right) = \max\left\{t_{veh}^{d}\left(x_s\right) + t_{crossLink} + t_{LCH}, t_{l_{veh}^{c_{out}}}^{d}\left(x_t\right) + \frac{L_{l_{veh}}}{V_l} + T_{veh}\right\} \quad (5.3.23)$$

VehicleTriesToLeaveLinkEnd

This event replicates the actions of the VehicleArrivesLinkEnd event with a few different implementation details. In this case, when the situation that blocks the passage of the vehicles through the node changes, the event tries to know if one of the waiting vehicles can pass. If the vehicle can continue its assigned route, VehicleTriesToLeaveLinkEnd event generates an event that reproduces the departure of the vehicle from the end of the link. If the vehicle cannot pass, the VehicleTriesToLeaveLinkEnd event generates an event that emulates its waiting until the blocking situation changes. Moreover, in this case, the event generates a CheckNode event to make it possible for other waiting vehicles to cross the node.

Its event time is equal to the time when the situation that blocked the passage of the vehicle through the node changes.

VehicleLeavesLinkEnd

This event is responsible for emulating all the actions that a vehicle must take into account when it leaves the link in its departure lane. If the link is the last link of its assigned route, this event generates the ExitVehicle event. If not, the event must generate a VehicleArrivesLinkOrigin event to emulate the arrival of the vehicle to the next link in its assigned route.

In this case, it is important that the event check if some other vehicle is waiting for space in the lane where the first vehicle has been removed. If the vehicle is waiting at the origin of the link, a CheckDepartureFromLinkOrigin event must be generated. If the vehicle is waiting to arrive to the end of the link, a VehicleArrivesLinkEnd event must be generated.

Finally, this event updates the statistics collected at departure.

Its event time is calculated by following Equation 5.3.25, which is previously defined in the model presented in Section 5.3.2.2:

$$t^{d}_{veh}\left(x_t\right) = \max\left\{t^{a}_{veh}\left(x_t\right), t_{canPass} + T_{veh}\right\} \quad (5.3.25)$$

ExitVehicle

This event only removes the vehicle (entity) from the network (system).

Its event time is the same as the time when the event is generated, i.e., it coincides with the time when the vehicle leaves the end of the last link of its assigned route.

EndSimulation

This event is responsible for treating the collected data before it is used in other components of the DTA process, or before it is shown as a result of the DNL method.

Its event time is equal to the total duration of the simulation, which is the sum of the duration of all demand intervals.

5.4.2 Model Design

The design of the proposed DNL model is based on the abovementioned events.

The process starts with two events in the event list: StartSimulation and End-Simulation events. So, first of all, we process the StartSimulation event,which is responsible for generating as many ChangeInterval events as OD matrices are considered in the demand of the DNL problem. The first of these ChangeInterval events starts the simulation process by generating an EnterVehicle event for each of the OD pairs that appear in the first OD matrix demand.

All the vehicles departing from the different origins of the network proceed with the same logic, which is illustrated in Figure 5.4.1 and is explained as follows.

When a vehicle enters the network, another EnterVehicle event is generated from the same OD pair. Moreover, a VehicleArrivesLinkOrigin event is generated to emulate the arrival of the vehicle at the first link of its assigned route. When this event is processed, it is time to notify the node that the vehicle has freed its space, so a CheckNode event is generated. It is also time to generate the corresponding VehicleLeavesLinkOrigin event. If the vehicle can leave the origin of the link and it can advance through the link, then a VehicleArrivesLinkEnd event is generated. In this case, perhaps the vehicle must alert another vehicle that is waiting for this vehicle to enter the link after it (because the first one is the leader in the departure lane of the waiting vehicle). So, a VehicleCheckDepartureFromLinkOrigin is generated for this waiting vehicle. Moreover, the vehicle must inform the node that the cell at the beginning of the link is free through a CheckNode event. After this, the processing of the VehicleArrivesLinkEnd event is a bit more complicated. If the vehicle can exit the link, a VehicleLeavesLinkEnd event is generated. But, if the vehicle cannot pass through the node or does not have space in the following link, it needs to wait until the situation changes. In this case, a VehicleTriesToLeaveLinkEnd is generated in order to revise if some of these situations change, in which case a VehicleLeavesLinkEnd may be generated. Finally, when the vehicle leaves the end of the link a VehicleArrivesLinkOrigin event is generated to emulate the arrival of the vehicle to the next link of its assigned route. If the current link is the last link in the assigned route, then instead of the VehicleArrives-LinkOrigin event, an ExitVehicle event is generated. In both cases, if there is a vehicle waiting for the departure of this vehicle from the end of the link, a VehicleArrivesLinkEnd is generated for it. If a vehicle is waiting at the origin of the link for space in this lane, a CheckDepartureFromLinkOrigin event must

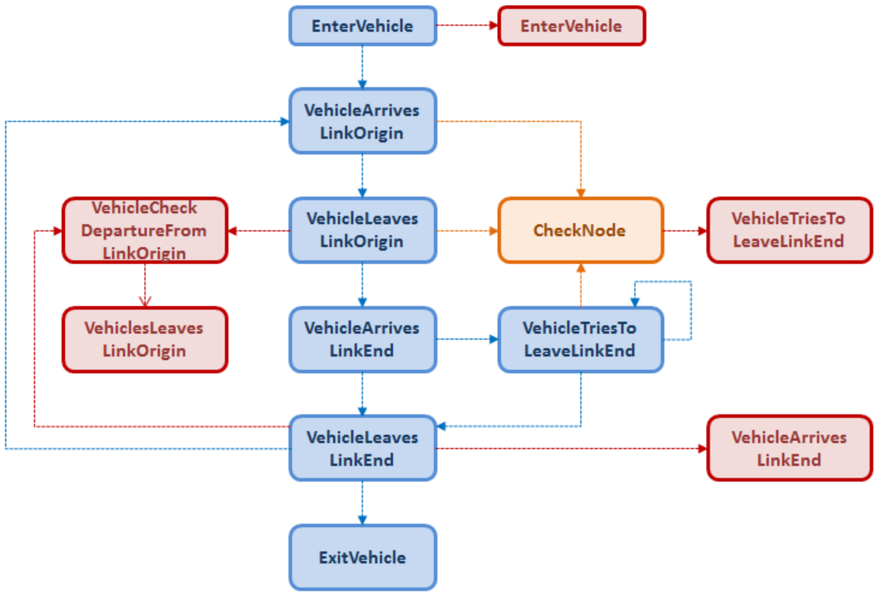

EnterVehicle
EnterVehicle
VehicleArrives LinkOrigin
VehicleCheck DepartureFrom LinkOrigin
VehicleLeaves LinkOrigin
CheckNode
VehicleTriesTo LeaveLinkEnd
VehiclesLeaves LinkOrigin
VehicleArrives LinkEnd
VehicleTriesTo LeaveLinkEnd
VehicleLeaves LinkEnd
VehicleArrives LinkEnd
ExitVehicle

Figure 5.4.1: Event conceptual model of the proposed mesoscopic traffic simulation.

be generated. Of course, the ExitVehicle event emulates the exit of the vehicle from the network, so this entity (vehicle) is removed from the system.

When some of the blocking situations change, a CheckNode event is generated. In this case, the node needs to revise if there is some vehicle waiting for this reason. It generates a VehicleTriesToLeaveLinkEnd event in order to allow these waiting vehicles to advance or not.

When it is time to process a VehicleCheckDepartureFromLinkOrigin event for a certain vehicle, the vehicle can advance into the link if the previous conditions have changed. So a VehicleLeavesLinkOrigin event is generated for this vehicle.

Link Number	1	2	3	4	5	6	7	8
Length (meters)	110	100	110	130	130	290	290	130
Free Flow Speed (Km/h)	50	50	50	50	50	50	50	50
Number of lanes	3	3	2	2	2	2	3	2

Table 5.1: Input data of each link of the network.

5.5 Computational Experiences

In this section the proposed DNL model is evaluated using laboratory tests. In order to demonstrate the proposed multilane multiclass model, the following computational experiments are conducted.

First, the model is applied on a small network and the corresponding fundamental diagrams are examined for basic correctness. Then, we study the propagation of the shockwaves arising from an incident. To achieve that, the model is run over a small freeway corridor where we simulate an incident during certain time intervals. Finally, the model is tested on two real networks. In both cases, the obtained results are compared with those obtained from a microsimulator.

5.5.1 Fundamental Diagram Accomplishment

In this section, the performance of the proposed model is validated over a sample network. This exercise tries to experimentally investigate the accomplishment of the fundamental diagram by the presented mesoscopic simulation model. The traffic fundamental diagram describes the relations among the main macroscopic traffic variables: flow, speed and density.

With this aim, we execute our model over a small test network. Then, the results are collected for density, flow and speed. Finally, we graphically compare these simulation results with a theoretical macroscopic proposal.

5.5.1.1 Test Network

The proposed sample network is shown in Figure 5.5.1. It consists on eight links, three nodes and five centroids. Table 5.1 shows the input data of each link of the network.

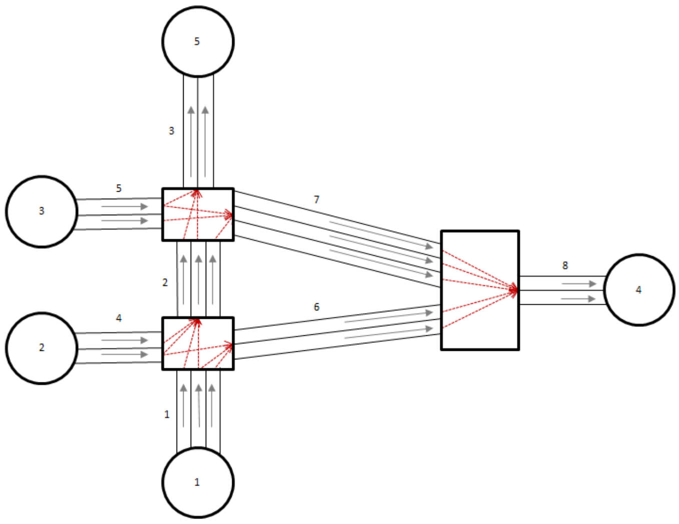


Figure 5.5.1: Sample network.

In order to validate the multilane characteristic of the proposed DNL model, we consider a longitudinal discretization of the links. Moreover, we define a set of turns for each node. As we can see in Figure 5.5.1, not all the feasible turns are defined. In some cases, only some lanes are allowed to be used by vehicles to go from certain link to its downstream link. For example, only the vehicles situated in the right lane of Link 2 can turn to Link 7. So, we trigger lane changes when vehicles enter into Link 2 through the other two lanes, and they want to go to Link 7.

Centroids 1, 2 and 3 generate demand, while centroids 4 and 5 only can receive demand. All the feasible combinations are used, so, we consider the following six OD pairs: (1,4), (1,5), (2,4), (2,5), (3,4) and (3,6). Additionally, we consider all the possible paths among the different origins and destinations (one or two paths, depending on the OD pair).

5.5.1.2 Experiment Design

The proposed experiment investigates the relationships among the values of the main macroscopic traffic variables simulated by the proposed DNL model.

In order to obtain a wide range of simulated values, the demand among the feasible OD pairs is varied over different runs of the simulation. We execute the proposed mesoscopic simulation model with a light demand. Then, we gradually increase the demand until no more vehicles can enter certain link. In order to induce more or less congestion in the different links in the network, we play with different demands. Some demands take into account only one OD pair, while other demands combine some OD pairs that share links belonging to their respective paths.

A 2-hour long simulation determines the average link measurements for each 3-min interval. Two classes of vehicles have been considered: light vehicles with an effective length of 5 meters (90% of the total demand) and reaction time of 0.5 seconds, and heavy vehicles with an effective length of 9 meters (10% of the total demand) and reaction time of 0.75 seconds.

In the following, we show a macroscopic variable relationship of the proposed model for some links in the sample network. In order to graphically analyze the results, we superimpose onto these plots the corresponding theoretical relationships. In this case, these relationships are derived from the Fundamental Traffic Equation (see Equation 5.5.1) and from the speed-density relationship proposed by Underwood (1961) shown in Equation 5.5.2.

$$q = k \cdot v \tag{5.5.1}$$

$$v = v_f \cdot \exp(-k/k_m) \tag{5.5.2}$$

where:

q	Flow (veh/h/lane).
k	Density (veh/Km/lane).
v	Average speed (Km/h).
v_f	Free-flow speed (Km/h).
k_m	Average density (veh/Km/lane).

5.5.1.3 Computational Results

Figure 5.5.2 shows the flow-density relationship of the proposed model for some links in the sample network. It is important to note that this relationship can take a different curvilinear shape. This form depends on local conditions of the link: the demand that its upstream node can supply and the demand that its downstream node is able to drawn in. In this case, links 2 and 7 are intermediate links, while links 4 and 5 are input links on the network.

It is important to note that some other links of the network (Link 3, Link 6 and Link 8) do not achieve the same density level as the remaining links in the network. This phenomenon is a consequence of the local conditions of these links in the network. In these cases, the demand that their upstream nodes can supply is less than the demand rate needed to achieve saturation. So, in these cases, we only can reconstruct the first part of the diagram. Because this zone only represents the flow under light density conditions, we do not consider meaningful to show their corresponding graphics.

In conclusion, the obtained test results experimentally show that the presented model is able to reproduce the fundamental diagram that describes the relationships among the main macroscopic traffic variables.

5.5.2 Study of the Shockwaves Propagation

In this section we want to study the propagation of shockwaves arising when a blocking incident occurs. The objective is to complement the fundamental diagram in demonstrating the basic traffic performance on links in the proposed mesoscopic model.

Shockwaves in traffic are defined as discontinuities in one of the macroscopic variables (flow, density or speed) in the space-time domain. There is a transition zone between two traffic states that moves through a traffic environment like a propagation wave.

In order to attain that, the proposed model is run over a network that mimics a small freeway corridor where we simulate an incident during a certain time interval.

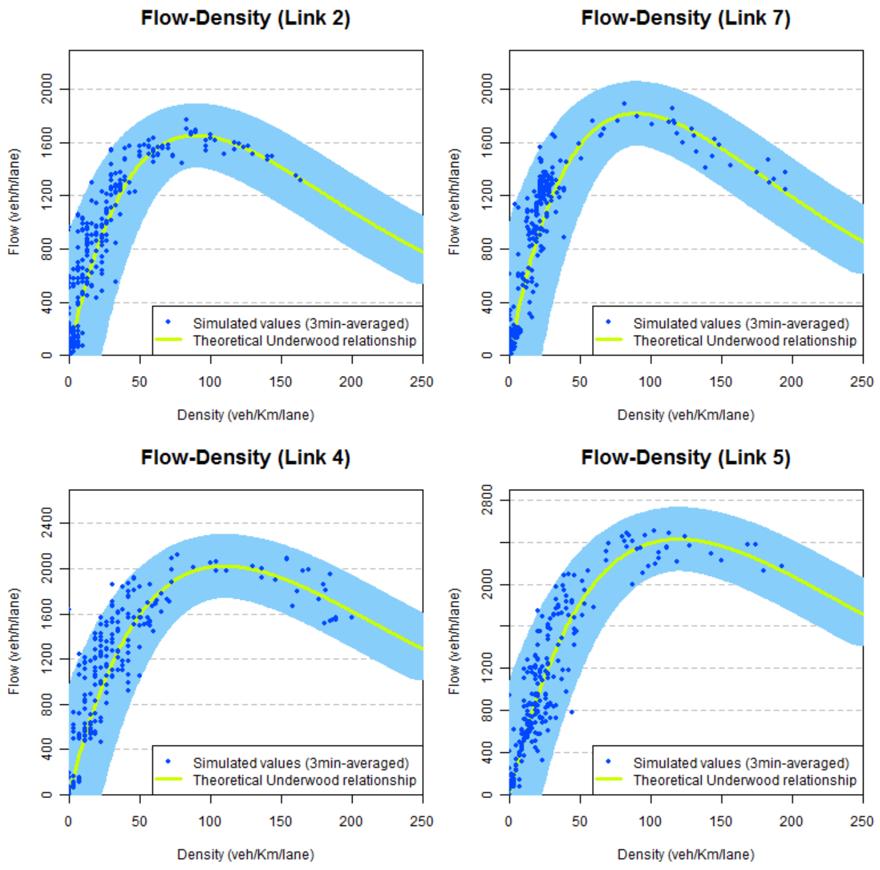


Figure 5.5.2: Graphics of flow vs. density.

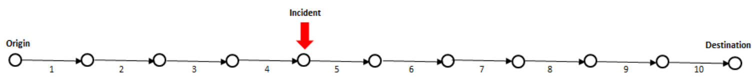


Figure 5.5.3: Network to study the shockwave phenomenon.

5.5.2.1 Test Network

The proposed test network (depicted in Figure 5.5.3) is a simple network consisting of 10 consecutive links. It simulates a small freeway corridor. The link segments are each 400 meters long, have two lanes, and a free flow speed of 80 Km/h.

Demand is represented by four 15-min matrices used to perform a one-hour simulation. The total number of vehicles is 3,000. Two vehicle classes are considered with an effective length of 4 and 6 meters, respectively.

5.5.2.2 Experiment Design

The proposed experiment studies if the implemented model reproduces the propagation of the congestion properly. In order to achieve that, we create an incident in the proposed small freeway network and then we run our model over this scenario.

As Figure 5.5.3 shows, a blocking incident is created in the node downstream from Link 4. The incident starts 20 minutes after the beginning of the simulation and its duration is 10 minutes. Both lanes of the network links are affected by the incident, and no vehicle can pass as long as it exists.

In order to obtain the cumulative flow of all the links in the network, we execute a one-hour long simulation. The proposed mesoscopic simulation determines the number of vehicles that have entered the link (cumulative inflow) for each 1-min interval. Also, we collect the number of vehicles that have exited each of the links in the network (cumulative outflow) during each minute of the simulation.

In the following, both cumulative flows over time are plotted for all the links in the test network.

5.5.2.3 Computational Results

To illustrate the propagation of the shockwave that arises from the generated incident, the cumulative inflow and outflow plots are shown in Figure 5.5.4. The slope of the lines are the flow rates and the distance between the lines represents the density of the links.

First of all, we analyze the cumulative outflow plot, where each line represents the cumulative number of vehicles that have one exit link in the network. In this case, Figure 5.5.4 shows how the incident starts on the downstream node of Link 4, 20 minutes after the simulation begins. At $time = 1200$ seconds, the slope of the line corresponding to Link 4 presents a slope equal to zero, so no vehicles are exiting from this link because the node is blocked by the incident. Also, we can observe how the incident backs up to Link 3 and Link 2. And also, how Links 5, 6, 7, 8 , 9 and 10 progressively tend to this zero-slope (during the next two minutes of the simulation). At $time = 1800$ seconds, the blocked situation is removed from the scenario. At this time, we can observe that the slope of the line corresponding to Link 4 becomes different to zero, so vehicles exit from this link. This behaviour is propagated via Links 3 and 2. Some seconds later, vehicles arrive at the downstream links and their slopes become different to zero. Around $t = 2100$ seconds traffic situation is completely restored. Link 1 is unaffected by the incident.

Secondly, we analyze the cumulative inflow plot. In this case, each line represents the cumulative number of vehicles that have entered one link in the test network. Figure 5.5.4 shows how the incident starts on the upstream node of Link 5. So, at $time = 1200$ seconds, the line corresponding to Link 5 presents a zero-slope, i.e., no vehicles are entering this link because its upstream node is blocked by the incident. Also, we can observe how the incident backs up to Link 4 and 3, and how Links 6, 7, 8, 9 and 10 progressively tend to this slope of zero. At time $t = 1800$ seconds, the blocked situation is removed from the network. At this time, we can observe that the slope of the line corresponding to Link 5 becomes different to zero, so the vehicles enter this link. This behaviour is propagated to Links 4 and 3. Around $t = 2100$ seconds the incident is completely clear and the initial traffic situation is restored. In this case, Links 1 and 2 are unaffected by the incident.

This simple experiment graphically shows that the proposed model performs properly when changes in traffic conditions occur. We observe that the model respects the propagation of the congestion, ensuring correct temporal and spatial

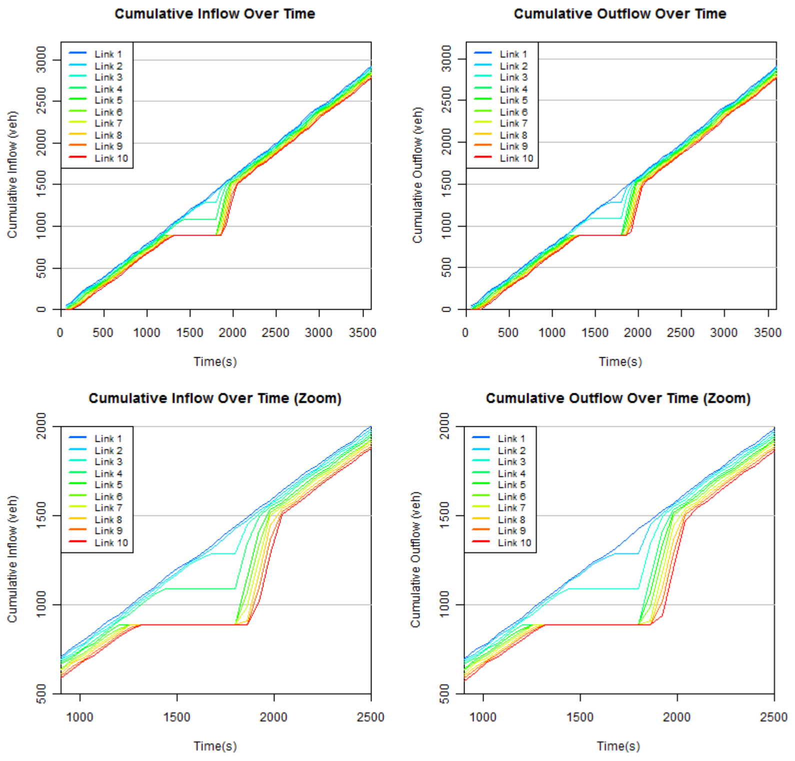


Figure 5.5.4: Cumulative inflow/outflow of all the links in the test network with an incident.

location of the congestion at the link level. However, it is important to note here the special behaviour of our model in respect to the propagation of the queue dissipation shockwave.

In our mesoscopic model space is considered discrete: the link length. As we mentioned before, we lose continuous control of the vehicle location in the network. However, we can know how much free space we have at each lane of the link, without knowing where it is exactly located in the lane. It is for this reason that we can observe in both plots the following phenomenon.

See for example the cumulative outflow case. Just before the blocked incident is removed from the network, we can observe that the first affected link is totally full. When the situation changes, one vehicle can exit this link, setting free its own effective length in this lane. In this case, our model immediately detects that this lane has free space and alerts the vehicle that is waiting in the first cell of that lane. If there is enough space, the vehicle leaves the first cell and goes into the link. Consequently, because this vehicle sets free the first cell, the model alerts the vehicle that it is waiting in the upstream link to go into this link through this lane. It is for this reason that we can observe that Links 4, 3 and 2 start the process of dissipation practically at the same time.

So, the queue dissipation shockwave propagates faster through our proposed model than it does in a real situation. However, we consider that this special behaviour it is not relevant in overall model performance. The vehicles that are waiting in the first cell can advance faster into the link through its corresponding lane, but they cannot exit the link until their leaders do it. So, at the link level, the density maintains the correspondence with the density of a real situation.

5.5.3 Proposed model vs Microsimulation

Finally, in this section the model is applied to two real networks. The aim of this exercise is to validate the correctness of the proposed DNL model. Our model is less detailed than microscopic models that are considered the most realistic simulation tools for emulating the flow of individual vehicles. So, in order to experimentally investigate the performance of the proposed multilane multiclass model, we compare the results obtained through a microsimulator with the results obtained through the proposed model. Aimsun is chosen as the benchmark microsimulator.

We begin by briefly describing the Aimsun microsimulation model and its main characteristics. Then, we introduce some goodness-of-fit measures which we use to evaluate the correctness of the model. Then, we presents the test networks. After this, we proceed to outline the experiments performed for both network scenarios. And finally, we analyze and discuss the obtained results taking as a reference different graphics and the goodness-of-fit measures we presented previously.

5.5.3.1 Aimsun

As we explained in Section 3.2.1, Aimsun is a fully integrated suite of traffic and transportation analysis tools, developed by a research group at the Technical University of Catalonia and led by professor Jaume Barceló (1986). It can be used for transport planning, microscopic simulation and dynamic traffic assignment, among others.

Our aim is to validate the correctness of our proposed model. Microscopic traffic simulators are the simulation tools that most realistically emulate the flow of individual vehicles in a road network. This is why we decided to compare the results obtained through the proposed model with the results obtained through a microscopic traffic simulator. So, in this section, we focus on the use of the microscopic component of Aimsun.

Most of the currently existing microscopic traffic simulators rely on the family of car-following, lane-changing and gap-acceptance models to model vehicle behaviour. In Aimsun, car-following and lane-changing models have evolved from the seminal model by Gipps (1981). Aimsun's implementation of the car-following model took into account the additional constraints on the braking capabilities of vehicles. They are imposed in the classic safe-to-stop-distance hypothesis carried out by Mahut (1999).

The lane-changing process is modeled as a decision-making process that emulates the behaviour of the drivers when they are analyzing the necessity of the lane change, the desirability of the lane change and the feasibility conditions for the lane change. A lane change also depends on the location of the vehicle on the road network. In order to achieve a more accurate representation of the behaviour of the drivers in the lane changing decision process, three different zones inside a section are considered. Each of these zones corresponds to a different lane-changing motivation. The distance up to the end of the section characterizes these zones and the next turning point.

The Aimsun microscopic simulation assigns vehicles to routes according to a route choice model. The vehicles follow paths from their origin in the network to their destinations. So, the first step in Aimsun's simulation process is to assign a path to each vehicle when it enters the network. In the Aimsun implementation the candidate paths can be of different types: user-defined paths or calculated shortest path trees.

A vehicle of a certain vehicle type traveling from origin to destination can choose one path according to a discrete choice model from the set of alternative paths. Route choice functions represent implicitly a model of user behavior and the most likely criteria employed by the user to decide between alternative routes: perceived travel times, route length, expected traffic conditions along the route, etc. The most used route choice functions in transportation analysis are those based on the discrete choice theory, i.e., Logit functions that assign a probability to each alternative route between each origin-destination pair, depending on the difference in the perceived utilities. The inability of the Logit function to distinguish between two alternative routes when there is a high degree of overlapping is the main reason for implementation of the C-Logit model (Cascetta et al. (1996); Ben-Akiva and Bierlaire (1999)) in Aimsun. As we will explain in Section 6.1, this model calculates the probability of each alternative path belonging to the set of available paths connecting an O/D pair, taking into account the degree of overlapping through a commonality factor for each path.

The traffic condition to be simulated could be defined by an OD matrix, which gives the number of trips from every origin centroid to every destination centroid, for each time slice and for each vehicle type. Vehicles are generated at each origin centroid as input into the network. Then, vehicles are distributed along the network following the shortest path between origin and destination centroids.

Aimsun defines some parameters by vehicle type. The attributes that characterize a vehicle type are: name, length (m), width (m), maximum desired speed (Km/h), maximum acceleration (m/s^2), maximum deceleration (m/s^2), speed acceptance, minimum distance between vehicles (m), maximum give-way time, percentage of guided vehicles, guidance acceptance parameter ($0 <= \lambda <= 1$), fuel consumption, and pollution emission. Moreover, Aimsun defines some parameters at the experimental level: reaction time (s), look-ahead distance variability (percentage), or statistical distribution of vehicle arrivals.

In this dissertation we use the following version of the proposed microsimulator: Aimsun v7.0.

5.5.3.2 Used Goodness-of-fit Measures

In the following two goodness-of-fit measures are introduced.

Root Mean Squared Error The Root Mean Square Error (RMSE) is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modeled. In this case, we use the results obtained through the microsimulator Aimsun as the values observed from the environment. So, we use RMSE to measure the difference between values obtained by two different models: the microscopic simulator Aimsun and the proposed mesoscopic simulation. The RMSE serves to aggregate these individual differences into a single measure.

The RMSE of our model, with respect to the obtained Aimsun value for the corresponding variable, is defined as the square root of the mean squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (X_{Aimsun,i} - X_{proposedModel,i})^2}{n}} \tag{5.5.3}$$

where:

X_{Aimsun} is the value obtained through Aimsun.

$X_{proposedModel}$ is the value obtained through the proposed model.

i is the link index.

n is the number of links in the network.

Normalized Root Mean Squared Error Non-dimensional forms of the RMSE are useful because often one wants to compare RMSE with different units. The Normalized Root Mean Squared Error (NRMSE) normalizes the RMSE to the range of observed data. In our case, we normalize RMSE to the range of obtained values through Aimsun.

$$NRMSE = \frac{RMSE}{X_{Aimsun,max} - X_{Aimsun,min}} \tag{5.5.4}$$

GEH Geoffrey E.Harvers' statistic GEH (Highways Agency, 1996) is a measurement widely accepted by practicioners because it provides an overall view which is considered more useful than the individual measurements. GEH calculates the index for each link (GEH_i).

$$GEH_i = \sqrt{\frac{2\left(X_{Aimsun,i} - X_{proposedModel,i}\right)^2}{X_{Aimsun,i} + X_{proposedModel,i}}} \tag{5.5.5}$$

If the deviation of the proposed model values with respect to the Aimsum values is smaller than 5% in at least 85% of the cases, then the proposed model is accepted.

In addition, the GEH average for all links can also be calculated. In this case, the proposed model is accepted if this value is smaller than 4%.

There is neither theory behind the method to determine the value thresholds that are purely empirical based on practice.

Theil's Indicator The use of aggregated values to validate a simulation seems contradictoryt if one takes into account that it is dynamic in nature, and thus time dependent. Consequently, other analysts propose statistical methods which account specifically for the comparison of the disaggregated time series of the values. Theil (1961) defined a set of indices aim at this goal. The first index is Theil's indicator (U) which provides a normallized measure of the relative error that smoothes out the impact of large errors.

$$U = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_{Aimsun,i} - X_{proposedModel,i}\right)^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_{Aimsun,i}\right)^2} + \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_{proposedModel,i}\right)^2}} \tag{5.5.6}$$

The index U is bounded, $0 \leq U \leq 1$, with $U = 0$ for a perfect fit between the two values. For $U \leq 0.2$, the obtained values with the proposed model can be accepted as replicating the microscopic simulated values acceptably well. For values greater that 0.2, the proposed model is rejected.

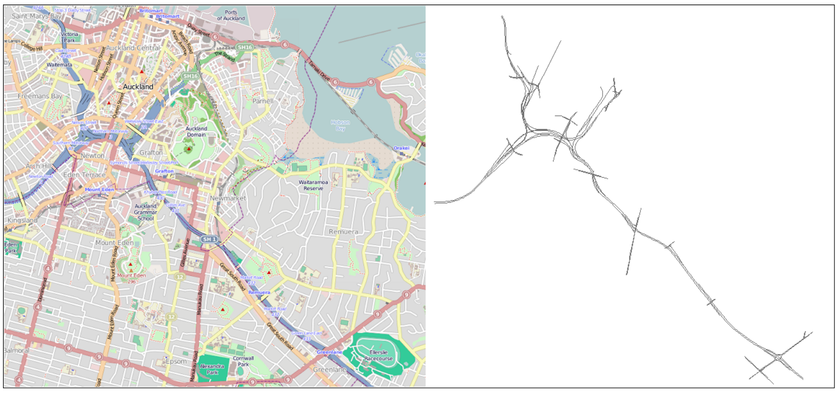


Figure 5.5.5: Proposed freeway test network.

5.5.3.3 Case Study 1: Freeway Network

Real Freeway Test Network

Figure 5.5.5 shows the case study used to test the proposed DNL model against the selected microscopic simulator (Aimsun v7.0.). It is the SH1 freeway in Auckland (New Zealand). This proposed test network consists of 119 nodes, 223 links and 276 turns. Figure 5.5.5 also gives a general overview of the exported network.

The lengths are distributed among the minimum length of 7.91m and the maximum length of 1,801.77m. The mean length is equal to 215.09m. The maximum allowed speed goes from 50 Km/h to 100Km/h, depending on the link type. Each link has its corresponding longitudinal division by lanes among 1 and 6 lanes.

The main input of the DNL, besides the network itself, is demand. In this case, we consider a multiclass demand. Table 5.2 shows the characteristics of the three vehicle classes considered: A, B and C.

In this experiment we use a real demand corresponding to the interval from 6:45 a.m. to 9:00 a.m. So, for each vehicle class we use nine 15-min matrices. A total of 27 matrices provide the origin-destination demand data for 34 zones resulting in 221 OD pairs. The total number of trips in the matrices is 69,272 corresponding with a congested scenario. Table 5.3 summarizes this demand.

Vehicle Class	Length (m)	Reaction Time (s)
A	4.4	0.75
B	6.5	0.75
C	7.7	0.75

Table 5.2: Vehicle class characteristics.

Auckland Demand (veh)				
	Vehicle Class			Total
Time Interval	A	B	C	
6:45-7:00	5,395	643	108	6,416
7:00-7:15	7,725	934	157	8,816
7:15-7:30	7,419	894	123	8,436
7:30-7:45	7,231	875	115	8,221
7:45-8:00	7,021	849	150	8,020
8:00-8:15	6,796	820	120	7,736
8:15-8:30	6,453	772	144	7,369
8:30-8:45	6,242	750	190	7,182
8:45-9:00	6,372	771	203	7,346
Total	60,654	7,308	1,310	**69,272**

Table 5.3: Demand of the Auckland case study.

Experiment Design

In the proposed experiment we run each simulation model over the previously presented freeway network. The objective is to perform a comparison between both results, so it is important to run both models with simulation experiments as close as possible. This means that we need to use the same network geometry, demand and vehicle classes.

Moreover, we decided to use the same set of paths for each OD pair. With this, we want to generate the exact same traffic situation to be simulated. If we use the same paths, we are avoiding the differences in the traffic flow variables on links which arise from a different flow distribution. In this case, we propose an experiment that uses among one and three different paths for each OD pair. These paths and their correspondings assignments are the result of a previous performed dynamic traffic assignment. Therefore, they are the paths with which we achieve the dynamic user equilibrium in Auckland network. This set of DUE paths is imported to both models respectively.

In addition, a calibration procedure was conducted for the proposed DNL model, which consisted of adjusting only the following parameters: lane change penalty (t_{LCH}) and cross node time penalty ($t_{crossNode}$). The final adjusted values for these penalty parameters were: $t_{LCH} = 0.1s$, $t_{crossNode} = 0.7s$.

After this calibration, we execute a 135 minutes long simulation through both models. In the following, all the obtained results are summarized and discussed through different goodness-fit-measures and statistical plots.

Computational Results and Benchmark

In order to obtain the computational results of the proposed experiment, we execute the simulation. It is important to note that a relevant aspect of the simulation tests are the replications. They are needed to account for the stochastic nature of the models and their results. The number of replications needed depends on the variability found between simulation runs. There exists methods to determine the number of replications that are required, however, their application is outside the scope of this thesis. In this case, in order to obtain reasonable results from the simulated values, ten replications are made for each of the simulation models. In the following, the average results over these runs are compared.

We study quantitatively the results obtained through the proposed mesoscopic simulation. We analyze link density and travel time for both models (proposed

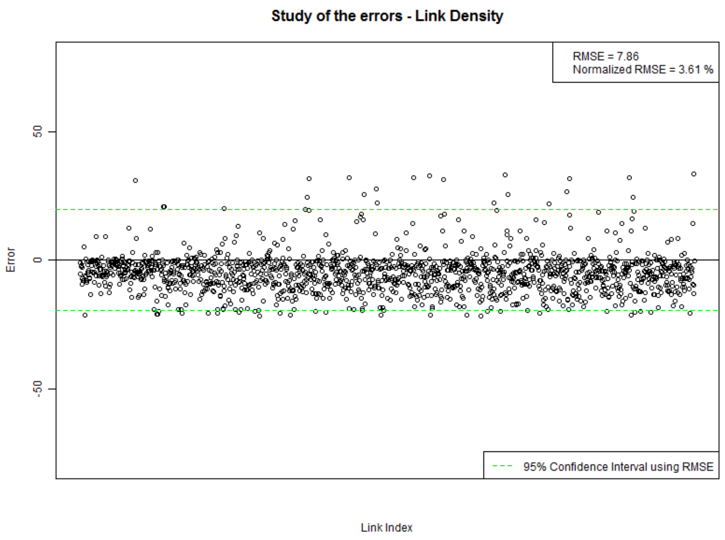


Figure 5.5.6: Residual Analysis for Link Densities (Benchmark vs. proposed model).

vs benchmark) for each link in the network at the end of each 15-min simulation interval. A residual analysis is performed in order to evaluate the correctness of the proposed model. We calculate the Root Mean Squared Error (RMSE) and the Normalized Root Mean Squared Error (NRMSE). In addition, we show a residual analysis with the 95% Confidence Interval using RMSE.

Density
Figure 5.5.6 shows the residual analysis for the obtained densities at the end of each time interval for all the links in the network. The density is expressed as hourly vehicles per lane (veh/h/lane). The RMSE has the same unit as the original measurements: veh/h/lane, and the NRMSE shows percentage. The NRMSE obtained value is 3.61% and the RMSE values is 7.86 veh/h/lane.

Travel Time
Figure 5.5.7 shows the residual analysis for the obtained travel times on each

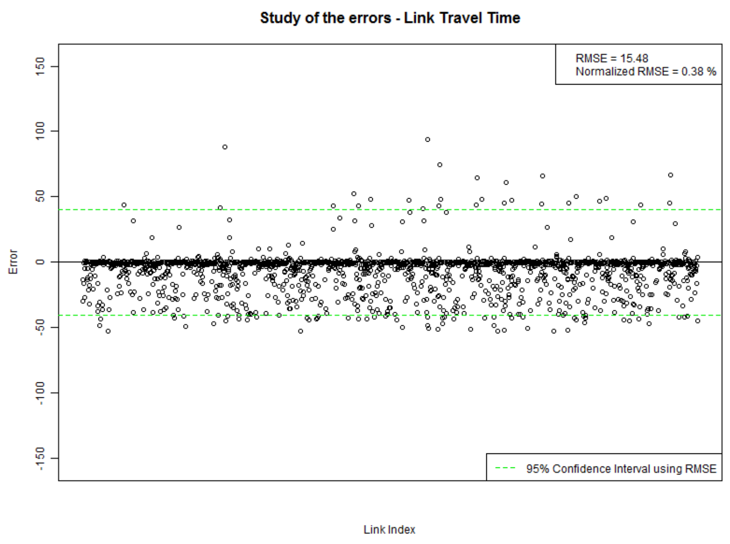


Figure 5.5.7: Residual Analysis for Link Travel Times (Benchmark vs proposed model).

link in the network at the end of each 15-min simulation interval. Travel time is expressed in seconds (sec). The obtained NRMSE value is 0.38% and the RMSE value is 15.48 seconds.

In summary, the results obtained for the performed computational experiences demonstrate the ability of the developed model to reproduce multilane multi-class traffic behavior for freeway networks.

5.5.3.4 Case Study 2: Urban Network

In the previous experiment, we showed the proper performing of our mesoscopic model when is deployed in a freeway scenario. This network type is relatively simple in terms of traffic propagation. Since the good obtained results, we now consider appropiate to increase the level of difficulty of the test network. So, we repeat the experiments of comparison with a microscopic traffic simulator but using a urban network instead. Specifically, we use a european network with roundabouts and short length links that complicate the propagation process.

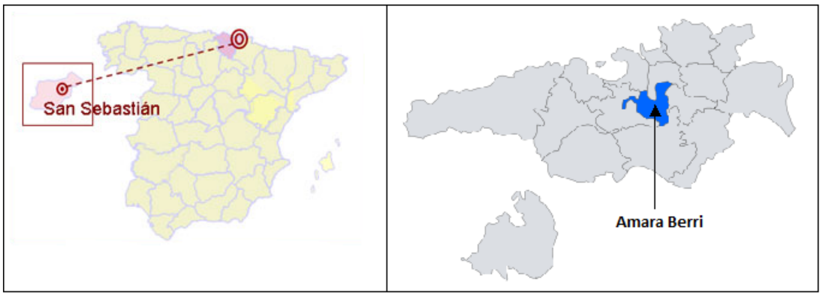


Figure 5.5.8: Location of the city of San Sebastian (Spain). Amara Berri district in the city of San Sebastian.

In these experiments, we want to make sure that we have the same high correspondence between the results obtained with the benchmark microscopic simulator and our simulator. This is very interesting considering the high difference of the number of parameters to calibrate in both cases. The option is a hundred of parameters on the micro simulator versus the only two parameters to calibrate in the proposed model. In this case, besides the goodness-of-fit measures used for the previous case, we have calculate the GEH measurement and the Theil's Indicator, and also we have added a qualitatively comparison of the results disaggregated by time intervals.

Real Urban Test Network

Figure 5.5.8 shows a road network corresponding to Amara Berri district in the city of San Sebastian (Spain).

The proposed test network consists of 76 nodes, 192 links and 301 turns. Figure 5.5.9 gives a general overview of the network.

Figure 5.5.10 shows the distribution of the lengths of the links. These lengths are distributed among the minimum length of 10.19m and the maximum length of 374.93m. The mean length is equal to 82.23m. It is important to note here that each of the links with smaller lengths belongs to one of the four roundabouts present in this network. The maximum allowed speed goes from 35 Km/h to 60Km/h, depending on the link type. Each link has its corresponding longitudinal division by lanes among 1 and 5 lanes, following the distribution shown in Figure 5.5.10.

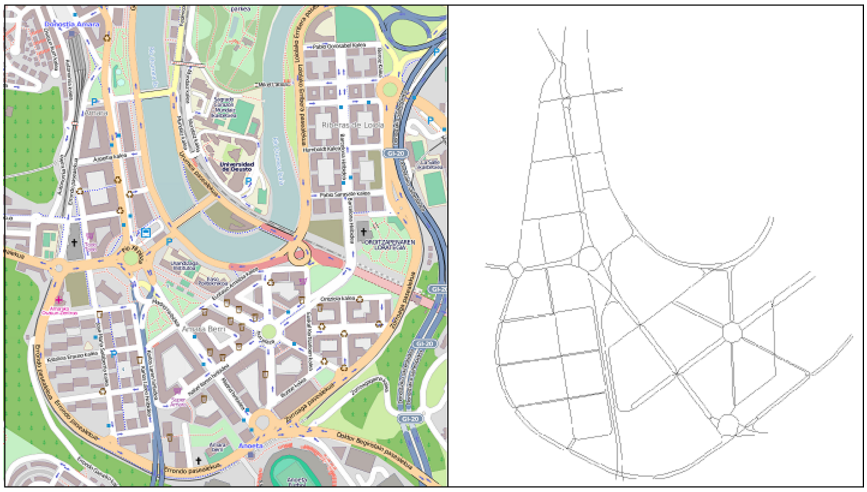


Figure 5.5.9: Proposed test network of Amara.

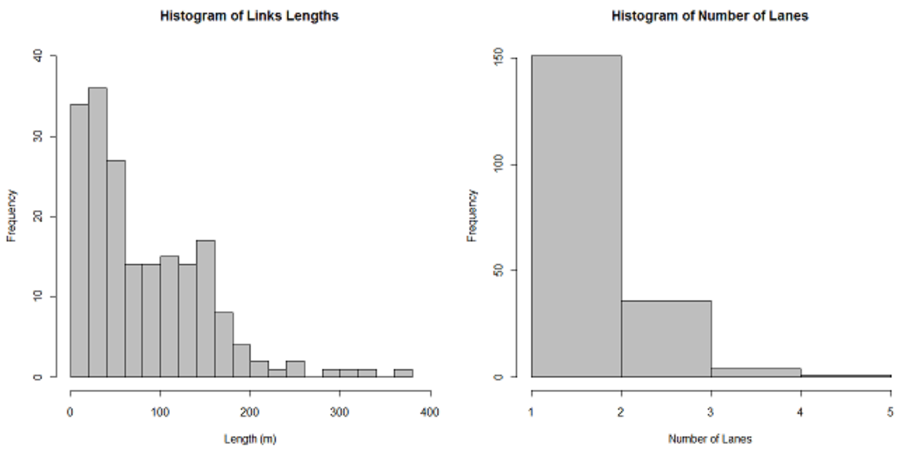


Figure 5.5.10: Distributions of lengths and number of lanes in the links of Amara network.

The main input of the DNL, besides the network itself, is demand. In this case, four 15-min matrices provide the origin-destination demand data for 13 zones, resulting in 80 OD pairs. The total number of trips in the matrices is on the order of 8,428 vehicles. Two vehicle classes are considered: 90% of the demand corresponds to light vehicles with an effective length of 5 meters, while the remaining 10% of the demand corresponds to a heavy vehicle class with an effective length of 9 meters.

Experiment Design

In the proposed experiment we run each simulation model over the previously presented urban network. The objective is to perform a comparison between both results, so it is important to run both models with simulation experiments as close as possible. This means that we need to use the same network geometry, demand and vehicle classes.

With respect to the attributes of different vehicle classes, our model distinguishes classes taking into account only two vehicle attributes: the effective length and the reaction time. As we said before, we consider two vehicle classes: light and heavy, with effective lengths of 5 and 9 meters respectively. With respect to the reaction times, we consider the times defined in the calibrated Aimsun model: light vehicle class = 0.75 seconds and heavy vehicle class = 1.5 seconds.

Moreover, we decided to use the same set of paths for each OD pair. With this, we want to generate the exact same traffic situation to be simulated. If we use the same paths, we are avoiding the differences in the traffic flow variables on links which arise from a different flow distribution. In this case, we propose an experiment that uses three different paths for each OD pair. These paths are calculated with an external shortest path (standard Dijskstra algorithm), and they are imported to both models respectively.

Moreover, a calibration procedure was conducted for the proposed DNL model, which consisted of adjusting only the following parameters: lane change penalty (t_{LCH}) and cross node time penalty ($t_{crossNode}$). The final adjusted values for these penalty parameters were: $t_{LCH} = 0.2s$, $t_{crossNode} = 0.6s$.

After this calibration, we execute a one hour long simulation through both models. In the following, all the obtained results are summarized and discussed through different goodness-fit-measures and statistical plots.

Computational Results and Benchmark

In order to obtain the computational results of the proposed experiment, we execute a one-hour long simulation. In this case, as in the previous case study, ten replications are performed for each of the simulation models to obtain reasonable results from the simulated values. In the following, the average results over these runs are compared.

First of all, Figures 5.5.11 and 5.5.12 show the link density at the end of each 15-min interval of the simulation. For an easy visual comparison, we show the results obtained through the proposed model next to those through Aimsun for the same simulation interval. We can see four simulation intervals which coincide with the previously defined demand intervals.

The results obtained for both models (proposed and benchmark) are very similar, except for the congestion caused by roundabouts. The proposed model overestimates congestion at the roundabouts, causing significant differences in some adjacent links compared with the results obtained through Aimsun.

Secondly, we show quantitatively the results obtained through the proposed mesoscopic simulation. We analyze density, travel time and vehicles per link for each link in the network for each interval.

First, in Table 5.4 we show the obtained results for the GEH measurements and the Theil's indicator for each of the measured variables. We can observe that the deviations of the proposed model values with respect to the Aimsum values for the three measured variables are smaller than 5% in more than 85% of the cases (98.87%, 89.47% and 91.41%). Moreover, the GEH statistics for the sum of all the links are smaller than 4% in all cases. In addition, the obtained values for the Theil's Indicator are smaller (or equal) than 0.2 for the three studied variables. So, taking into account these measurement, the proposed model can be accepted as replicating the microscopic simulated values acceptably well.

After this, the following figures show the corresponding variable for both models (proposed vs. benchmark) at the end of each 15-min simulation interval, all integrated in the same graphic. So, each point plotted in the graphics has the following 2D-coordinates: the measured variable obtained through Aimsun (abscissa axis) and the measured variable obtained through the proposed model (ordinate axis). Superimposed on these plots is the 45-degree line that would represent an identical simulation results for both models. Dots over the line represent that the proposed model overestimates the variable values, while dots under the line represent an underestimation. Moreover, we study the obtained errors in order to evaluate the correctness of the proposed model. We calculate

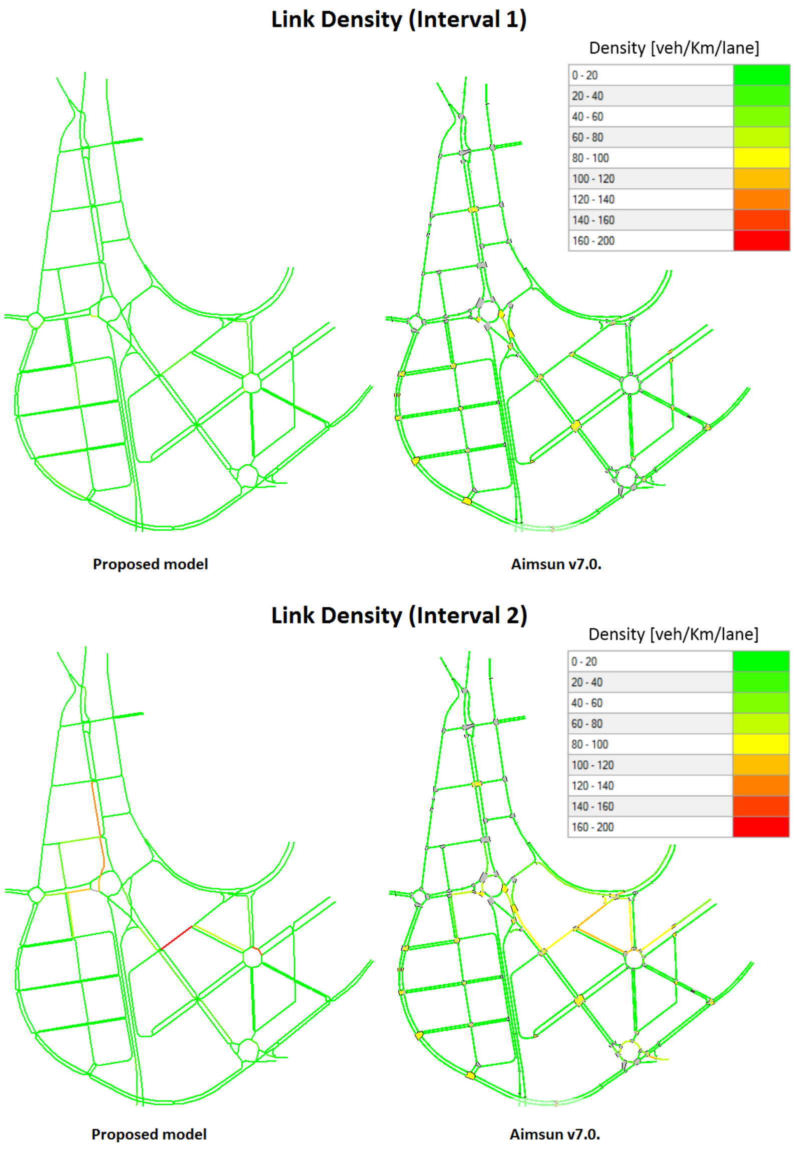


Figure 5.5.11: Average link densities in the network at the end of 15-min simulation intervals 1 and 2 .

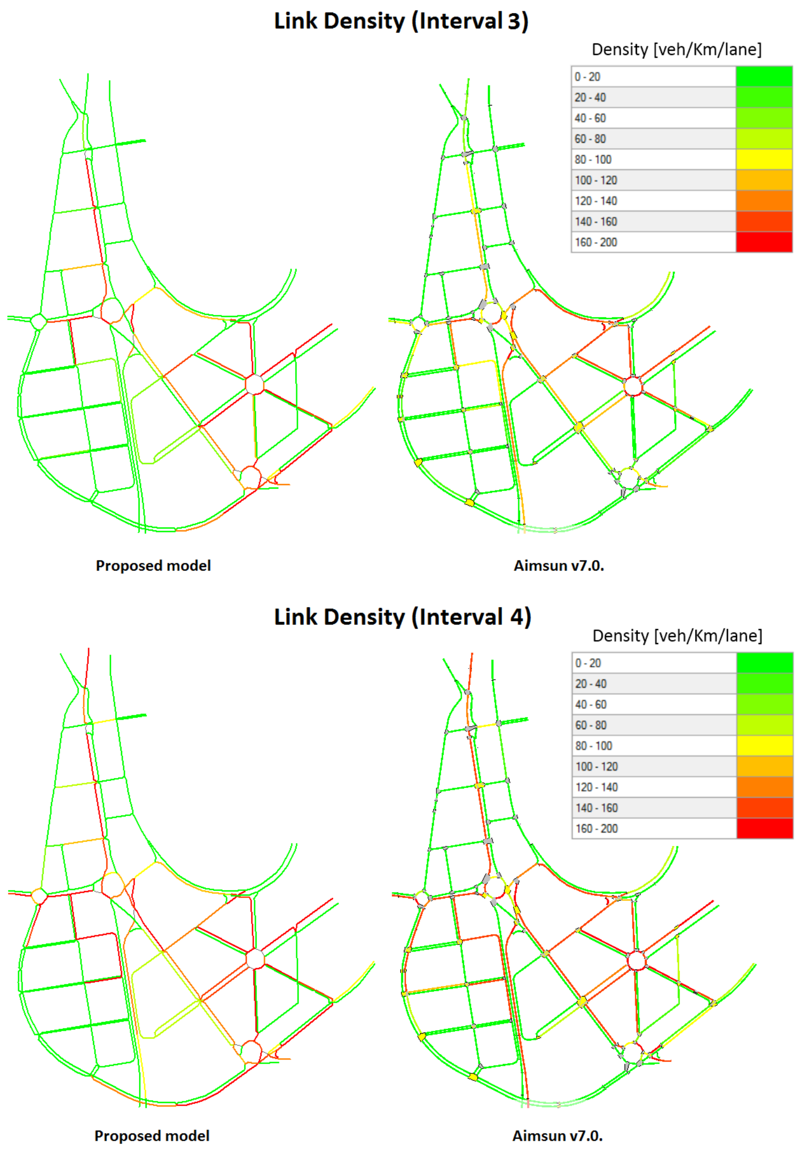


Figure 5.5.12: Average link densities in the network at the end of 15-min simulation intervals 3 and 4.

Criteria and Measures	**Acceptance Targets**
GEH Statistic $<$ 5 for Individual Link	**$>$ 85% of cases**
Vehicles Per Link	98.87 %
Travel Time	89.47 %
Density	91.41 %
GEH Statistic for Sum of All Link	**GEH $<$ 4 for sum of all link counts**
Vehicles Per Link	0.82 %
Travel Time	1.44 %
Density	2.07 %
Theil Indicator(U)	**U<=0,2**
Vehicles Per Link	0.14
Travel Time	0.20
Density	0.14

Table 5.4: Results of GEH measurements and Theil's Indicator.

the Root Mean Squared Error (RMSE) and the Normalized Root Mean Squared Error (NRMSE). Finally, we show a residual analysis with the 95% Confidence Interval using RMSE.

Density

Figure 5.5.13 shows the densities at the end of each time interval for all the links in the network. Each plotted dot corresponds with one link for one interval, and its coordinates are the density obtained through the bechmark microsimulator and the density obtained through the proposed model. The density is expressed as hourly vehicles per lane (veh/h/lane).

The RMSE has the same unit as the original measurements: veh/h/lane, and the NRMSE shows percentage. The NRMSE obtained value is 8.08% and the RMSE values is 17.32 vehicles per hour per lane.

Although the proposed DNL model takes into account considerably fewer parameters than the benchmark microsimulator, the reproduced densities are similar in both cases. However, we think that the topology of the network is impacting in the results. The test network of Amara is a typical European urban network with many short links. This causes small variations in the number of vehicles in the links results in large changes in density values. The variable "vehicles per link", which does not consider the length of the link explicitly, removes this effect.

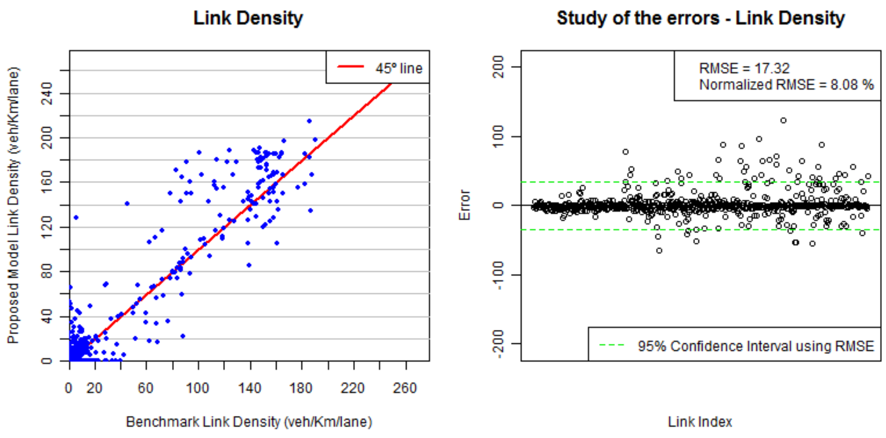


Figure 5.5.13: Link Densities: Benchmark vs. proposed model.

Vehicles per Link

Figure 5.5.14 shows the number of vehicles on each link in the network at the end of each 15-min simulation interval. Each dot of the plot corresponds with one link. Its coordinate x is the number of vehicles in this link obtained through the benchmark microsimulator, and its coordinate y is the number of vehicles in this link obtained through the proposed model. The vehicles per link measure is expressed in vehicles (veh).

The NRMSE value is 2.93% and the RMSE is approximately 4 vehicles. As expected, the residual analysis shows only that for few links the results obtained through both models are significantly different.

Travel Time

Figure 5.5.15 shows the travel time on each link in the network at the end of each 15-min simulation interval. Each plotted dot corresponds with one link. The x-coordinate is the travel time simulated by the benchmark model, while the y-coordinate is the travel time obtained through the proposed mesoscopic model. In this case, we average the travel time experienced for all vehicles that exit each link during each simulation interval. Travel time is expressed in seconds (sec).

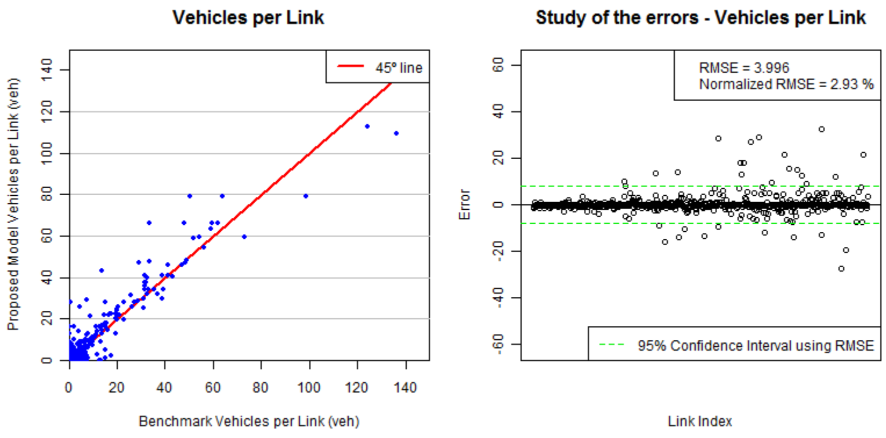


Figure 5.5.14: Vehicles per Link: Benchmark vs. proposed model..

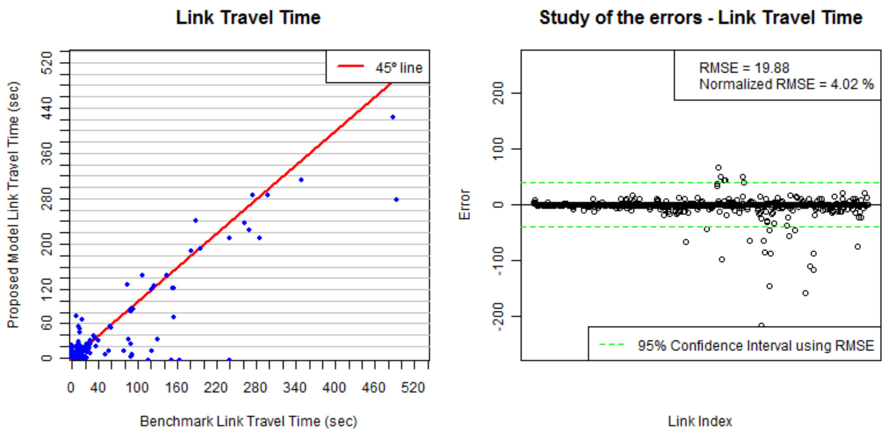


Figure 5.5.15: Link Travel Times: Benchmark vs. proposed model.

The NRMSE value is 4.02% and the RMSE value is 19.88 seconds.

It is important to say that both models assume the following situation can occur: any vehicle may exit a certain link during a certain time interval. In this case, the Aimsun model returns the value " − 1" for this link for this interval. In order to facilitate the comparison, we adopt the same convention. It is for this reason that, if we observe the results, we can appreciate that there are some dots plotted over the axis. Actually, they are not over the axis. These dots have one coordinate (or both) equal to " − 1" (although the resolution of the graphic may or may not show this).

Summary of the results
If we analyze the aggregated results for the total simulation duration (one hour), we observe that that the results obtained for both models are very similar for travel time and vehicles per link variables, with a NRMSE of 4.02% and 2.93%, respectively. However, in the case of density, an analysis of of the errors shows that the proposed model overestimates the density . So, the NRMSE rises up until 8.08%.

In summary, we consider that the experiments performed over the urban network of Amara look promising. The results obtained for both models (proposed mesoscopic model and benchmark micro) are very similar, except for the congestion caused by roundabouts. Moreover, our model overestimates the congestion on short links (less than 20 meters) when congestion appears in the network. Because Amara is a typical European network, links of this dimension can appear. So, it could be interesting to include in our model a specific treatment of roundabouts and short links.

5.5.4 Conclusions

In this section, an intensive computational experiments were conducted in order to test the developed mesoscopic simulation model.

The first experiment tried to demonstrate that the proposed model was able to reproduce the fundamental diagram that relates the main macroscopic variables: flow, density and speed. With this objective, we graphically compared the obtained simulation results with the macroscopic theoretical relationship proposed by Underwood (1961). We conclude that the obtained test results experimentally show that the developed DNL model is able to reproduce the fundamental diagram.

The second proposed experiment studied whether the developed model can reproduce the propagation of congestion properly. The main objective of this test was to complement the fundamental diagram test for demonstrating basic traffic performance in the links through the proposed mesoscopic model. In this case, the results show that our model respects the propagation of congestion, ensuring correct temporal and spatial location of the congestion on links.

In the third set of experiments, we tested our model against a microscopic simulator. As we previously specified, the proposed model is less detailed than microscopic models, which are considered the most realistic simulation tools for emulating the flow of individual vehicles. So, to experimentally investigate the performance of the developed model, we compared the results obtained through a selected microsimulator (Aimsun) with the results obtained through our model. In this case we used two real networks for the experiments: a freeway network in Auckland and the urban Amara network.

In the first case, the performed residual analysis shows the correctness of the proposed model, obtaining an NRMSE value of 0.38% for link travel time variable and 3.61% for density.

In the urban case, we performed two different analysis of the results. First of all, we visually compared the density of the links in the network and showed here the density obtained after running both models at the end of each 15-min interval of the simulation. We see that obtained densities are very similar in all four intervals. However, we note that our model overestimates the density at roundabouts, causing differences in some adjacent links compared with the results obtained through Aimsun.

After this, we quantitatively analyzed the obtained results for three link variables: density, vehicles per link and travel time. In this case, the residual analysis shows an acceptable NRMSE of 8.08%, 4.02% and 2.93% for density, travel time and vehicles per link variables, respectively.

In summary, we conclude that all the experiments performed look promising. The final experiment shows that, although the proposed model takes into account fewer parameters than Aimsun (only two), the reproduced traffic behaviour results similarly in both cases. We note that our model overestimates congestion on roundabouts and at links of short dimension (less than 20 meters) against the microscopic simulation results. The results obtained for the performed computational experiments demonstrate the ability of the developed model to reproduce multilane multiclass traffic behaviour for medium-sized networks.

5.6 Summary and Contributions

In this chapter a new DNL model has been developed in order to be embedded into the DTA scheme proposed in Chapter 4.

Most DNL models proposed in the literature for a DTA scheme have problems when applied to medium-large networks. On the one hand, DTA models require more details than macroscopic simulation models can offer. On the other hand, there exist microscopic models which are difficult to calibrate due to the great number of parameters usually needed.

In this chapter, a DNL problem has been solved with the proposed mesoscopic simulation model. Unlike other existing mesoscopic models, the presented DNL model is able to reproduce a multiclass urban traffic urban through the disaggregated treatment of demand while considering each individual vehicle. Additionally, the proposed model pays special attention to lane-changing procedures, which cannot be ignored because of their impact on traffic flow propagation.

First of all, this chapter has summarized the relevant literature regarding DNL models. The approach proposed in this thesis works around the DNL process based on simulation. Thus, in this chapter the discussion focuses exclusively on DNL models that base their link flow propagation on simulation: macroscopic, mesoscopic and microscopic models.

A wide range of published research has been reviewed. Before this review, a new classification scheme of DNL models based on simulation was presented. This representation is adapted from the well-known proposal by Astarita (2002). This scheme serves as a basis for a more detailed classification.

The DNL problem is crucial to performing DTA. The objective of this thesis is to develop a DTA that can reproduce traffic dynamics, specially in urban situations. Taking this into account has been essential in developing an efficient multiclass multilane DNL model. In this chapter, we have presented our DNL model based on a mesoscopic scheme that considers a continuous-time, link-based approach with complete demand discretization. The adapted classification scheme shows how our model can be categorized in the existing literature about DNL models based on simulation.

The developed mesoscopic traffic flow simulation model is solved using an algorithm based on discrete events. The motivation for adopting an event-based simulator in our DNL component was that it can potentially be much more computationally efficient than a time-step model. Moreover, we have taken into account that our flow propagation process considers vehicle to vehicle and that it is formulated from the space-dependent time relationship. So, we have considered that it is more suitable to use an event-based paradigm,which coincides with the idea of considering time as a dependent variable of the problem.

Then, the proposed DNL model was evaluated using laboratory tests. In order to demonstrate our multilane model, an intensive computational experiments were conducted. First, the model was applied to a small dummy network and the corresponding fundamental diagrams were examined for basic correctness. Then, in order to study the propagation of the shockwaves arising from an incident, the model was run over a simple freeway corridor where a blocking incident was simulated. Finally, the model was computationally tested on two real networks: a freeway network in Auckland and the urban Amara network (Spain). The obtained results were compared to those obtained through a selected benchmark microsimulator (Aimsun v.7.0).

The first experiment investigated the relationship among the values of the flow and the density simulated by the proposed DNL model. With this aim, we graphically compared these simulation results with the macroscopic theoretical relationship proposed by Underwood (1961). The results show links where relationship between flow and density is very similar to the Underwood case. We conclude that the obtained test results have experimentally shown that the developed model is able to reproduce the fundamental diagram.

The aim of the second proposed experiment was to complement the fundamental diagram of the first experiment in demonstrating basic traffic performance on links in the proposed mesoscopic model. The experiment studied if the implemented model reproduces the propagation of the congestion properly. In order to achieve this, our DNL model was run over a small freeway corridor where a blocking incident was simulated during a certain time interval. The obtained cumulative flows over time were plotted for all the links in the test network. The corresponding analysis has shown that the proposed model respects the propagation of congestion, ensuring correct temporal and spatial location of congestion at the link level.

In the last experiment, two real networks were used to test the obtained results against a microscopic simulation. The objective of this experiment was to validate the correctness of the developed DNL model. This model is less detailed than microscopic ones, which are considered the most realistic simulation tools. So, to experimentally investigate the performance of the proposed model, we compared the results obtained through a microsimulator (Aimsun) with those of our model.

The obtained results have been analyzed and discussed using different graphics and goodness-of-fit measures as references. We have studied densities, vehicles per link and travel times obtained through both models at the end of each 15-min simulation interval. In order to evaluate the correctness of the proposed model, we have studied the errors between both measures (GEH, Theil's Indicator, RMSE and NRMSE). The residual analysis demonstrates good performance of the developed model.

In the urban case, we also have visually compared link densities at the end of each 15-min interval of the simulation, obtained through the proposed model and through Aimsun. We conclude that the density results are very similar, except for the congestion caused by roundabouts. The proposed model overestimates congestion at the roundabouts, causing differences in some adjacent links compared with the results obtained through Aimsun.

The experiments performed on the two networks look promising. Although the proposed model takes into account only two calibration parameters, versus a hundred of parameters into the microsimulator, the reproduced traffic behavior is very similar in both cases. So, the results obtained for this third set of experiments demonstrate the good quality of the proposed model. Furthermore, the results demonstrate the ability of the model to reproduce multilane multiclass traffic behaviors for medium-sized urban networks. However, our model overestimates congestion on short links (less than 20 meters), so, it could be interesting to improve the model by including a specific treatment of roundabouts and short links, which are typical of European networks.

Chapter 6

Flow Reassignment

6.1 Introduction

As we see in Chapter 4, the iterative process proposed for solving the DTA problem has two fundamental components: the DNL and the flow reassignment method. The second one, the flow reassignment component, refers to methods that determine the new flow assignment on each link at each new iteration of the global process. Usually, these processes take the assignment of the travel demands in the paths used in the last DNL, and review or adjust these allocations for the next iteration, i.e., for the next dynamic loading. This reassignment among paths is usually based on link costs according to link travel times which have been obtained from the dynamic loading of the previous DTA iteration.

The DNL component has been extensively investigated in numerous publications, while the flow reassignment methods have been generally neglected. So, in this chapter we present the most studied reassignment methodologies that appear in the literature.

It should be mentioned that, although flow reassignment takes much less time and uses fewer computing resources than DNL (over 95% of the total computational time, Carey and Ge (2012)), the former has more direct influence on the convergence speed of the solution (number of iterations) and also on whether the global process converges to the DUE or not. Obviously, this is because the DUE is defined in terms of balancing the total travel cost on the used paths, and it is the flow reassignment process which is responsible for reallocating the flow to achieve this equality. On the other hand, the DNL process is only responsible for calculating the link travel times and, consequently, the link travel costs, both of which arise from the corresponding assignment.

In Section 2.2.3, we formulated DUE with the mathematical model defined by system Equations 6.1.1.

$$\begin{cases} c_{odpt} - c^*_{odt} \geq 0 \\ f_{odpt} \cdot (c_{odpt} - c^*_{odt}) = 0 \\ f_{odpt} \geq 0 \end{cases} \tag{6.1.1}$$

And the flow balance Equation 6.1.2.

$$\sum_{p \in P_{odt}} f_{odpt} = q_{odt} \; \forall o, d, t \tag{6.1.2}$$

In 1993, Friesz showed that this problem is equivalent to solving the problem of variational inequalities of finite dimension, which consists of finding a link flow vector f^*, such that Equation 6.1.3 is satisfied (Friesz et al. (1993)).

$$\begin{gathered} [f - f^*]^T \cdot c \geq 0 \; \forall f \in \aleph \\ \aleph = \left\{ f_{odpt} \middle| \sum_{p \in P_{odt}} f_{odpt} = q_{odt} \; \forall o, d, t, \; f_{odpt} \geq 0 \right\} \end{gathered} \tag{6.1.3}$$

Wu (1991), and especially Wu et al. (1998) showed that it is equivalent to solving discretized variational inequality 6.1.4.

$$\sum_t \sum_{p \in P_{odt}} c_{odpt} \cdot \left(f_{odpt} - f^*_{odpt} \right) \geq 0 \tag{6.1.4}$$

The existence and uniqueness of the solution of this model may be demonstrated depending on the properties of the function $c_{odpt}(f)$, i.e., link and path travel costs depend on path flows, and path flow depends on link and path travel costs. The properties of this function are not easy to check (because the function is the output of a simulation model and not an analytical function), usually do not make claims about the existence or uniqueness of a solution The equilibrium principle is used to calculate an approximate solution of the variational inequalities discretized in time.

However, it is important to note that not all computer implementations based on this algorithmic framework provide solutions that achieve DUE. The flow reassignment algorithms can be grouped into two categories: preventive and reactive. In the first, it is implicitly assumed that the network traffic conditions are predictable, so they are responsible for decision making, based on their historical experience. While the reactive algorithms assume that network traffic conditions are not predictable because of the possibility that incidents occur, demand variability, and the stochasticity of traffic systems, among other factors. However, in this case, the users have real-time information about the current traffic conditions and travel times experienced. Therefore, they can make decisions while they are en route.

In 1993, Friesz et al. showed that DUE solutions are achieved through a preventive flow reassignment mechanism, combining the experienced travel times with time predictions of flow and travel cost variations (Friesz et al. (1993)).

A wide variety of preventive algorithms have been proposed for explicitly solving the set of variational inequalities presented above, thus creating a DUE solution: projection methods, methods of alternating directions, and different adaptations of the Method of Successive Averages. We will review these processes in Section 6.2 of this chapter.

Other proposals, which can be considered DTA with a flow reassignment component based on a reactive approach are those that model the process with discrete choice theory (Akiva and Lerman (1985)). This approach considers that P_{odt}, the set of all possible paths from an origin o to a destination d departing at the time interval t, is a finite set of choice alternatives, each of which has a utility by the responsible decisions making, the traveler. It can be considered a random variable that for the alternative k consists of:

- the measurable utility, a deterministic systematic component $C_k\left[v\left(t\right)\right]$, where $v\left(t\right)$ is a vector of values of the variables which depend on the utility in time t, and
- a random error $\varepsilon_k\left(v\right)$, which represents the perceptual error due to the lack of perfect information.

Thus, the perceived utility of alternative k (path k) at time t is shown in Equation 6.1.5.

$$U_k\left(t\right) = -\theta \cdot C_k\left[v\left(t\right)\right] + \varepsilon_k\left(v\right) \ \forall k \in P_{odt} \tag{6.1.5}$$

Where θ is a positive parameter when $C_k\left[v\left(t\right)\right]$ is the expected value of a negative utility, such as expected travel cost or travel time. Assuming that the random term satisfies the condition that the expected values are $E\left[\varepsilon_k\left(v\right)\right]=0,\ \forall k$, and which are independently and identically distributed with a Gambel distribution, it can be proved that the probability of choosing the alternative k at time t is given by the logit function 6.1.6.

$$P_k\left(t\right)=\frac{e^{-C_k\left[v(t)\right]}}{\sum\limits_{j\in P_{odt}} e^{-C_j\left[v(t)\right]}} \tag{6.1.6}$$

A known drawback of using logit functions for the route choice is that this function does not distinguish overlapping paths in order to overcome the unwanted side effects of wrong choices. Some researchers like Cascetta et al. (1996) or Ben-Akiva and Bierlaire (1999) proposed a modified logit that adds to the utility definition a penalty term, which depends on the degree of overlap between the alternative paths. In this model, the choice probability P_k of each alternative path k belonging to P_{odt} is defined by the function 6.1.7.

$$P_k\left(t\right)=\frac{e^{-\theta\cdot\left\{C_k\left[v(t)\right]+CF_k\right\}}}{\sum\limits_{j\in P_{odt}} e^{-\theta\cdot\left\{C_j\left[v(t)\right]+CF_j\right\}}} \tag{6.1.7}$$

where:

$C_k\left[v\left(t\right)\right]$is the same measurable utility as in the Logit function case ,

θ is the same scale factor as in the Logit function case, and

CF_k is the common factor of the path k that is directly proportional to the degree of overlap of the path k with other alternative paths. Therefore, the paths that present considerable overlap with others, have a factor CF_k greater, and thus less utility with respect to similar paths. An example of CF_k, proposed by Cascetta et al. (1996), might be the Equation 6.1.8.

$$CF_k=\beta\cdot\ln\left(\sum_{j\in P_{odt}}\left(\frac{L_{jk}}{L_j^{1/2}\cdot L_k^{1/2}}\right)^{\gamma}\right) \tag{6.1.8}$$

where:

L_{jk} is the length of the links that paths j and k have in common,

L_j, L_k are the lengths of paths j and k, respectively,

β and γ give greater or lower weight to the common factor, depending on their value.

Since the main objective of this thesis is to develop a DTA model based on user behavior that follows a DUE approach, we will not pay more attention to the reactive methods briefly introduced before.

Section 6.2 summarizes the relevant literature regarding flow reassignment methods based on a preventive approach. We present the main algorithms that appear in the literature, paying special attention to the Method of Successive Averages and all its limitations and opportunities. In Section 6.3, we define a new flow reassignment method based on an MSA scheme that tries to take advantage of the information from the DNL component. Finally, in Section 6.4 we use a real network example in order to verify the performance and feasibility of the flow reassignment method embedded into the proposed DTA scheme. We then compare the obtained results with other MSAs proposed in the literature.

6.2 Literature Review

As we justified in the previous Section 6.1, here we only revise the algorithms proposed in the literature to solve the set of variational inequalities under a preventive approach, so that the solution is a DUE. Such methods are:

- Projection methods, directly extrapolated from those of the static problem. (Wu (1991), Wu et al. (1998) and Mahut et al. (2007))
- Methods of alternating directions, Lo and Szeto (2002).
- Different versions and adaptations of the Method of Successive Averages (MSA). (Tong and Wong (2000), Varia and Dhingra (2004), Mahut et al. (2003, 2004, 2007) and Sbayti et al. (2007))

6.2.1 Projection Methods

Since the analytical formulation of DUE is a generalization of the variational inequalities with time discretization, it seems logical that in this thesis the first approximation to this problem is to explore the application of projection procedures into the space of the paths in order to solve it.

The equilibrium algorithms used in the static equilibrium models, which operate in the space of the path flows, provide some ideas that can be adapted heuristically in order to obtain solving methodologies for equilibrium DTA. These algorithms are adaptations of the classic methodologies like the Simplex method or the reduced gradient algorithm implemented using Jacobi or Gauss Seidel decomposition. Some references on the subject areLeventhal et al. (1973), Dafermos (1971) and Patriksson (1994).

As we discussed in the previous Section 6.1, Wu (1991); Wu et al. (1998) showed that the variational inequality problem shown in Equation 6.1.3 is equivalent to the proposed user dynamic equilibrium conditions shown in Equation 6.1.1.

The solution approach used in the work of Wu is based on the time discretization already defined in Chapter 2, where we formulate the DUE. Thus, following the defined notation of Section 2.2.1, time is discretized into a finite number of periods of duration $\triangle t$, each of which is called t.

To solve the problem in 6.1.3, given an initial feasible solution f^0 at iteration $k = 0$, Wu et al. (1998) presented an algorithm that iterates between a DNL and path flow reassignment while using approximations of discrete time. At each iteration, the time dependent path flow reassignment is calculated using an adaptation of the projection method, which involves the quadratic problem shown in Equation 6.2.1.

$$\min_{f_{odpt}^{k+1} \in \aleph} \left\{ \sum_t \sum_{p \in P_{dot}} \left[\left(f_{odpt}^{k+1} - f_{odpt}^{k} \right) c_{odtp} + \frac{1}{2\alpha} \left(f_{odpt}^{k+1} - f_{odpt}^{k} \right)^2 \right] \right\} \tag{6.2.1}$$

where:

$c_{odpt}^{k} = c_{odtp} \left(f^k \right)$

α is a positive constant used to weight the two terms in square brackets.

Solving this problem yields a new path flow vector f^{k+1}, which is used as input to the DNL which in turn is used to update the network travel times vector c_{odpt}^{k+1}. This iterative process continues until the Equation 6.2.2 is achieved.

$$\sum_{t} \sum_{p \in P_{odt}} \left(f_{odpt}^{k+1} - f_{odpt}^{k} \right)^2 < \varepsilon \ (\varepsilon > 0) \tag{6.2.2}$$

Since the above problem is separable for each time interval t and for each OD pair, it can be solved separately by solving the problem 6.2.3 for each interval and for each OD pair:

$$\begin{aligned} \min_{f_{odpt}^{k+1} \in \aleph} \sum_{t} \sum_{p \in P_{odt}} & \left[\left(f_{odpt}^{k+1} - f_{odpt}^{k} \right) c_{odpt} + \tfrac{1}{2\alpha} \left(f_{odpt}^{k+1} - f_{odpt}^{k} \right)^2 \right] \\ s.t. \quad & \sum_{p \in P_{odt}} f_{odpt} = q_{odt} \\ & f_{odpt} \geq 0 \end{aligned} \tag{6.2.3}$$

In his thesis, Wu (1991) developed an efficient algorithm for solving the problem shown in Equation 6.1.1. There were many inquiries about how to solve a single constraint problem with bounded variables (Kennington and Helgason (1980), Dussault et al. (1986) and Pardalos and Kovoor (1990))). In this case, the problem was rather different in that the variables had no upper bound. Wu solved it using the same idea as that proposed in the work by Kennington and Helgason (1980)).

Later, Florian et al. (2001) proposed DTA models using mesoscopic simulation for DNL. They proposed a path flow reassignment that reproduces the work of Wu based on projection methods (Wu et al. (1998)).

Another work based on the projection algorithms is the proposal included in the comparison of assignment methods based on simulation that Mahut and Florian presented in 2008. This algorithm operates in the space of path flows, and hence it is very attractive to adapt the equivalent of the projected gradient and the reduced gradient algorithms, even though there is no formal objective function that can be identified and the model formulation is a time discrete variational inequality. Since there is no objective function, the step sizes adopted are those of the MSA or the modified MSA described below.

In order to state the algorithms, the notation used is the following:

K^+	Set of paths with positive flow;
s_k	Cost (time) of a path;

$\bar{s}$	Average value of the path costs;
p_k	Proportion of input flows to the paths $k \in K^+$;
d_k	Direction of change for each path;
d_k^n	Normalized direction;
α	MSA step size.

The quasi-projected gradient algorithm proposed by Mahut et al. (2007) modifies the flow changes by using the following steps:

Step 1 Compute the vector of $d_k = \bar{s} - s_k, k \in K^+$;

Step 2 Normalize the vector $d_k^n = \frac{d_k}{\sum\limits_k |d_k|}$;

Step 3 Check for α_{max}, the largest value of α: $\alpha_{max} = \max\left\{\frac{p_k}{d_k^n} \,|d_k^n < 0\right\}$, which would diminish the input proportion of a path to 0;

Step 4 The step size is $\alpha = \min\{\alpha_{max}, \alpha_{MSA}\}$;

Step 5 Update the path proportions $p_k = p_k + \alpha \cdot d_k^n$.

The quasi-reduced gradient algorithm modifies the flow changes by using the following steps:

Step 0 Select the path that has the largest flow: $k^* = \arg\max\{f_k\}$;

Step 1 Compute the vector of $d_k = s_k^* - s_k,\ k \neq k^*$and $d_k^* = -\sum\limits_{k \neq k^*} d_k$;

The remaining steps are the same as in the case of a quasi-projected gradient. Also, note that the trips that are assigned to be loaded on each path k are simply the product of the demand for the O-D pair, multiplied by the path proportions p_k.

6.2.2 Alternating Direction Methods

In order to facilitate understanding of the method, the formulation shown in Equation 6.1.3 is simplified using matrix notation. Thus, the variational inequalities equations would be as follows in Equation 6.2.4.

$$\begin{array}{c} [f - f^*]^T c(f) \geq 0 \; \forall f \in \aleph \\ \aleph = \{f \,|Af = q\} \end{array} \tag{6.2.4}$$

where:

f	Time-dependent path flow vector, $f = \{f_{odpt} \; \forall o, d, t, p \in P_{odt}\}$.
f_{odpt}	Number of trips from origin o to destination d departing on the time interval t by the assigned path $p \in P_{odt}$.
$c(f)$	Vector of $c^t(f) \; \forall t$.
$c^t(f)$	Vector of c_{odpt}.
$c_{odpt}(f)$	Travel times experienced by vehicles departing from origin o to destination d at time interval t through the assigned path $p \in P_{odt}$.
A	Diagonal matrix with diagonal elements equal to $A(t)$.
$A(t)$	Incidence OD matrix of dimension $\|od\| \times \|P(o,d,t)\|$ where the item $(o, d, p) = 1$ if the path p departing a time interval t belongs to the path set P_{odt}, and 0 otherwise.
q	Vector of $q^t \; \forall t$.
q^t	Vector of q_{odt}.
q_{odt}	Total trips from origin o to destination d departing on the time interval t.

These methods begin by adding a Lagrangian multiplier u to the linear demand constraint $Af = q$ in order to obtain the following equivalent formulation for the variational inequalities problem to be discussed.

Find $x^* \in \psi$, such that Equation 6.2.5 is satisfied.

$$(x - x^*)^T F(x^*) \geq 0,\ x \in \psi \tag{6.2.5}$$

where: $x = \begin{pmatrix} f \\ u \end{pmatrix}$, $F(x) = \begin{pmatrix} c(f) - A^T u \\ Af - q \end{pmatrix}$

The motivation for this modification is to build a set ψ that has an easier projection than the original set $\aleph$.

For such problems, Gabay and Mercier (1976) and Gabay (1983) proposed the following alternating direction method:

- Given $\left(f^k, u^k\right)$ at the k-th iteration, find f^{k+1} such that:

$$\left(f^{'} - f^{k+1}\right)^T \left\{c\left(f^{k+1}\right) - A^T\left(u^k - \beta_{k+1}\left(Af^{k+1} - q\right)\right)\right\} \geq 0\ \forall f^{'} \tag{6.2.6}$$

- Then, update u via:

$$u^{k+1} = u^k - \beta_{k+1}\left(Af^{k+1} - q\right) \tag{6.2.7}$$

  where $\beta_{k+1} > 0$ is a given penalty parameter (or sequence of parameters) for the linear constraint $Af = q$.

When u^k and β_{k+1} are known, then$\left\{c\left(f^{k+1}\right) - A^T\left(u^k - \beta_{k+1}\left(Af^{k+1} - q\right)\right)\right\}$ is a function of f^{k+1}. Thus, 6.2.6 is a subproblem which involves only f^{k+1}. Alternating directions methods are attractive for large-scale problems if the subproblem can be solved efficiently. Nevertheless, the exact solution of the subproblem may be computationally intensive by itself.

To overcome this deficiency of the Gabay approach, in 1996 Zhu and Marcotte considered the case of co-coercive functions and proved that many iterative schemes based on the projection method converge to a solution of the proposed variational inequalities problem (Zhu and Marcotte (1996)). Motivated by this study, Han and Lo (2002) developed a modified alternating directions method, extending the proposal by He and Zhou (2000), where they proposed a solution method for non-linear functions which sought the convergence of the method when the function was co-coercive[1].

[1]Let ϕ a not empty convex closed subset of $\mathbb{R}^n$, a function $F : \phi \rightarrow \mathbb{R}^n$ is co-coercive if there is a positive constant μ such that:

$$(u - v)^T \cdot [F(u) - F(v)] \geq \mu \cdot \|F(u) - F(v)\|^2\ \ \forall u, v \in \phi$$

According to Zhu and Marcotte (1996), the co-coercive functions are monotone, but perhaps they do not have to be strictly monotone. On the other hand, the strictly monotone and Lipschitz continuous functions are co-coercive. So, you could say that co-coercive functions are an intermediate concept between monotone and strictly monotone functions.

Given the above concepts, Han and Lo (2002) proved the convergence of these new alternating directions methods based on:

- The function $c(f)$ is co-coercive.
- The solution set of the proposed variational inequalities problem is not empty.

In 2002, Lo and Szeto proposed a new formulation based on cells for the DTA problem with a DUE hypothesis and following the approach of the variational inequalities problem (Lo and Szeto (2002)). In order to solve this formulation, they used the alternating direction method developed by Han and Lo (2001) for solving co-coercive valid inequalities.

This method requires that the path travel cost based on travel time is a coercive function of the path flow and also that the solution set is not empty.

The slightly smoothed nature of the path travel time functions makes the solution methods based on derivatives difficult to apply. That is why the authors developed a resolution method for the variational inequalities shown in Equation 6.1.3, based on projection methods.

Their proposed alternating directions algorithm is specified below.

First of all, they denote the set K, the space in which it shall be projected; and the residual error function for the problem is considered:

$$e(x,\beta) = e(f,u,\beta) = \begin{pmatrix} f - P_K\left\{f - \beta\left[c(t) - A^T u\right]\right\} \\ \beta(Af - q) \end{pmatrix}$$

where β is a certain penalty parameter and $P_K\{\cdot\}$ is the projection in K.

They verified that if $e(x,\beta) = 0$ then, the proposed variational inequalities are solved. Furthermore, they define:

$$r(x,\beta) = \begin{pmatrix} f - P_K\left\{f - \beta\left[c(t) - A^T(u - \beta(Af - q))\right]\right\} \\ \beta(Af - q) \end{pmatrix}$$

The proposed variational inequalities system is equivalent to finding the point where the function $r(x, \beta)$ is zero.

The detailed steps of the algorithm are:

Step 0 Given an arbitrary initial point $x = \left(f^0, u^0\right)$, and positive constants β, μ, ε such that $\beta < 4\mu$ and $\varepsilon > 0$. Set iteration number $k = 0$.

Step 1 Calculate $\bar{x}^k = \left(\bar{f}^k, \bar{u}^k\right)^T$ as follows:

$$\bar{f}^k = P_K \left\{ f^k - \beta \left\lfloor c\left(f^k\right) - A^T\left(u^k - \beta\left(Af^k - q\right)\right)\right\rfloor \right\}$$
$$\bar{u}^k = u^k - \beta\left(A\bar{f}^k - q\right)$$

Step 2 Calculate $x^{k+1} = \left(f^k, u^k\right)^T$

$$x^{k+1} = x^k - t_k\gamma_k\left(x^k - \bar{x}^k\right)$$

where:

$$t_k = \delta_k\left(1 - \frac{\beta}{4\mu}\right), \quad \delta_k \in (0,2) \text{ so that } t_k \in (0,1)$$
$$\gamma_k = \frac{\left\|r\left(x^k,\beta\right)\right\|^2}{\left\|r\left(x^k,\beta\right)\right\|_H^2}$$

where:

$$H = \begin{pmatrix} I + \beta A^T A & 0 \\ 0 & I \end{pmatrix}$$
$$\left\|r\left(x^k,\beta\right)\right\|_H^2 = r^T\left(x^k,\beta\right)Hr\left(x^k,\beta\right)$$

Step 3 If $\left\|r\left(x^{k+1},\beta\right)\right\| < \varepsilon$, STOP.

Otherwise, set $k = k + 1$ and Go to Step 1.

This flow reassignment method consists mainly of two steps. First it calculates the flow projections on the path flows and the associated multipliers, taking into account the traffic conservation constraints. In the second step, the path flows and the multipliers are updated, replacing them with either weighted averages of their own values of the same iteration or weighted averages of their projections.

Each iteration requires the determination of the path travel costs vector $c\left(f^k\right)$ from the path flow given in the last iteration. To this end, a DNL procedure is executed at each iteration. Furthermore, the algorithm requires projections over the non-negative quadrant without the need for any matrix inversion. In this way, the computational load per iteration is mild, so the algorithm resolution for large-scale networks is feasible within a reasonable time.

The specified method uses four parameters: β, μ, δ_k and t_k, which are not independent of each other as t_k is determined from the other three. Since μ, δ_k are not used elsewhere in this method, it is only necessary to assign numerical values to β and t_k. Following Lo and Szeto (2002) can ensure that $1 \geq t_k \geq 0$ and $\beta > 0$.

Finally, it should be noted that in the alternating directions methods, the search strategy involves convergence towards a feasible solution. The degree to which flow conservation constraints are met is given by the value $\left\| r\left(x^k, \beta\right) \right\|^2$. However, for the other methods analyzed in this section, the generated path flows are always feasible, i.e., they are non-negative and they satisfy the flow conservation constraints.

6.2.3 Method of Successive Averages

Among different methods of solving the general static assignment problem, perhaps the Method of Successive Averages (MSA) is the most simple and efficient. This appears to be the most widely used method for the path flow reassignment component in a DTA scheme.

MSA was introduced by Robbins and Monro (1951) for quite a different type of problem. The method was later used in transportation modeling (e.g.,in Powell and Sheffi (1982)) and it has been widely used since then. In the present context of path flow reassignment, MSA consists of removing a fraction of the flow (step size) from each of the currently used paths and adding this amount to the flow of the current shortest path for each OD pair.

MSA takes the flow on a link as a linear combination of: the previous flow and an auxiliary flow from an all-or-nothing assignment. The method is based on a predetermined series of step sizes for overcoming the problem of allocating too much traffic to congested links. With the proper choice of the step size at each iteration, the method converges to the Wardrop equilibrium solution in static traffic assignment (Sheffi (1985)).

In order to ensure this MSA theoretical convergence, the step size series must satisfy the two conditions shown in Equation 6.2.8.

$$\begin{aligned} \sum \lambda^k &= \infty \\ \sum \left(\lambda^k\right)^2 &< \infty \end{aligned} \tag{6.2.8}$$

where λ^k is the step size of iteration k.

One of the most convenient move size sequence that satisfies the above two conditions is the reciprocal of the iteration number; i.e., $\lambda^k = \frac{1}{k}$. With this step size, path assignments of the k iteration f^k may be updated to obtain the path assignment f^{k+1} for iteration $k+1$, following Equation 6.2.9.

$$f^{k+1} = f^k + \frac{1}{k} d^k = f^k + \frac{1}{k}\left(y^k - f^k\right) = \left(1 - \frac{1}{k}\right) f^k + \frac{1}{k} y^k \tag{6.2.9}$$

where:

$d^k = y^k - f^k$	is the search direction of iteration k
y^k	is the auxiliary path assignment obtained by using the all-or-nothing assignment in iteration k.

When it is applied to the dynamic assignment, the MSA needs a slight modification of the algorithm. Since in static assignment the path flows are time independent, the application of the averaging process on the path flows is equivalent to that on the link flows. This simplifies the calculation of an updated traffic pattern at each iteration, since the path flows accumulated in previous iterations of MSA do not need be assigned to the network again in the current iteration.

However, in dynamic assignment the application of the averaging process to link flows will lead to erroneous results. This is because, when a small amount of traffic is taken away from a previous path, the movements of the remaining vehicles on that path will be affected, as well as vehicles on many other paths in the network. These vehicles may move slower or faster on some links, and this will result in a totally different link flow pattern. Therefore, the averaging process should be applied to the path flows rather than the link flows, as for the static assignment (Tong and Wong (2000)).

As the assignment is dynamic in nature, the new path generated at any time interval of the current iteration (having no flow during the previous iteration) raises the problem of flow distribution as soon as it is no longer a shortest path in MSA. Tong and Wong (2000) proposed the following problem formulation and resolution in order to overcome this problem.

Let $f^k = f^k_{odpt} \forall o, d, t, p \in P^k_{odt}$ be the path flow vector at the kth iteration, where P^k_{odt} is the set of all paths obtained from all iterations so far, and f^k_{odpt} is the corresponding path flow values. These path flows are then loaded onto the network using a DNL process. Then, an auxiliary set of time-dependent shortest paths for all OD pairs for all time intervals, $Y = y_{odt} \forall o, d, t$, is determined by a time-dependent shortest path algorithm, where y_{odt} is the shortest path for a vehicle traveling to the destination node d from the origin node o departure at time t.

If the auxiliary path is newly generated (i.e., $P^{k-1}_{odt} \bigcap y_{odt} = \emptyset$), the updated path flow vector is determined by:

$$f^k_{odpt} = \begin{cases} (1-\lambda_k) f^{k-1}_{odpt} & if\ p \in P^k_{odt} \\ \lambda_k q_{odt} & if\ p = y_{odt} \end{cases} \quad \forall o, d, t$$

And the set of used paths at the k iteration is updated as $P^k_{odt} = P^{k-1}_{odt} \bigcup y_{odt}$.

However, if the auxiliary path is an old path (i.e., $y_{odt} \in P^{k-1}_{odt}$), the updated path flow vector becomes:

$$f^k_{odpt} = \begin{cases} (1-\lambda_k) f^{k-1}_{odpt} & if\ p \neq y_{odt} \\ (1-\lambda_k) f^{k-1}_{odpt} + \lambda_k q_{odt} & if\ p = y_{odt} \end{cases} \quad \forall o, d, t$$

And the set of used paths remains unchanged (i.e., $P^k_{odt} = P^{k-1}_{odt}$).

In the MSA method referred to above, and in the other methods considered in this chapter, the step size λ^k varies with the iteration number. Depending on the values that are given to these coefficients λ_k, we will obtain different implementation schemes of the MSA procedure: Tong and Wong (2000), Varia and Dhingra (2004), Florian et al. (2001), Mahut et al. (2003, 2004). Another possibility would be to choose a predetermined fixed λ^k, and let it remain constant over all iterations. Carey and Ge (2012) showed that the results obtained with MSA with constant step sizes $(1 > \lambda > 0)$ are in all cases worse than those obtained by MSA with a variable step size $\lambda^k = \frac{1}{k}$.

In 2004, Varia and Dhingra proposed a modification in the MSA procedure, using variable weight coefficients defined from a Logit distribution function, in order to take into account the cost (travel times) of the alternative paths for the flow reassignment of certain paths (Varia and Dhingra (2004)). The proposed process for the different conditions is given as follows.

In this case, aside from differences that depend on whether the generated auxiliary path is new or not, the first iteration of the process of the remaining iterations is also separate.

- First iteration:

  $$f_{odpt}^{k}=\begin{cases} \lambda q_{odt} & if\ p=y_{odt} \\ (1-\lambda)\, q_{odt}\alpha_{odpt}^{(1)} & if\ p\neq y_{odt} \end{cases} \quad \forall o,d,t$$

  where :

  λ is the flow distribution factor

  $\alpha_{odpt}^{(1)}=\frac{\exp(-\beta_{odpt})}{\sum\limits_{p}\exp(-\beta_{odpt})}$

  β_{odpt} is the instantaneous travel time[2] of the path p departing at time interval t.

- Next iterations:

  - If generated shortest path y_{odt} is in the set of paths of previous iteration P_{odt}^{k-1}:

    $$f_{odpt}^{k}=\begin{cases} \left(1-\frac{\lambda s}{k+1}\right) f_{odpt}^{k-1}+\left(\frac{\lambda s}{k+1}\right) q_{odt}\alpha_{odpt}^{(2)} & if\ p=y_{odt} \\ y_{odt} & if\ p\neq y_{odt}\, and\, p\in P_{odt}^{k-1} \\ \left(\frac{\lambda s}{k+1}\right) q_{odt}\alpha_{odpt}^{(2)} & if\ p\notin P_{odt}^{k-1} \end{cases}$$
    $$\forall o,d,t$$

    where:

    s is a suitable constant

    $\alpha_{odpt}^{(2)}=\frac{\exp(-\beta_{odpt})}{\exp\left(-\beta_{ody_{odt}t}\right)+\sum\limits_{p\notin P_{odt}^{k-1}}\exp(-\beta_{odpt})}$

[2]Instantaneous path travel time: Sum of link travel times along the path estimated at the time when drivers enter the path from origin.

– If generated shortest path y_{odt} is not in the set of paths of previous iteration P_{odt}^{k-1}:

$$f_{odpt}^{k} = \begin{cases} \left(\frac{\lambda s}{k+1}\right) q_{odt}\alpha_{odpt}^{(2')} & if\ p \notin P_{odt}^{k-1} \\ \left(1 - \frac{\lambda s}{k+1}\right) f_{odpt}^{k-1} & if\ p \in P_{odt}^{k-1} \end{cases}$$

where:
$\alpha_{odpt}^{(2')} = \frac{\exp(-\beta_{odpt})}{\sum\limits_{p \notin P_{odt}^{k-1}} \exp(-\beta_{odpt})}$

The different iterative algorithms iterate until some selected convergence criterion is satisfied. The convergence criterion often used is the relative gap for any feasible solution of DUE, proposed by Janson (1991). (See Section 4.3.6).

Generally, the successive averages algorithm ends when:

- It gets an acceptable error $GAP^k < \varepsilon$.
  Because this gap measures the deviation of the MSA solution from a true equilibrium solution, a 5% gap can be considered acceptable (Tong and Wong 2000).

- It reaches a maximum number of iterations: $k > k_{max}$.

Mahut et al. (2003, 2004) described a DTA model based on simplified microscopic traffic simulation. To this end, Mahut determined time-dependent route flows solving the MSA problem. One of the novel contributions of the author was the definition of another gap measure. While no formal convergence proof can be given for this type of algorithm, a measure of gap inspired by Janson's relative gap may be used for qualifying a given MSA solution, since the network loading map does not have an analytical form. We call this the refined relative gap.

The refined relative gap is the difference between the total travel cost experienced and the total travel cost that would have been experienced if all vehicles had the travel cost (over each interval) equal to that of the current shortest path. (See Section 4.3.6).

A relative gap of zero would indicate a perfect DUE flow.

In 2007, Sbayti, Lu and Mahmassani showed some of the limitations that arise when using the technique of successive averages in the field of DTA (Sbayti et al. (2007)).

Although it has been established that using the successive averages method for static assignment satisfies the Wardrop equilibrium solution, its convergence properties for the DTA are questionable since the majority of the studies on this subject base their conclusions on small networks.

These authors pointed out two drawbacks of the MSA, which tend to worsen when network congestion levels or network sizes grow. They explained that most of the works on this subject have avoided the use of MSA for large or congested networks.

The first problem is that MSA expressly requires the storage of all paths and its flow assignments at each iteration of the process. This can consume a lot of memory when implementing the algorithm, even for medium-sized networks.

The other disadvantage is that, when you try to divert traffic from the path with the worst travel times to the current optimum paths, the MSA does it indiscriminately, regardless of whether the path is the worst or only slightly worse than the optimal. This penalizes the paths that are only slightly lower as much as the paths that are far lower. The authors showed that this feature is inconvenient for getting good solutions when the congestion level is high.

Sbayti, Lu and Mahmassani proposed two new techniques to avoid these disadvantages of MSA applied to DTA. These techniques exploit the properties of mesoscopic simulation used by authors to perform the DNL component of the DTA. This simulation reproduces vehicle movement in platoons, but it also keeps track of each individual. This ability to track vehicles means that you can rebuild the whole path set and its path flow assignment from the vehicle trajectories. Therefore, it is a powerful tool for saving memory (especially when working with large networks) and for avoiding the first MSA problem mentioned.

To address the second mentioned drawback, the authors proposed a second implementation based on an ordering of the vehicles. For a given triplet, vehicles are ranked according to their travel times and only the slower vehicles are forced to update their current path towards a better route, provided by the current best path. The authors tested this new technique on two real-sized networks with excellent results.

But we can say that, with respect to the first mentioned disadvantage, possibly one of the most efficient modifications of the implemented methodologies is the proposal by Mahut et al. (2003), which limits the number of alternative paths for each origin destination pair and for each time interval. Let N_{odt} be the maximum number of different alternative paths that we want to get for a certain OD pair (o, d) for a certain time interval t. Taking into account the possibility of repeating the shortest path from one iteration to the next, a proper implementation of the algorithm will require a different number of iterations for each origin destination pair for each time interval if it is going to achieve these N_{odt} paths (K_{odt}).

In their proposed variant of the MSA, an initial feasible solution is computed by assigning the demand for each time period to a set of successive shortest paths. Starting at the second iteration, and up to a pre-specified maximum number of iterations,K_{odt}, the time-dependent link travel times after each loading are used to determine a new set of dynamic shortest paths that are added to the current set of paths.

For iteration k, $k \leq K_{odt}$, the volume assigned as input flow to each path in the set is $q_{odt}/k \, \forall o, d, t$. After that, for iteration k, $k > K_{odt}$, no new paths are added to the choice set. Only the shortest route among used paths is identified and the path input flow rates are redistributed over the known paths. This algorithm stops when the total travel time (sum of travel times of all vehicles) approaches the "optimal" travel time (the sum of vehicle travel times if they were to have taken the shortest paths). Also, it may stop if it achieves a predefined maximum number of total iterations K_{max}.

The algorithm specification is shown below:

```
Step 0    Initialize iteration count $k = 1$.
          Compute dynamic shortest paths based on free-flow travel times.
          Load the demands to obtain an initial solution.
          Set $k = k + 1$.
Step 1    If $k \leq K_{odt}$:
              • Compute a new dynamic shortest path.
              • Assign to each path $k \in K$ the input flow $f^k_{odtp} = \frac{q_{odt}}{k}$.
          If $k > K_{odt}$:
              • Identify the shortest among used paths ($y_{odt}$).
```

- Redistribute the flows as follows:

$$f^k_{odpt} = \begin{cases} \frac{k-1}{k} f^{k-1}_{odpt} & if\ p \neq y_{odt} \\ \frac{k-1}{k} f^{k-1}_{odpt} + \frac{1}{k} q_{odt} & if\ p = y_{odt} \end{cases} \quad \forall o, d, t$$

Step 2 If k reaches a pre-specified maximum number of iterations K_{max} or $RGap \leq \varepsilon$: STOP

Otherwise: Return to Step 1.

In 2007, Mahut, Florian and Tremblay made an interesting comparison between two alternative methodologies for solving DTA equilibrium: the above commented MSA modification and an adaptation of classical convex projected gradient methods. In order to compare them, they modified both algorithms, which was found to be successful (Mahut et al. (2007)). This modification is a powerful contribution of the authors to the area of algorithms for solving DTA, with particular regard to path flow reassignment. Here, we present the phenomena observed by the authors in the different tests that they performed, and the proposed modification for eluding them.

A basic observation on the behavior of the MSA algorithm is that the assignment for specific departure-time intervals is further away from the equilibrium conditions with later departure time intervals. In fact, for most studied real networks, the relation between the departure time and the relative gap is monotonically non-decreasing.

One explanation for this phenomenon is that the travel times of later-departing vehicles are affected by earlier-departing vehicles; and thus the convergence for a later-departing interval cannot be achieved until it has first been achieved for the prior interval. This inherent property of the method suggests the possibility that the higher values of relative gap in the later-departing intervals might be partially a result of the fact that the MSA step-size is the same for all departure-time intervals at each iteration. To state it simply, the idea is that when a latter time interval starts its convergence, the step size does not move a sufficient amount of flow to the shortest path.

Another reason for the increased relative gap value in latter time departure intervals is that the departing vehicle in these time intervals may cause much congestion. Therefore, it is more difficult for the algorithm to achieve equilibrium conditions if congestion increases.

These observations are the basis of a time-varying step-size heuristic proposed by Mahut (2008), which gradually modifies the step-sizes applied to latter intervals.

The heuristic uses an integer reset parameter n, and it is applied to the previously mentioned modified MSA in this section (Mahut et al. (2003)). So, the first K_{odt} iterations remain unchanged. And in the next step of the procedure the modification is the following.

Let D be the number of departure time intervals, and let $t = 0, \ldots, D-1$ be the time intervals in increasing order. For the first $n \cdot D$ iterations, the MSA step size is calculated as Equation 6.2.10 shows.

$$M_k = \begin{cases} \frac{1}{k-\left(\left\lfloor \frac{k-K_{odt}}{n} \right\rfloor + 1\right)n} & if\, t > \left\lfloor \frac{k-K_{odt}}{n} \right\rfloor \\ \frac{1}{k-tn} & else \end{cases} \tag{6.2.10}$$

After the iteration $k = K_{odt} + n \cdot (D-1)$, the step size is calculated: $M_k = \frac{1}{k-tn}$.

The proposed modified MSA, with this step-size adjustment rule significantly accelerated the convergence results. This was observed in the various tests performed on real networks (Stockholm, Montreal, Calgary, ...) by Mahut.

6.3 Proposed Flow Reassignment Method

6.3.1 Justification of the Proposed Method

The main goal of this thesis is to develop a competitive DTA model. In order to achieve this, it is very important to be able to implement a flow reassignment method that efficiently converges to the desired DUE. The proposed iterative flow reassignment algorithm is based on a modification of the MSA.

The main objective of the proposed modification is to overcome some of the limitations observed during our study of the state-of-the art about DTA with MSA for the reassignment component. Among the main drawbacks of the method, we pay attention to two of them previously presented in Section 6.2, which we summarize in the following.

The first observed limitation is that MSA requires the storage of all paths and its flow assignments at each iteration of the process. This can require a lot of memory space in order to perform the usual implementation of the algorithm.

Furthermore, this drawback of the MSA tends to worsen when the size of the network grows or when the level of congestion in the studied network is high. It is for this reason that most of the early works on this subject avoided the use of MSA for large or congested networks.

The second problem is the standard flow reassignment used to divert traffic from the paths used in the last iteration to the current optimum paths. At each iteration of the process (or during some iterations), the MSA adds a new path to the set of used paths for each OD pair and for each time interval. This new path is the one with a lower cost, and the MSA takes into account the cost of the links obtained in the last DNL. So, the objective of the method is to divert some flow from each used path to this new best path. The original MSA does it indiscriminately, and it extracts the same amount of flow from each used path regardless of the path costs, i.e., regardless of whether the path is the worst or only slightly worse than the optimal. This way of flow reassignment is not intuitive, because the paths that have appropriate costs should not suffer the same flow discharge as paths that have high costs, from which we would intuitively remove flow to reduce congestion and consequently to reduce costs.

In the next Section 6.3.2, we propose a new modification of the MSA to improve these two limitations in order to obtain better results for the proposed DTA model.

6.3.2 The Proposed Modification of the MSA

Regarding the previously mentioned drawbacks of the MSA (Section 6.3.1), we develop a flow reassignment algorithm based on another new modification of this method. With this proposal, we try to improve the currently available options proposed in the literature to address the limitations of the algorithm, which sometimes solve one of the problems but not both. Taking into account the proposed solutions (previously specified in Section 6.2), we develop a new MSA that combines some of these modifications with the addition of new ones.

As we mentioned in the previous section, one of the common problems that these methods present is the large computational charge associated with their implementation, because MSA needs to store the new best paths found for each OD for each time interval for each of the iterations. In the state-of-the art (Section 6.2.3), we mention the solution that Mahut et al. (2003) proposed to alleviate this problem, which is based on the idea of limiting the number of alternative paths for each OD pair for each possible departure time interval.

That is why we consider the general scheme formulated by Mahut (2003) as a starting point of our algorithm.

Mahut's algorithm performs the reassignment of the flow in a different way, depending on whether not the maximum number of paths (defined previously) has been reached yet. Or, it performs the reassignment conversely if the maximum has been reached. In the first case, it must recalculate the minimum path using the link costs obtained from the performance of the last DNL. In this case, Mahut assigns the flow equally to all the possible paths (including the new one). In the case of having reached the minimum number of paths, it is not necessary to recalculate any more shortest paths, so the set of paths remains stable until the end of the procedure. From this point forward, the flow is distributed among the possible paths using a classic MSA scheme.

We can say that in this case the problem with path storage is solved. However this solution does not take into account the different costs of each path when it reassigns the flow, i.e., the second MSA drawback still holds. That is why our method uses the same idea of dividing the process into two parts, according to whether or not the maximum number of paths for each OD pair for each time interval is reached; but, the presented flow reassignment for each iteration is different from the proposal by Mahut et al. Here is where we incorporate the second and most important modification, which is developed to eliminate the second MSA problem. Therefore, we propose an algorithm that solves both conflicts simultaneously.

In the classical version of the MSA, the MSA parameter used to distribute the flow among the possible paths depends on the current iteration of the assignment process, and usually it is equal to the inverse of the number of iterations (i.e., $\frac{1}{k}$). In that case, as already mentioned, the diversion of flow to the best path from the remaining paths is made indiscriminately, without taking into account the cost of all the possible paths.

To overcome this limitation, Varia and Dhingra (2004) (Section 6.2.3) proposed a modification to the procedure using a new factor based on a logit distribution of demand flow and according to instantaneous travel time on the corresponding paths. So the reassignment explicitly takes into account the cost of the alternative paths when it diverts the flow. Thus, the main part of the flow assigned to the new shortest path comes from the worst paths, i.e., from the paths with higher costs.

It is important to note here that improvements in performance allowed by the algorithm proposed by Varia and Dhingra (2004) are negligible in comparison those that could have been induced by a classical approach. This is because the logit factor that Varia and Dhingra proposed is based on instantaneous path travel times rather than on the actual path travel times[3].

The main idea of the proposed MSA modification is to complement the usual parameter of MSA in some specific parts of the proposed scheme, trying to take advantage of the information from the previous DNL (the other main component of the DTA model). Therefore, when at a certain iteration of the process a new shortest path is found, the proposed factor improves the method by taking into account the cost of the alternative paths (all the paths belonging to the set of paths used in the previous iteration) in the flow reassignment.

Looking at the commented proposal of Varia and Dhingra (2004), this new factor (called in the following a diversion factor) is based on a logit distribution, but in this case it follows the actual costs of alternative paths. The method considers the costs based on the link actual travel times obtained by the DNL in the previous iteration of the procedure. Thus, the expected improvements will be significant in comparison to the results obtained through a classic MSA procedure.

The diversion factor (δ_{odpt}) is defined for each path p from origin o to destination d departing at time interval t as Equation 6.3.1 shows.

$$\delta_{odpt} = \frac{\exp\left(-c_{odpt}\left(tt_{odpt}\right)\right)}{\sum\limits_{p} \exp\left(-c_{odpt}\left(tt_{odpt}\right)\right)} \tag{6.3.1}$$

where c_{odpt} is the cost of the path p based on the actual travel times tt_{odpt} of the path.

Finally, we comment on some ideas about the first iteration of the proposed flow reassignment method. At the initialization step, we calculate a static shortest path for each OD pair. Here, we do not need to calculate a time-dependent shortest path for each OD pair for each departure time interval, because we base the calculation on the free flow link travel times. Obviously, all travel times are the same for all time intervals. In this way, we can use a static shortest path algorithm in the initialization. We can also calculate the same set of paths for each departure time interval for each OD pair.

[3]Actual path travel times: Sum of the link travel times along the path estimated at the time when driver enters each link.

We also propose calculating more than one path for each OD pair in the first step. This is because, if we start with only one possible path for assigning all the flow of each OD pair, the possibility of generating false congestion is very high. If this occurs, all the main links of the network can present congestion and, consequently, very high costs. In the next step of the process we find a shortest path that does not use these congested links, so we can go very far from the equilibrium solution and will need more iterations in order to converge. However, if we start with a small set of paths for each OD pair and assign the flow inversely proportional to each path cost, then we will need fewer iterations to achieve equilibrium. The number of initial paths (M) depends on the network characteristics. The most convenient way would be to test different options during the network calibration process. (see Chapter 7).

In summary, an adaptation of the MSA that combines the following two specified solutions is presented:

- Limit the maximum number of available paths for each OD pair for each departure time interval t, in order to reduce the computational storage needed in the original MSA.
- Use a diversion factor based on actual travel times in the reassignment process in order to perform a more realistic flow reassignment for the alternative paths.

6.3.2.1 Global Scheme of the Proposed Flow Reassignment Algorithm

In the following scheme, we show the specific proposed algorithm:

1. Initialization ($k = 1$)
   (a) Static shortest path calculation based on costs according to free flow link travel times . All path sets P_{odt} are generated with the same number of paths defined previously (M).
   (b) Initial flow assignment (inversely proportional to the path costs).
   (c) DNL to obtain an initial solution.
   (d) Update iteration count ($k = k + 1$).

2. Path Flow Reassignment

   For all OD pairs (o, d), for all departure time intervals t

   - If the maximum number of paths is not achieved $\left(\left|P_{odt}^{k-1}\right| < N_{odt}\right)$
     (a) Time-dependent shortest path (y_{odt}) calculation based on the link costs according to the actual link travel times obtained in the last DNL.
     (b) Path flow reassignment:
       - If the shortest path is new$\left(y_{odt} \notin P_{odt}^{k-1}\right)$
         i. Assign the flow f_{odpt}^{k} $\forall p \in P_{odt}^{k-1}$ and on y_{odt} following (a).
         ii. Update path set: $P_{odt}^{k} = P_{odt}^{k-1} \cup y_{odt}$.
         iii. Update number of paths.
       - Else $\left(y_{odt} \in P_{odt}^{k-1}\right)$
         i. Assign the flow f_{odpt}^{k} $\forall p \in P_{odt}^{k-1}$ following (b).
         ii. Update path set: $P_{odt}^{k} = P_{odt}^{k-1}$.
   - Else (the maximum number of paths is achieved) $\left(\left|P_{odt}^{k-1}\right| \geq N_{odt}\right)$
     (a) Identify the shortest path (y_{odt}) among those already used P_{odt}^{k-1}.
     (b) Path flow reassignment:
       i. Assign the flow f_{odpt}^{k} $\forall p \in P_{odt}^{k-1}$ following (b).
       ii. Update path set: $P_{odt}^{k} = P_{odt}^{k-1}$.

3. Dynamic Network Loading

   (a) Flow propagation using mesoscopic simulation.
   (b) Update link travel times and, consequently, their costs.

4. Convergence criteria

   - If the maximum number of iterations (K_{max}) is reached $(k = K_{max})$ or the *RGap* is satisfied $\Longrightarrow$ STOP.
   - Else
     (a) Update iteration count $(k = k + 1)$.
     (b) Go to Step 1.

6.3.2.2 Proposed Flow Reassignment Options

Considering the algorithm proposed in the previous section, now it is time to define the different flow reassignment processes for each of the possibilities:

- Flow reassignment (a)

  If the calculated shortest path (y_{odt}) is new, the flow reassignment proposed for it is shown in Equation 6.3.2.

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} & if\ p = y_{odt} \\ (1-\lambda_k)\, q_{odt}\delta_{odpt} & if\ p \neq y_{odt} \end{cases} \tag{6.3.2}$$

  where:

  f_{odpt}^{k} is the flow assigned to the path p departing at time interval t at iteration k.

  λ_k is the MSA parameter depending on the iteration k.

  q_{odt} is the demand from o to d entering the network during the time interval t.

  δ_{odpt} is the diversion factor.

  c_{odpt} is the cost of the path p based on the actual travel times tt_{odpt} obtained by the DNL performed at the previous iteration $k-1$.

- Flow reassignment (b)

  If the calculated shortest path (y_{odt}) belongs to the previous iteration path set or if no more paths are calculated because the maximum number of paths for each OD pair for each interval is achieved, the flow reassignment proposed for it is shown in Equation 6.3.3. In the second case, y_{odt} is the best path among those already used.

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} + (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p = y_{odt} \\ (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p \neq y_{odt} \end{cases} \tag{6.3.3}$$

Depending on the different proposed values for the MSA parameter (λ_k), we have some different results. Our proposal is the standard $\lambda_k = \frac{1}{k+1}$, but other options have also been tested. In the next Section 6.4, we propose an experiment design which includes three different options for the MSA parameter.

6.4 Computational Experiences

In the previous Section 6.3, we presented a modification of the MSA algorithm. The objective of this section is to test the performance of this proposed flow reassignment approach. In particular, we want to verify the quality of the convergence of the proposed method measured by the number of DTA iterations needed to reach convergence. Moreover, we take advantage of this computational experience to verify the correct selection of the MSA parameter $\lambda_k = \frac{1}{k+1}$.

In order to perform an evaluation of the flow reassignment method without taking into account the other components of the presented DTA scheme, we propose integrating the new proposed MSA into an external DTA environment.

During the development of this thesis, we collaborated with the Division of Transportation and Logistics of the Department of Transportation Sciences of the Royal Institute of Technology (Kungliga Tekniska Högskolan - KTH) in Stockholm. They had researched widely the topics referred to in this work, with developments like their hybrid mesoscopic-microscopic simulation model, which used their own proposal for the mesoscopic component: MEZZO. So, we considered it appropriate to take the opportunity to work with our flow reassignment method in a MEZZO environment and integrate the DNL and the shortest path algorithm proposed in MEZZO.

In Section 6.4.1, we begin by briefly describing the MEZZO model and its assignment structure. Then, in Section 6.4.2, we discuss several assumptions (about our different structure assignment) that were employed to work around practical constraints and considerations. After this, in Section 6.4.3, we explain in detail the computational tests performed, and finally in Section 6.4.4 we summarize the analysis of the obtained results.

6.4.1 MEZZO

As explained in Section 3.2.12, MEZZO is a mesoscopic simulator developed by W. Burghout in 2004. It is an event-based simulator with changes in traffic states calculated only when something happens (like in Dynameq). Events in MEZZO are defined by vehicles entering a link, exiting a link, making a new route choice, etc.

It was developed as the mesoscopic component of a hybrid mesoscopic-microscopic simulation model. The use of event-based simulation in the mesoscopic model

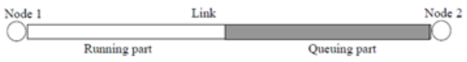


Figure 6.4.1: MEZZO link structure.

provides a natural way of synchronizing with the usually time-based microscopic models. Each time-step of the microscopic model enters as an external event in the event list of the mesoscopic model.

MEZZO is a synthesis of a number of models and processes that capture all the important aspects of the operations of traffic networks. These models and procedures include the network representation, the movement of vehicles in links and intersections, and travel behavior (route choice).

6.4.1.1 Dynamic Network Loading in MEZZO

The model is based on the familiar queue-server paradigm, where the main assumption is that queues are only generated from nodes, not inside links. At each time instant, the link from Node 1 to Node 2 is divided into two sections: a running section and a queuing section (Figure 6.4.1).

The boundary between the queuing section and the running section is dynamic. This means that the queue length may change, depending on the inflow of the vehicles through Node 1 and the outflow through Node 2. In the extreme cases, the queue fills the entire link (and possibly blocks back onto other links) or it vanishes. The running segment contains vehicles that run at a steady traversal speed, which is chosen upon entering the link as a function of the density of the running segment at that moment. Using the traversal speed, an earliest exit time is calculated for the vehicle. At any given time t, the vehicles whose earliest exit time is smaller than t and are still on the link are considered to be part of the queuing section. Conversely, all vehicles whose earliest exit time is greater than t are part of the running section. At the queue exit, the node has one turn server for each turning movement. The turn server determines the rate at which the vehicles are taken from the queue and transferred to the next link.

6.4.1.2 Assignment in MEZZO

The assignment structure in MEZZO is documented, as we show in Figure 6.4.2.

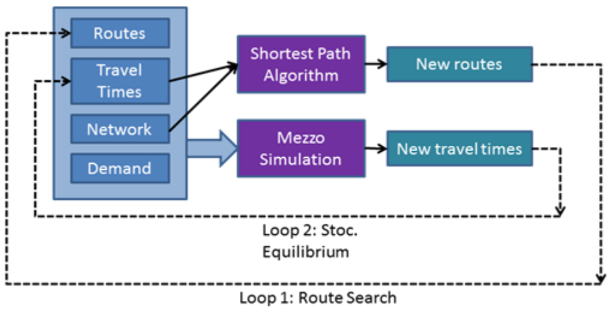


Figure 6.4.2: MEZZO structure.

This is done in the following way. Starting from the free-flow travel times for each link, a shortest path is calculated for each OD pair by using a label correcting algorithm. Then, a small random noise is generated, is added to the link travel times, and each route search is repeated a number of times with different draws of the random noise components. This results in multiple paths being available at the start of the first iteration. After this, each OD pair demand is assigned to those paths in proportion to their travel time. Then, using all this information, the MEZZO DNL is executed and it generates new route travel times and consequently new route assignments.

The route assignments are saved, as well as the resulting link travel times, and these are used as input for the DNL in the next iteration. The historical travel times for iteration $n + 1$ are calculated as a moving average of the historical travel times for iteration n and the resulting link travel times of iteration n. The iterations come to an end when the historical travel times and resulting travel times are equal. During this part of the process the number of paths for each OD pair does not vary. (See Loop 2 in Figure 6.4.2).

When travel time convergence achieves Loop 2, the process calls Loop 1, which generates new routes and adds them to the set of known routes. Then, Loop 2 is run another time. So, Loop 1 contains Loop 2 as an inner loop, making sure that route searches are conducted only on converged travel times.

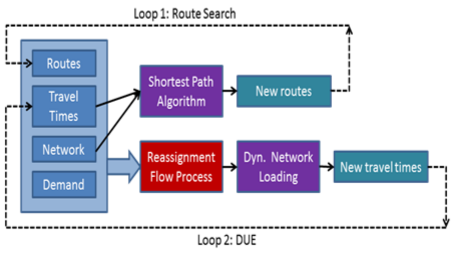


Figure 6.4.3: Proposed DTA structure.

6.4.2 Integration

6.4.2.1 Decisions about the Integration

In Chapter 4, we presented our computational framework for DTA that considerably differs from the MEZZO assignment structure. In order to use the different components of the Mezzo structure, our assignment is proposed in the following way.

Starting from the free-flow travel times for each link, a shortest path is calculated for each OD pair. Then, a small random noise is generated and is added to the link travel times. Now, each route search is repeated $k-1$ times with different draws of the random noise components (with k as the predefined number). This results in k paths being available for each OD pair at the start of the first iteration (initialization iteration). And all vehicles are assigned to these shortest paths in proportion to their travel times.

Then, using all this information, the DNL is executed generating new link travel times. These times are used as input in the next iteration, which starts by calculating a new shortest path for each OD pair (only one). The existing route sets are augmented with these new shortest routes if they are found, and each OD pair demand is assigned into those paths following the proposed flow reassignment process: the modified MSA. The current iteration continues the DNL with these new route sets and route flows. The iterations come to an end when the DUE convergence is achieved (measured with the relative gap).

Inspired by the MEZZO scheme, our assignment structure is shown in Fig. 6.4.3.

In this case, Loop 1 generates new routes and adds them to the set of known routes, and Loop 2 calculates the new travel times by executing the DNL. Obviously, the process is different than the MEZZO approach, because here "Loop 2" is not a real loop, because we only execute one iteration of Loop 2 into each iteration of Loop 1.

The second difference between the two processes is that we need an independent module for the flow reassignment process. In the MEZZO model, the route assignment is performed instantaneously when a new vehicle is generated in the DNL process (MEZZO simulation). This is because it is very simple (the route flow is proportional to the route travel time), so a separate module is not needed for its treatment.

Due to these differences, we have decided to use for this computational experiment only some parts of the MEZZO model: the shortest path algorithm and the DNL (MEZZO simulation). Consequently, we finally implement all our DTA process as a separate function of the DTA MEZZO, but have also included it in the same project in order to take advantage of all the MEZZO features.

6.4.2.2 Computational Integration Details

The the MEZZO model was implemented in C, and it was strictly object oriented, both in design and implementation. So, all the new developments in the MEZZO Project have been in C. In order to use the maximum number of MEZZO functions, we have not added any new class in the MEZZO project. We take advantage of the existing classes by adding new attributes and functions to them.

We consider that the specific details of all the new functions and the attributes are not pertinent to this thesis.

6.4.3 Computational Tests and Results

In the previous section, we presented an iterative scheme for the resolution of the DTA problem, which includes a new flow reassignment process based on a modification of the MSA algorithm. The objective now is to use a real example to carry out a primary performance test of the proposed approach.

With this aim, we design a set of experiments that test the proposed flow reassignment approach. This set is based on a factor design that takes into account three different elements:

- The incorporation of the proposed diversion factor into the classical MSA approach.
- The use of the limitation on the number of paths for each OD pair for each departure time interval.
- The MSA parameter used.

To this end, we compare the proposed flow assignment method (including the diversion factor) with other standard MSA approaches existing in the literature. Moreover, to test the goodness of the limitation about the number of paths, we propose experiments with and without this bound. Ultimately, in order to quantify the importance of the MSA parameter in the proposed flow reassignment approach, we test three different options for λ_k.

It is important to note here that the flow reassignment method proposed in Section 6.3.2 coincides with the experiment that considers the diversion factor, takes into account the limitation about the number of paths and uses the MSA parameter $\lambda_k = \frac{1}{k+1}$. So, we want to test if it is the best option of all the possible combinations.

With this objective, all the different methods obtained through the different combinations of the design factors are embedded into the DTA scheme presented before (see Section 6.4.2) and executed for the studied real network. Finally, the results obtained with all of these are compared to each other.

We begin by briefly describing the studied network and the demand data used for the computational experiment, and we proceed to outline the experiment design. Then, we present results for the DTA process that verify the performance and feasibility of the flow reassignment method that we propose in this chapter.

6.4.3.1 Example Dataset

Network Description

A set of computational experiments is conducted with an example network corresponding to the Södermalm district in central Stockholm (Figure 6.4.4), with the infrastructure corresponding to 2007. This network, depicted in Figure 6.4.5, has 1,101 sections, 409 intersections, 168 centroids and 462 OD pairs.

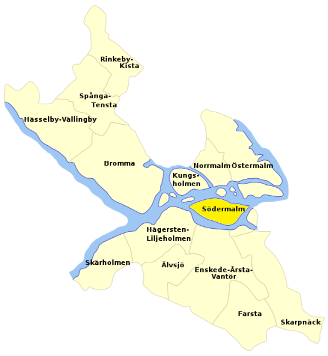


Figure 6.4.4: Södermalm district in central Stockholm.

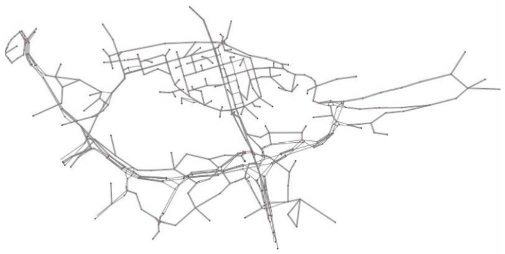


Figure 6.4.5: Södermalm MEZZO model.

Data Description
The main input of the DTA model is time-dependent demand. In this case, the performed experiments use a synthetic demand that assigns to the Södermalm network the total flow of 34,451 trips. The demand is split into eight different time-dependent OD matrices corresponding to eight different time slices of 15 minutes.

One of the common MEZZO input files contains the description of the set of known routes from each origin to each destination (chains of links). Multiple paths per OD pair are possible, and the file is augmented by MEZZO if new shortest paths are found. In this case, we prefer to start the process without this information; so we suppose no routes are available and an empty file is be provided. As shown in the previously proposed algorithm, the process calls the shortest path algorithm to initialize this information before the first DNL execution.

The other necessary input files that are common in the MEZZO environment remain unchanged, i.e., we use the files attached to the Södermalm network.

6.4.3.2 Experiment Design

An experimental design is conducted in order to evaluate the performance of the proposed flow reassignment method. The selected design factors are:

- Flow reassignment option.
- MSA parameter.
- Limited-Unlimited number of paths.

We compare the different DTA solutions achieved with the proposed algorithm when we use each of the combinations among the design factors.

Flow reassignment
The performed experiments take into account two different flow reassignment options. As we showed in Section 6.3.2.1, the presented MSA scheme needs two possible assignments for each option ((a) and (b)), depending on whether a path belongs to the current path set or not. So, we propose these two possibilities for the two proposed flow reassignment options, which are shown as follows:

- The flow reassignment option proposed in Section 6.3.2.2 (from now on "Flow Reassignment 1"):
  - Flow Reassignment (a)

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} & if\ p = y_{odt} \\ (1-\lambda_k)\, q_{odt}\delta_{odpt} & if\ p \neq y_{odt} \end{cases} \tag{6.4.1}$$

  - Flow Reassignment (b)

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} + (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p = y_{odt} \\ (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p \neq y_{odt} \end{cases} \tag{6.4.2}$$

- A flow reassignment inspired by other literature options proposed in Section 6.3.2.2 (from now on "Flow Reassignment 2"):
  - Flow Reassignment (a)

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} & if\ p = y_{odt} \\ (1-\lambda_k)\left(\frac{q_{odt}}{N_{odt}}\right) & if\ p \neq y_{odt} \end{cases} \tag{6.4.3}$$

  - Flow Reassignment (b)

$$f_{odpt}^{k} = \begin{cases} \lambda_k q_{odt} + (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p = y_{odt} \\ (1-\lambda_k)\, f_{odpt}^{k-1} & if\ p \neq y_{odt} \end{cases} \tag{6.4.4}$$

We can see that flow reassignment (b) is the same for both options (1 and 2), so the difference between both approaches concerns flow reassignment (a).

MSA parameter

The performed experiments take into account the following three different MSA parameters (λ_k):

- $\lambda_k^A = \frac{1}{k+1}$

- $\lambda_k^B = \frac{1}{2}$
- $\lambda_k^C = \frac{k}{k+1}$

The first one $\left(\lambda_k^A\right)$ is a standard option used by most methods in the literature. It is an MSA parameter that depends on the current iteration of the method and satisfies Equation 6.2.8. These two conditions ensure the MSA theoretical convergence.

With the second proposal $\left(\lambda_k^B\right)$, we want to test what happens if the parameter does not depend on the current iteration, i.e., it is a fixed diversion parameter within the new path and the remaining paths of the path set. So, we propose a surprising parameter which diverts half of the total demand to the new path and distributes the other half of the total demand among all the other paths of the set. It is important to mention that this parameter does not satisfy Equations 6.2.8.

Finally, we want to test the importance of using an MSA parameter that satisfies Equation 6.2.8. With this aim, we propose an MSA parameterλ_k^C that depends on the current iteration, that does not satisfy these two conditions, and with which we want to test the “bad”convergence of the MSA in this case.

Limited-Unlimited number of paths

One of the important changes in the proposed modified MSA is the limitation of the maximum number of available paths for each OD pair for each time interval. This is to reduce the computational storage needed in the original MSA. In order to test this adjustment, the experimental design takes into account two specific situations. While the first one does not consider this limitation, the second strictly follows the DTA scheme proposed with a limited number of paths.

In this case, the quantity of bound paths is considered to be five, in accordance with the network characteristics. During the network calibration process included into this computational experienc, we test different options. Finally, we consider that five paths for each OD pair for each departure time interval is a good bound for the Södermalm network.

Proposed Computational Experiments

Table 6.1 summarizes the proposed experiments, detailing the design factor combinations.

EXPERIMENT DESIGN			
Experiment	**Path Limitation**	**Flow Assignment**	λ_k
Y1A	yes	1	A
Y1B	yes	1	B
Y1C	yes	1	C
Y2A	yes	2	A
Y2B	yes	2	B
Y2C	yes	2	C
N1A	no	1	A
N1B	no	1	B
N1C	no	1	C
N2A	no	2	A
N2B	no	2	B
N2C	no	2	C

Table 6.1: Set of the proposed experiments.

6.4.3.3 Computational Results

In this section we present the results of the DTA process executed for each of the proposed experiments. These results verify the performance and feasibility of the flow reassignment method proposed in Section 6.3.2. First, we present the Janson relative gap results in order to have a global idea of the good performance of the process. Then, using the refined relative gap proposed by Mahut, we try to refine the conclusions by taking into account the different departure time intervals in the analysis of the obtained results.

Relative Gap

The proposed DTA experiments are run for 16 iterations. The 120-min loading interval is divided into eight time intervals for the proposed MSA assignment algorithm. After each iteration, the relative gap proposed by Janson in 1991 (Section 4.3.6) is calculated. Table 6.2 presents the relative gap for each executed experiment after each iteration:

Figure 6.4.6 shows the results for the all experiments together.

It can be observed that the results for the MSA parameter $\lambda_k = \frac{k}{k+1}$ are the most unstable and they do not seem to converge. Figure 6.4.7 shows the results if we

	Experiment											
It	**Y1A**	**Y2A**	**N1A**	**N2A**	**Y1B**	**Y2B**	**N1B**	**N2B**	**Y1C**	**Y2C**	**N1C**	**N2C**
0	0.3054	0.3101	0.2993	0.3149	0.3139	0.3095	0.2926	0.3002	0.3012	0.3003	0.3064	0.2772
1	0.1529	0.1484	0.1357	0.1392	0.1484	0.1493	0.1385	0.1396	0.1505	0.1454	0.1331	0.1298
2	0.0658	0.0993	0.0463	0.0994	0.0755	0.0712	0.0380	0.0648	0.0361	0.0502	0.0824	0.0368
3	0.0391	0.0672	0.0459	0.0546	0.0292	0.0421	0.0450	0.0296	0.0684	0.0594	0.0703	0.0605
4	0.0283	0.0470	0.0324	0.0436	0.0452	0.0330	0.0510	0.0464	0.0299	0.0344	0.0212	0.0641
5	0.0227	0.0332	0.0280	0.0290	0.0228	0.0313	0.0234	0.0196	0.0359	0.0362	0.0164	0.0227
6	0.0207	0.0453	0.0303	0.0360	0.0132	0.0285	0.0146	0.0168	0.0607	0.0295	0.0438	0.0357
7	0.0199	0.0395	0.0197	0.0319	0.0231	0.0238	0.0136	0.0234	0.0247	0.0346	0.0301	0.0500
8	0.0167	0.0329	0.0146	0.0299	0.0084	0.0238	0.0159	0.0207	0.0211	0.0483	0.0422	0.0454
9	0.0225	0.0324	0.0154	0.0264	0.0151	0.0101	0.0216	0.0177	0.0842	0.0208	0.0288	0.0499
10	0.0160	0.0231	0.0126	0.0182	0.0115	0.0116	0.0107	0.0100	0.0325	0.0423	0.0367	0.0409
11	0.0187	0.0226	0.0147	0.0225	0.0155	0.0106	0.0063	0.0130	0.0329	0.0334	0.0650	0.0412
12	0.0187	0.0200	0.0131	0.0195	0.0192	0.0152	0.0235	0.0189	0.0199	0.0599	0.0480	0.0538
13	0.0153	0.0187	0.0133	0.0177	0.0154	0.0169	0.0210	0.0066	0.0301	0.0311	0.0478	0.0878
14	0.0144	0.0171	0.0129	0.0170	0.0136	0.0131	0.0226	0.0108	0.0339	0.0522	0.0240	0.0395
15	0.0137	0.0164	0.0105	0.0161	0.0112	0.0106	0.0180	0.0175	0.0787	0.0373	0.0724	0.0454

Table 6.2: Relative gap for each iteration for each executed experiment.

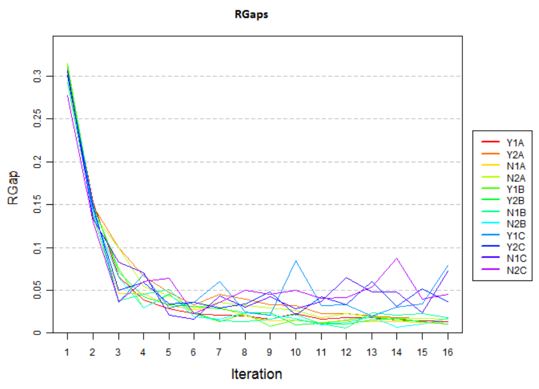


Figure 6.4.6: Relative gap for each experiment after each DTA iteration.

do not consider the experiments that included the mentioned MSA parameter option.

Now, we see that for all the combinations of flow reassignment methods (1 and 2) and MSA parameters ($\lambda_k = \frac{1}{2}$ and $\lambda_k = \frac{1}{k+1}$), the DTA process converges quickly and stably.

In Figure 6.4.8, graphic $N1$ shows the results for the experiments with unlimited number of paths for each OD pair for each interval, with the first proposed reassignment flow method (1) combining the three different MSA parameters (A, B and C). The same experiments were run with changes to the flow reassignment method to the second proposal (2). The results are shown in graphic $N2$.

Graphics $Y1$ and $Y2$ show the results for the same previous experiments with limited number of paths of each OD pair for each interval (5 paths).

These disaggregated results clearly show that the experiments performed with parameter $\lambda_k = \frac{k}{k+1}$ do not have good behaviour. Regarding the limitation on the number of paths, there are no significant differences between limiting or not. Nor are there differences between the two reassignment flow methods. Perhaps slightly better results could be obtained for the experiment performed with $\lambda_k = \frac{1}{2}$, combined with the second flow reassignment method.

In Figure 6.4.9, graphics NA and YA show the results for the experiments performed with unlimited and limited numbers of paths for each OD pair for

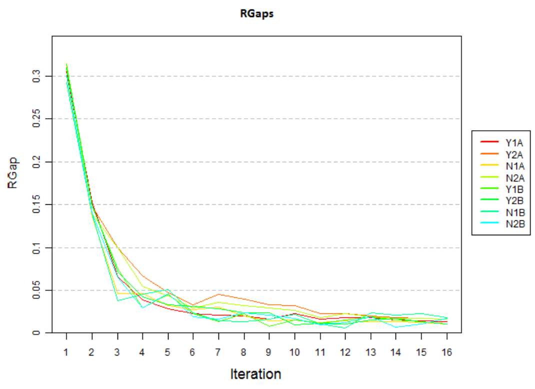


Figure 6.4.7: Relative gap for some experiments after each DTA iteration.

each interval (5 paths), with the first proposed MSA parameter $\left(\lambda_k = \frac{1}{k+1}\right)$ combining the two flow reassignment methods (1 and 2). The results for the same experiments with the second proposed MSA parameter $\left(\lambda_k = \frac{1}{2}\right)$ are shown in graphics NB and YB.

Looking at these results in terms of the relative gap, we may conclude that both reassignment methods have a similar rate of convergence that is or is not independent of the limitation to the number of paths. We only can consider that by using the first MSA parameter ($\lambda_k = \frac{1}{k+1}$), the first option converges faster than the second one. However, we cannot draw any conclusion for parameter $\lambda_k = \frac{1}{2}$.

Refined Relative Gap

The proposed DTA experiments are run for 15 iterations. The 120-min loading interval is divided into eight time intervals for the proposed MSA assignment algorithm. After each iteration, the refined relative gap proposed by Mahut in 2003 (see Section 4.3.6) is calculated for the vehicles departing from the origin during each of these intervals (see Figure 6.4.10 for the color legend of the intervals).

In Figure 6.4.11, graphics $Y1A$, $Y1B$ and $Y1C$ show the results for the experiments with a limited number of paths for each OD pair for each interval (5 paths). The first proposed reassignment flow method (1) combines the three

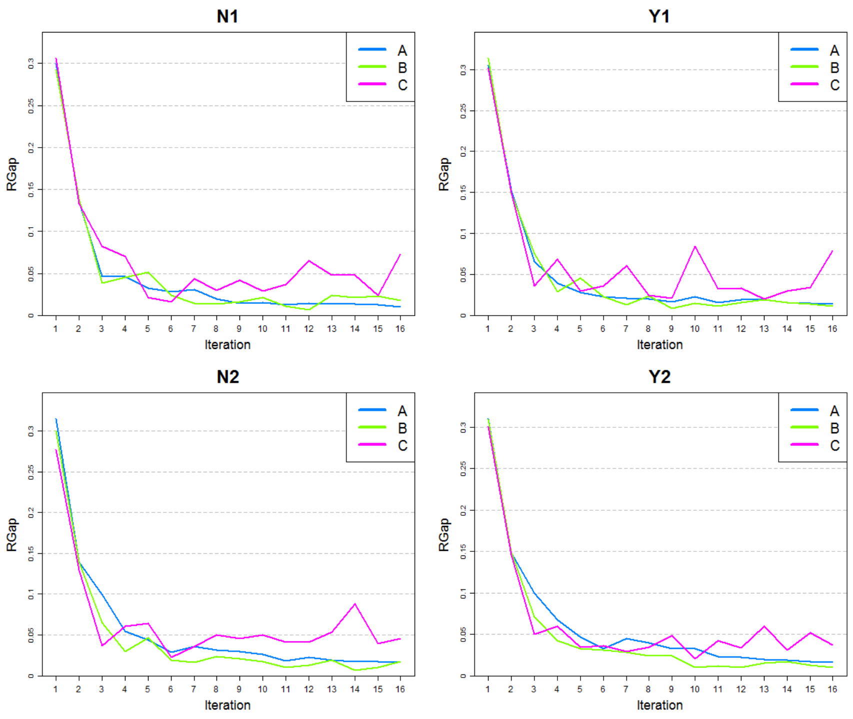


Figure 6.4.8: Relative gap of the experiments after each DTA iteration by differentiating the use of each MSA parameter option.

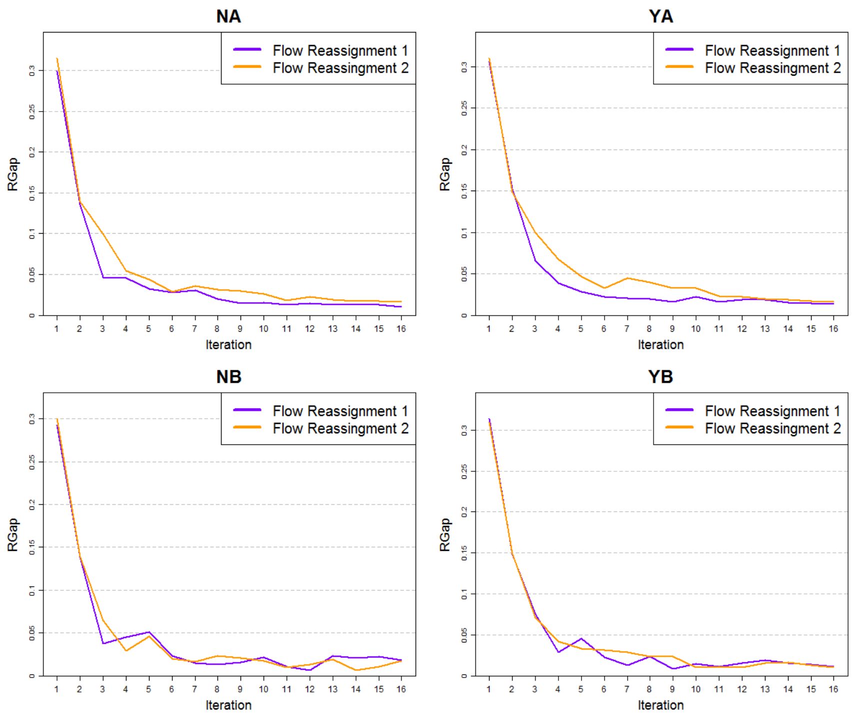


Figure 6.4.9: Relative gap of the experiments after each DTA iteration differentiating the use of each flow reassignment option.

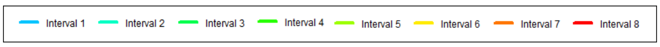


Figure 6.4.10: Color legend for the eight departure time intervals.

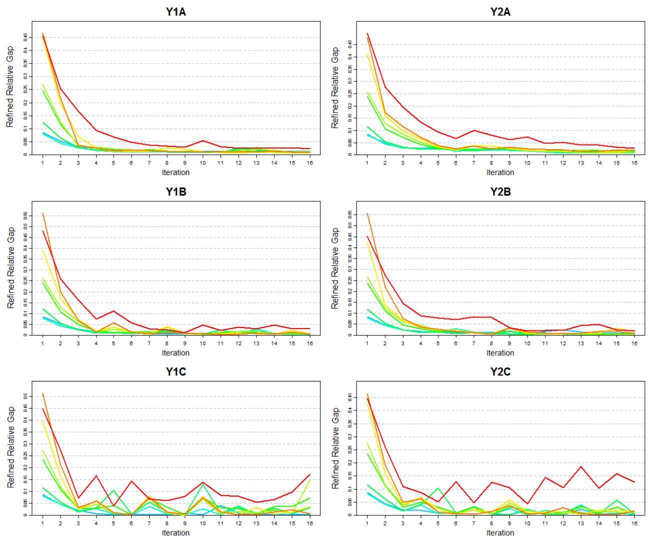


Figure 6.4.11: Limited number of paths refined relative gap experiments.

different MSA parameters (A, B, C). The same experiments were run but with changes to the flow reassignment method to the second proposal (2). The results are shown in graphics $Y2A$, $Y2B$ and $Y2C$.

With respect to the MSA parameter, the main observation is that, as expected, the MSA parameter $\lambda_k = \frac{k}{k+1}$ does not work well. It presents oscillations for several of the departure intervals in the last executed iterations of the MSA process. In contrast, the other two proposed options seem much more stable and we can distinguish the good performance of the option $\lambda_k = \frac{1}{k+1}$.

With respect to the flow reassignment options, if we consider $\lambda_k = \frac{1}{k+1}$, the first option improves the results produced by the second one. For the other combinations, the results are very similar for both proposals. It can also be observed that the results for the last departure time interval (red interval 8) are worse than the other departure intervals for all the proposed experiments (which is theoretically expected, as explained in previous sections). It appears that the new reassignment flow method (1) mitigates this effect.

In Figure 6.4.12, graphics $N1A$, $N1B$ and $N1C$ show the results for the experiments with an unlimited number of paths for each OD pair for each interval, with

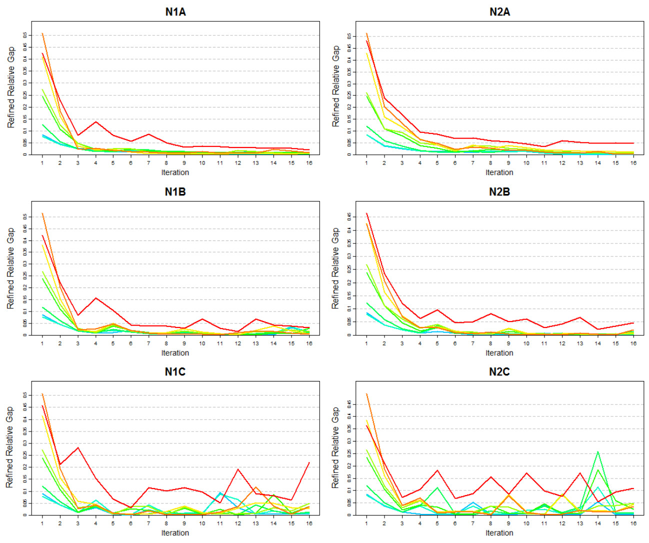


Figure 6.4.12: Unlimited number of paths refined relative gap experiments.

the first proposed flow reassignment method (1) combining the three proposed MSA parameters (A, B, C). The same experiments were run but with changes to the flow reassignment method to the second proposal (2). The results are shown in graphics $N2A$, $N2B$ and $N2C$.

In this case, the bad performance of the MSA parameter $\lambda_k = \frac{k}{k+1}$ is reflected another time in the results. We also repeat the same considerations for the other experiments with one exception: the first flow reassignment method works worse in this case (with unlimited number of paths), because it presents more instability for the last departure time interval.

Simultaneously analyzing the presented result sets, it could be said that the results of the experiments that have limitations in the number of paths are a bit more stable than the results of the experiments that are not restricted.

6.4.4 Conclusions

In the relative gap case, the quality of the solution achieved with the new flow reassignment method does not depend very much on whether or not we use a limited number of paths for each OD pair for each departure time interval. Therefore, the high computational storage that is typical with classical MSA is reduced without losing the quality of the reassignment solution.

However, the selected option of the MSA parameter has proved to be basic for the good quality of the solution. The solutions obtained for the experiments that use the MSA parameter $\lambda_k^C = \frac{k}{k+1}$ are as bad as we expected when we decided on this value for the MSA parameter, which does not satisfy the Equation 6.2.8. On the other hand, combining the MSA parameter $\lambda_k^A = \frac{1}{k+1}$ with flow assignment 1 is more stable than combining this flow assignment with the fixed MSA parameter $\lambda_k^B = \frac{1}{2}$. If the combination includes the other flow assignment (2), then, both MSA parameter options provide similar results.

If we now pay attention to the flow assignment, we can conclude that both methods have a similar rate of convergence. The first option converges faster than the second one when using the MSA parameter $\lambda_k^A = \frac{1}{k+1}$. However, we cannot draw any conclusions for parameter $\lambda_k = \frac{1}{2}$.

In summary, the best combination corresponds to experiment $Y1A$: flow reassignment 1 with a limited number of paths for each OD pair for each departure time interval with MSA parameter $\lambda_k^A = \frac{1}{k+1}$.

Regarding the refined relative gap results, the first observation is that the MSA parameter $\lambda_k^C = \frac{k}{k+1}$ does not work well, as expected. It presents severe oscillations for some of the departure time intervals, especially in the last executed iterations of the process. This phenomenon occurs with or without limitations in the number of paths for each OD pair for each departure time interval. The other two MSA parameters seem more stable, although the variable option $\lambda_k^A = \frac{1}{k+1}$ is the best option. Moreover, it is important to note here the successful performance of the experiments with the MSA parameter λ_k^B, which does not depend on the iteration number and does not satisfy Equations 6.2.8. However, it has been observed that the two proposed flow assignment methods become more unstable with this MSA parameter option, particularly in the latter iterations.

With respect to the flow assignment options, the first proposal (1) improves the results obtained by the second method. This proposal is faster than the other option, and an accepted relative gap is achieved some iterations earlier. This improvement is particularly notable when using the MSA parameter $\lambda_k^A = \frac{1}{k+1}$ with limitations in the number of paths ($Y1A$). In this case, the first proposal achieves good refined relative gap results (less than 0.055, Tong and Wong(2000)) in about half as many iterations as the other tested flow assignment method ($Y2A$).

The bad convergence of the latter departure time intervals (which was theoretically expected, as explained in previous Section 6.2.3) is clearly observed in the results from all the experiments performed on the refined relative gap in the Södermalm network (red interval 8). The first proposed method (1) mitigates this effect by using the diversion factor which enhances the flow reassignment among the alternative paths. In this way, no other sophisticated solutions, like the previously mentioned time-varying step-size adjustment, are required.

Moreover, concerning the refined relative gap, both flow assignment methods are better at limiting the maximum available paths for each OD pair for each interval.

In summary, the best combination corresponds to experiment $Y1A$: flow reassignment 1 with a limited number of paths for each OD pair for each departure time interval with MSA parameter $\lambda_k^A = \frac{1}{k+1}$. Therefore, we propose that this combination be included in the developed DTA model.

6.5 Summary and Contributions

In this chapter a new modification of the Method of Successive Averages has been developed in order to work around the improvement of the reassignment flow module of the proposed DTA scheme.

First of all, this chapter has reviewed flow reassignment methods for possible use in a DTA scheme. The discussion has focused exclusively on the algorithms proposed in the literature to solve a set of variational inequalities under a preventive approach, which results in a DUE solution. Such methods are: projection methods, alternating directions methods, and modifications of the popular MSA. A wide range of published research has been summarized.

Taking into account the objective of this thesis, it has been essential to implement a method of flow reassignment that converges efficiently to the DUE. Therefore, the proposed flow reassignment algorithm based on MSA was developed. The new method tries to overcome some detected drawbacks of the MSA by combining the following two specified solutions:

- Limit the maximum number of available paths for each OD pair for each departure time interval, in order to reduce the computational storage needed in the original MSA.

- Use a diversion factor based on actual travel times in the reassignment process, in order to make a more realistic flow reassignment among the alternative paths.

Then, a computational experiment was runin order to test the performance of the proposed flow reassignment approach. We have proposed integrating the new MSA into an external DTA environment (MEZZO) in order to evaluate the flow reassignment method without taking into account the other components of the presented DTA scheme.

Finally, we have run a computational experiment in order to test the feasibility of the proposed flow reassignment approach. With this aim, we have designed a set of experiments to execute on the real network of Södermalm in Stockholm. These tests incorporated the proposed diversion factor into the classical MSA approach and they used a limited number of paths for each OD pair for each departure time interval, as well as different MSA parameter values. Ultimately, we have analyzed the obtained results.

In the relative gap case, all the proposed methods (obtained through the different combinations of the design factors of the experiments) were found to generate solutions whose quality is independent of a limited number of paths for each OD pair for each departure time interval. Thus, the limitation solution can reduce the computational storage needed in the classical MSA without reducing the good performance of the process.

It has been shown that the solutions are more a function of the selected MSA parameter. It is interesting to note that λ_k^A(which satisfies Equation 6.2.8) and λ_k^B (which does not satisfy Equation 6.2.8) have achieved similar results for convergence speed. However, using λ_k^A produced stable solutions while λ_k^B produced unstable solutions. Moreover, as expected theoretically, the use of the MSA parameter λ_k^C had no good behaviour, which was common in all the proposed experiments.

With respect to the use of the diversion factor in the flow assignment method (flow assignment 1), and comparing it with approaches in the literature (flow assignment 2), both methods had similar convergence in the final iteration. When we combined these two options with the more stable MSA parameter $\left(\lambda_k^A\right)$ (with or without limitations in the number of available paths), the proposed method ($Y1A$ or $N1A$) achieved the accepted DUE gap (0.05, Tong and Wong (2000)) some iterations before the other evaluated option ($Y2A$ or $N2A$).

Regarding the refined relative gap, all the proposed methods were found to perform better by limiting the maximum available paths for each OD pair for each interval.

Another observation is that, as we expected, the MSA parameter λ_k^C did not work well. It presented fluctuations in the solutions of some of the departure time intervals. This occurred with or without the maximum number of path limitations. MSA parameters λ_k^A and λ_k^B have been more stable, especially the λ_k^A option, which provided good and stable results.

Moreover, when the MSA parameter λ_k^A was used, incorporating the diversion factor in the flow reassignment process (flow assignment 1) improved the results obtained with the other option (flow assignment 2). The first proposal achieved good refined relative gaps (less than 0.055) in about half as iterations as the other tested flow assignment method. For the other combinations, the results were very similar for both proposals.

The known bad convergence of the latter departure time intervals is observed in the solutions of all the proposed experiments. The first flow assignment method mitigated this effect by using the diversion factor which enhances the flow reassignment among the alternative paths.

It is important to note here that the evaluation of how the different methods perform differs, depending on whether we consider relative gap or the refined proposal as a reference measurement. In the DTA, the best convergence criteria seems to be the refined relative gap, because the goal of a DTA model based on DUE behaviour is to achieve equilibrium for all the departure time intervals. If we consider only the global relative gap, it appears to be a good solution; but only if we disregard the fact that it is bad for the latter departure time intervals. This is because it can be compensated by very good results for the initial departure time intervals.

The main conclusion is that the method corresponding to experiment $Y1A$ produces the best results for the studied network. Therefore, we propose this method be included in the developed DTA model. This flow assignment method considers: the proposed diversion factor in the reassignment among alternative paths, a limited number of available paths for each OD pair for each departure time interval, and an iteration-dependent MSA parameter $\lambda_k^A = \frac{1}{k+1}$.

Chapter 7

DTA Computational Experiences

7.1 Introduction

7.1.1 Previously Obtained Results

The main components of the proposed DTA model, the DNL and the flow reassignment have been treated independently in Chapters 5 and 6. Thus, their computational experiments have been conducted in the corresponding sections.

In Section 5.5 intensive computational experiments were conducted in order to test the developed mesoscopic simulation model. The results obtained in the first experiment experimentally demonstrate that the proposed mesoscopic model is able to reproduce the fundamental diagram. We have graphically shown this by comparing the obtained flow-density link simulation results and the Underwood macroscopic theoretical relationship. The second experiment complemented the fundamental diagram test that demonstrated the proposed model's basic traffic performance on links. In this case, the obtained results show that our mesoscopic model respects the propagation of congestion, ensuring the temporal and spatial location of congestion at the link level.

In order to experimentally investigate the performance of the developed simulation model, the third set of experiments presented in Section 5.5 tested our model against a microscopic simulator using a real urban networks as a test scenarios. We quantitavely analyzed density, vehicles per link and travel time results. We have studied the errors between measures of both models using RMSE and NRMSE goodness-of-fit measures. The residual analysis showed the correctness of the proposed model for the first scenario: a freeway network. In the second case, the urban network, we also visually compared the network link densities after running both models. The results at the end of each 15-min simulation interval were very similar. However, we note that our model overestimates the density at roundabouts, causing discrepancies in some adjacent links when compared with the results obtained through the microsimulator.

In summary, the obtained results demonstrate the ability of the developed mesoscopic simulation model to reproduce multilane multiclass traffic behaviour for urban networks.

In Chapter 6 we have presented our modification of the Method of Successive Averages (MSA). In Section 6.4 a computational experiment was performed in order to demonstrate the good performance of the proposed flow reassignment process. We used an external DNL model to validate this part independently of our mesoscopic simulation model. An experimental design was conducted by combining the following design factors: flow reassignment option, MSA parameters and limitations on the number of paths for each OD pair for each interval.

The main conclusion of Section 6.4 is that the proposed method ($Y1A$) produces better results for the studied network. Therefore, we have proposed this method to be included in the developed DTA model. This flow assignment method considers: the proposed diversion factor in the reassignment among the alternative paths, a limited number of available paths for each OD pair for each departure time interval, and an iteration-dependent MSA parameter $\lambda_k = \frac{1}{k+1}$.

7.1.2 Summary of the Chapter

In view of the above, we only need to validate whether it is correct to embed both the main components (DNL and flow reassignment) into the DTA method proposed in Chapter 4.

For that purpose, in this chapter the developed DTA model is applied to a real urban network. The aim of this exercise is to validate the correctness of the proposed DTA model. So, to do this, we compare the results obtained through

	15-min Demand Simulation Intervals			
Vehicle Class	Interval 1	Interval 2	Interval 3	Interval 4
Light	2179	3267	3541	1906
Heavy	244	367	398	214
Total	2423	3634	3939	2120

Table 7.1: Demand simulation distributed over the four 15-min simulation intervals.

our proposed DUE with the results obtained with a one-shot simulation. In addition, we analyze the importance of the initial number of paths in the beginning of the global DTA procedure and the importance of the MSA parameter in achieving DUE.

First, we briefly present the real medium-sized test network, which was also used in the computational experiments of the DNL model. Then, we proceed to outline the performed experiments. And finally, we analyze and discuss the obtained results, using different gap measures and other significant graphics as references.

7.2 Test Network

In order to test the behaviour of the developed DTA model in a network, we applied it to the same medium-sized network used previously (Section 5.5.3.4): the Amara Berri district in the city of San Sebastian (Spain).

The main input of the DTA model, besides the network itself, is the time-dependent demand. In this case, the performed experiments use a synthetic demand that assigns to the Amara network the total flow of 12116 trips. Four 15-min matrices provide the origin-destination demand data for 13 zones, resulting in 80 OD pairs. The total number of trips in the matrices is distributed in the manner shown by Table 7.1. Two vehicle classes are considered: 90% of the demand corresponds to light vehicles with an effective length of 5 meters, while the remaining 10% of the demand corresponds to a heavy vehicle class with an effective length of 9 meters.

7.3 Influence of the Number of Initial Paths

At the initialization step, the proposed DTA process requires a set of shortest paths from each origin to each destination. These path sets are determined through a static shortest path algorithm with uses link costs in free flow traffic conditions. As we have explained in Section 4.3, the decision about the number of paths for each OD pair is a key point in achieving convergence. A bad decision may result in false congestion at the beginning of the process, making DUE convergence more difficult.

As we have previously proposed, it is best to test different options during the network calibration process. Depending on the network characteristics, we must select the most convenient number of paths. In this experiment, we consider this parameter as a design factor in order to test its significance.

Because the proposed test network is not a large scenario, the OD pairs have only a few viable routes. In addition, due to the topology of the network, only relevant routes are generated. Thus, the experimental design takes into account one, two or three paths for each OD pair for each interval at the beginning of the DTA process. Moreover, for similar reasons, the quantity of paths is not bounded in these experiments.

It is important to note here that the developed DTA method can remove paths during the process. If some path flow becomes close to zero at a certain iteration of the global process, then this path is removed from the corresponding path set. So, in the next DTA iteration the total demand will be distributed among the remaining paths of the set.

7.3.1 Experiment Design

Table 7.2 summarizes the proposed experiments and details the corresponding design factor combinations.

In addition, with respect to the attributes of different vehicle classes, our DTA model distinguishes classes taking into account only two vehicle attributes: the effective length and the reaction time. As we say before, we consider two vehicle classes: light and heavy, with effective lengths of 5 and 9 meters, respectively. With respect to the reaction times, we consider: light vehicle class = 0.75 seconds and heavy vehicle class = 1.5 seconds. Moreover, a calibration procedure is conducted for the mesoscopic simulation model. This consists of adjusting

EXPERIMENTS	
Experiment	**No. Initial Paths**
ADUE1	1
ADUE2	2
ADUE3	3

Table 7.2: Set of the proposed experiments.

the following parameters: lane change penalty (t_{LCH}) and cross node time penalty ($t_{crossNode}$). The final adjusted values for these penalty parameters are: $t_{LCH} = 0.2s$, $t_{crossNode} = 0.6s$

After this calibration, we execute our DTA for a one-hour long demand. In the following, all the obtained results are summarized and discussed through different convergence measures.

7.3.2 Computational Results

In this section we present the results of the proposed experiments executed over the presented test network of Amara. First, we present the relative gap results in order to have a global idea of the performance of the processes. Then, using the relative gap proposed by Mahut, we try to refine the conclusions by taking into account the different departure time intervals in the analysis of the obtained results. Finally, we complement the convergence measure results by analyzing the total number of paths used during the DTA process and studying the specific path evolutions of certain OD pairs throughout the global procedure.

Relative Gap
The proposed DTA experiments are run for 30 iterations. The 60-min loading interval is divided into four 15-min time intervals for the proposed DTA model. After each iteration, the relative gap proposed by Janson in 1991 (Section 4.3.6) is calculated. Table 7.3 presents the relative gap for each experiment executed after each iteration. Figure 7.3.1 shows the results for the all experiments together.

All these experiments combine the MSA parameter $\lambda_k = \frac{1}{k+1}$ with the options for thenumber of initial paths (1, 2 or 3), as we proposed. We can observe that the developed DTA process obtains good results. The results are similar for all

Experiment							
It	**ADUE1**	**ADUE2**	**ADUE3**	**It**	**ADUE1**	**ADUE2**	**ADUE3**
1	1,32033	1,443510	1,98347	**16**	0,024884	0,057661	0,0440309
2	0,281416	1,02389	1,52189	**17**	0,0192511	0,0496009	0,0377936
3	0,123935	1,10864	1,54018	**18**	0,017192	0,0475279	0,0334088
4	0,0804506	0,438925	0,456263	**19**	0,0640334	0,082236	0,0379974
5	0,0543731	0,836755	0,282174	**20**	0,0207849	0,045409	0,0300988
6	0,0734269	2,38421	0,189246	**21**	0,023140	0,0379952	0,0309471
7	0,0409453	0,394492	0,132304	**22**	0,0211947	0,0325866	0,0341701
8	0,0387291	0,252831	0,110916	**23**	0,0163969	0,062268	0,0381676
9	0,0352297	0,137747	0,111853	**24**	0,066151	0,0530731	0,0334261
10	0,0400965	0,130131	0,0826172	**25**	0,0609422	0,0549249	0,0307209
11	0,0375388	0,0828837	0,0820837	**26**	0,0473897	0,0394265	0,0224247
12	0,0279188	0,0872711	0,071147	**27**	0,0285407	0,0372569	0,0250192
13	0,0251143	0,0740704	0,0574055	**28**	0,0428903	0,0505024	0,020432
14	0,0270387	0,0709509	0,0528119	**29**	0,0261308	0,0480049	0,0229247
15	0,0189426	0,0643989	0,0475976	**30**	0,027495	0,0338252	0,0224747

Table 7.3: Relative Gap for each iteration of each executed experiment.

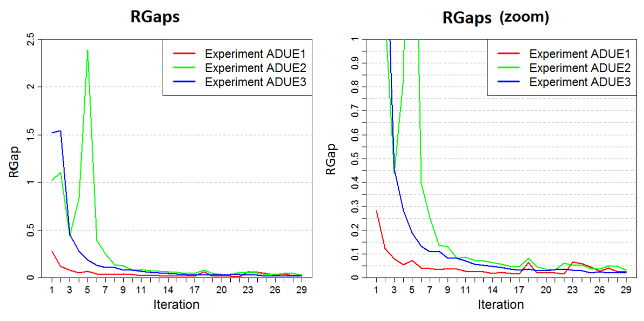


Figure 7.3.1: Relative Gap for each experiment after each DTA iteration.

the cases, but we can note that the results for the experiment that starts with three paths for each OD pair for each interval ($ADUE3$) is more stable than the other two experiments. It converges at the seventh iteration of the process without losing this convergence throughout the rest of the procedure.

Refined Relative Gap

The proposed DTA experiments are run for 30 iterations. The 60-min loading interval is divided into four 15-min time intervals for the proposed DTA algorithm. After each iteration, the refined relative gap proposed by Mahut in 2003 (see Section 4.3.6) is calculated for the vehicles departing from the origin during each of these intervals.

In Figure 7.3.2, graphics $ADUE1$, $ADUE2$ and $ADUE3$ show the results for the MSA parameter $\lambda_k = \frac{1}{k+1}$ combined with each of the three options for the number of initial paths (1, 2 or 3). To better analyze the results of the experiments, the zooms of the results are shown in graphics $ADUE1(zoom)$, $ADUE2(zoom)$ and $ADUE3(zoom)$.

With respect to the refined RGap results, we can observe that if the experiment that starts the DTA process with only one path for each OD pair for each interval ($ADUE1$), it seems better than the other experiments at the beginning of the process. In fact, ADUE1 achieves the DUE convergence (less than 0.05) at the 8th iteration. However, it is important to note here that at this iteration the process does not achieve stability. This is because two iterations later we can observe that the RGap grows above 0.05 for some departure time intervals.

In the case of the second experiment ($ADUE2$), we can observe the same absence of stability. In this case, the method achieves DUE convergence at iteration 18, but it is lost in the next iteration.

Similar to the case of the standard relative gap measure, the option $ADUE3$ improves stability when compared to the results obtained by the other two experiments. In this case, we need more iterations to achieve equilibrium (20 iterations). But in the following ten iterations the refined RGap measure results remain under 0.05.

Path Analysis

The proposed experiments are run for 30 iterations. The 60-min loading interval is divided into four 15-min time intervals for the proposed DTA model. Figures 7.3.3, 7.3.4 and 7.3.5 show the total number of paths used at each

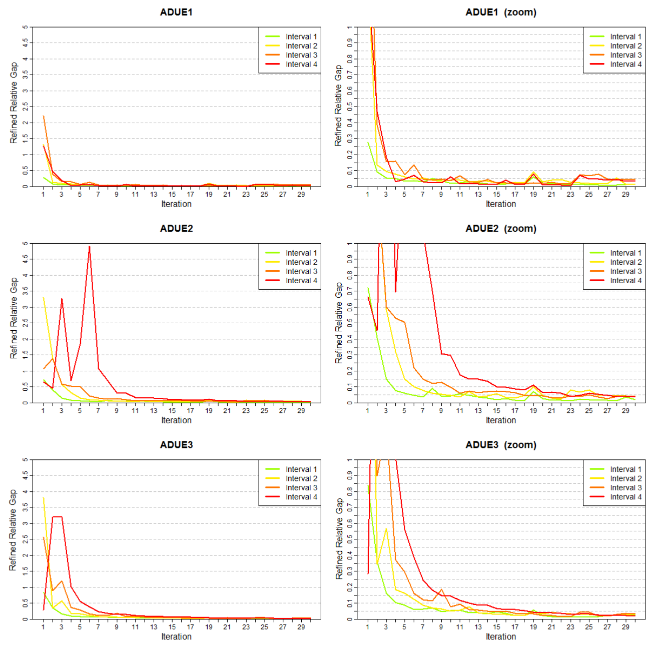


Figure 7.3.2: Refined Relative Gap experiments with MSA parameter $\lambda_k = \frac{1}{k+1}$.

Total no. of paths used at the 30th iteration				
	Interval 1	Interval 2	Interval 3	Interval 4
ADUE1	103	104	116	109
ADUE2	103	109	113	120
ADUE3	107	107	107	111

Table 7.4: Total number of paths used at the end of the 30th iteration.

DTA iteration for each 15-min simulation interval for each of the proposed experiments.

The first observation about the obtained results in the study of the paths is that not all the OD pairs can use two or more paths. When we observe the second and third proposed experiments ($ADUE2$ and $ADUE3$), we note the following. In the first case, we hoped that the number of paths used in the first DTA iteration would be 160 (80 OD pairs x 2 initial paths), but there are only around 140 paths at each simulation interval. In the second case, we hoped that the number of total paths used at the end of the first DTA iteration would be 240 (80 OD pairs x 3 initial paths), but there are around 200 paths at each simulation interval. So, we can conclude that some OD pairs do not have more than one path between their origin and their destination.

Along the DTA process, we can observe different numbers of total paths used for each of the proposed experiments. In the $ADUE1$ experiment the process starts with one path for each OD pair for each interval. We can observe that this total number quickly grows until reaching approximately 100 paths. After this, the DTA process remains stable in respect to the number of paths used. In the second proposed experiment, $ADUE2$, we can see that the total number of paths used starts growing until reaching 180 paths. After this, the DTA process starts to remove paths until reaching stability at 100 paths. Finally, the $ADUE3$ experiment shows that the process needs less than 3 paths for each OD pair for each interval. So, it removes some paths ofthe total set of the first iteration until reaching around 100 paths.

If we pay attention to the last iterations of the DTA process, we can observe that the total number of paths used at each 15-min simulation interval are similar for all the proposed experiments. In particular, Table 7.4 shows the results for the 30th iteration, when all the experiments have achieved DUE or are very close to this equilibrium.

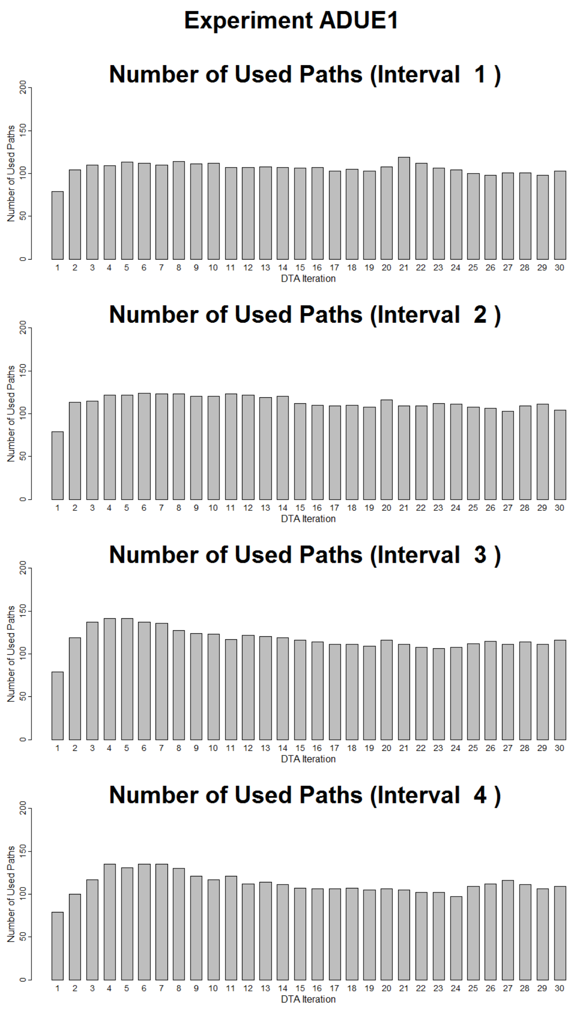


Figure 7.3.3: Total number of paths used for the experiment ADUE1.

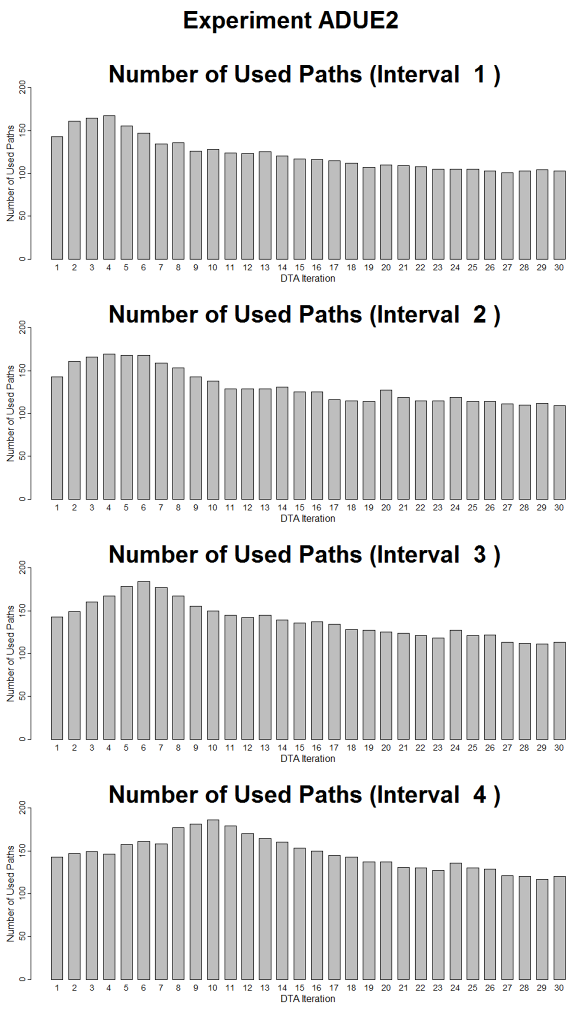


Figure 7.3.4: Total number of paths used for the experiment ADUE2.

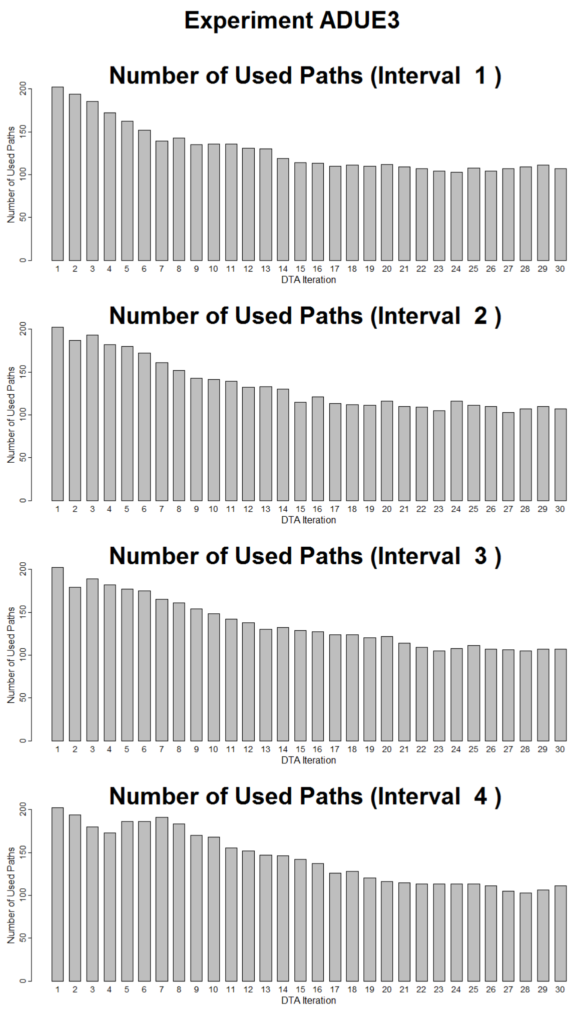


Figure 7.3.5: Total number of paths used for the experiment ADUE1.

Paths Used throughout the DTA process								
Experiment	Interval 1		Interval 2		Interval 3		Interval 4	
	No.	Paths	No.	Paths	No.	Paths	No.	Paths
ADUE1	3	1,2,3	4	1,2,3,5	3	1,2,7	5	1,2,3,4,5
ADUE2	4	1,2,7,8	5	1,2,5,7,10	5	1,2,5,6,7	7	1,2,4,5,7,11,12
ADUE3	5	1,2,5,7,13	7	1,2,3,5,7,12,13	6	1,2,3,7,12,13	6	1,2,7,9,12,13

Table 7.5: Paths used throughout the DTA process (OD Pair 806-783).

It is important to note that $ADUE3$ is the experiment that shows more stable results in terms of relative gaps (standard and refined). However, this is the experiment that starts with the largest excess of paths and the process that uses more paths at the last DTA iteration.

In order to better analyze the obtained results, we have decided to select one particular OD pair and to study the evolution of its paths used throughout the DTA procedure. We chose the OD pair formed at origin 806 and destination 783.

First of all, Table 7.5 summarizes the number of paths used at each 15-min simulation interval throughout each of the proposed experiments. Experiment $ADUE1$ uses a total of six paths throughout the procedure. $ADUE2$ uses a total number of ten paths. And experiment $ADUE3$ uses eight paths during the DTA process. In addition, Table 7.5 shows the identification number of each of the paths used. We can observe that there are paths used in all three experiments, other paths used in two experiments, and other paths that are used exclusively in one experiment. Finally, the total number of different paths used in all the experiments is 13.

Below, Figure 7.3.6 (zoom in Figure 7.3.7) describes in detail the evolution of the paths from origin 806 to destination 783 during the DTA process in experiment $ADUE1$. The process starts by using only one path ($Path\,1$). It is interesting to note that adding paths in the next iterations depends on the network situation, which is not the same at every interval. For instance, in the second iteration, the process adds a new path to the set of paths. In the case of Intervals 1 and 3, the added path is $Path\,2$, while in the case of Interval 2 it is $Path\,5$. And in Interval 4, it is $Path\,4$.

Finally, we can observe that some paths are added throughout the procedure. But in the end, DUE is achieved using 1, 2 or 3 paths.

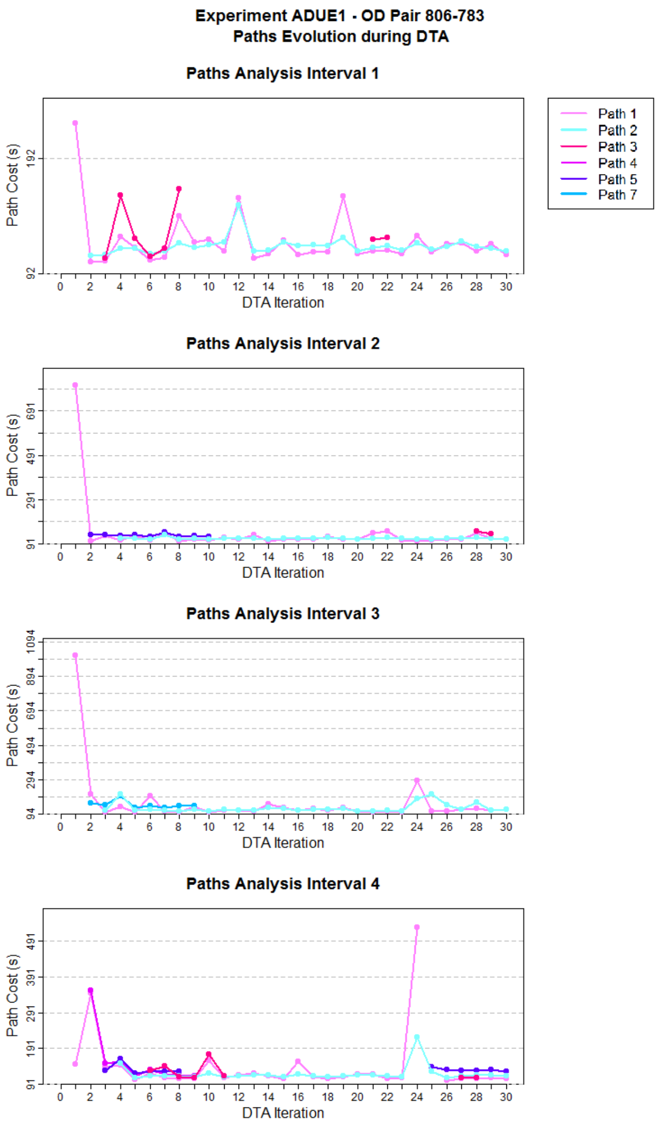


Figure 7.3.6: Example of OD pair path evolution during the DTA process with experiment ADUE1.

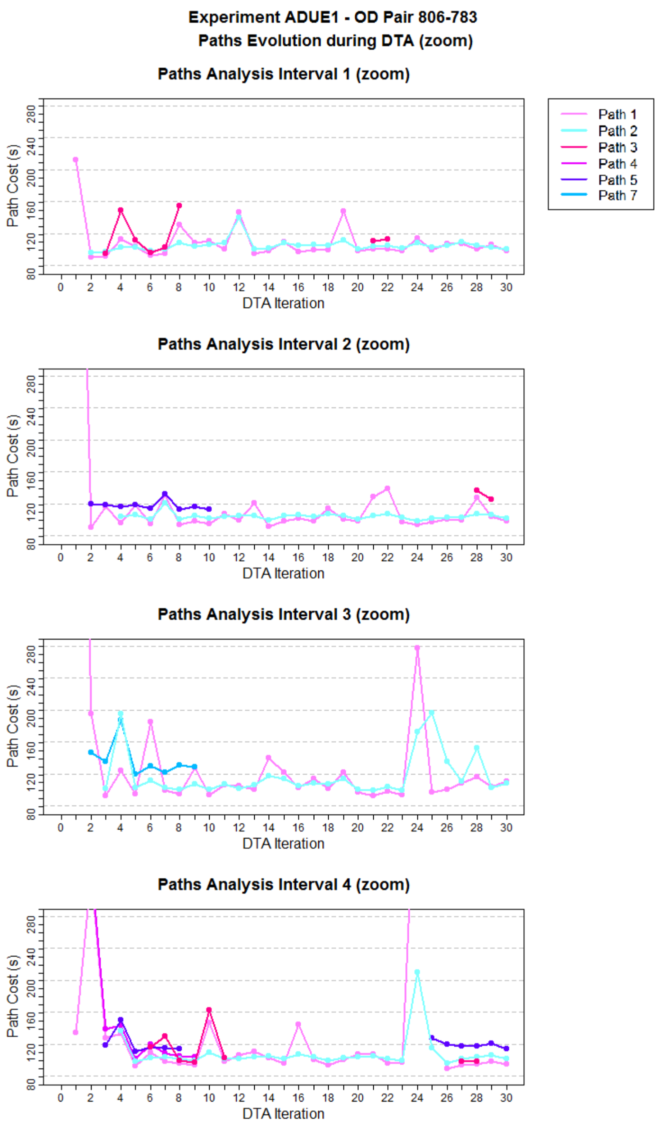


Figure 7.3.7: Example of OD pair path evolution during the DTA process with experiment ADUE1 (zoom).

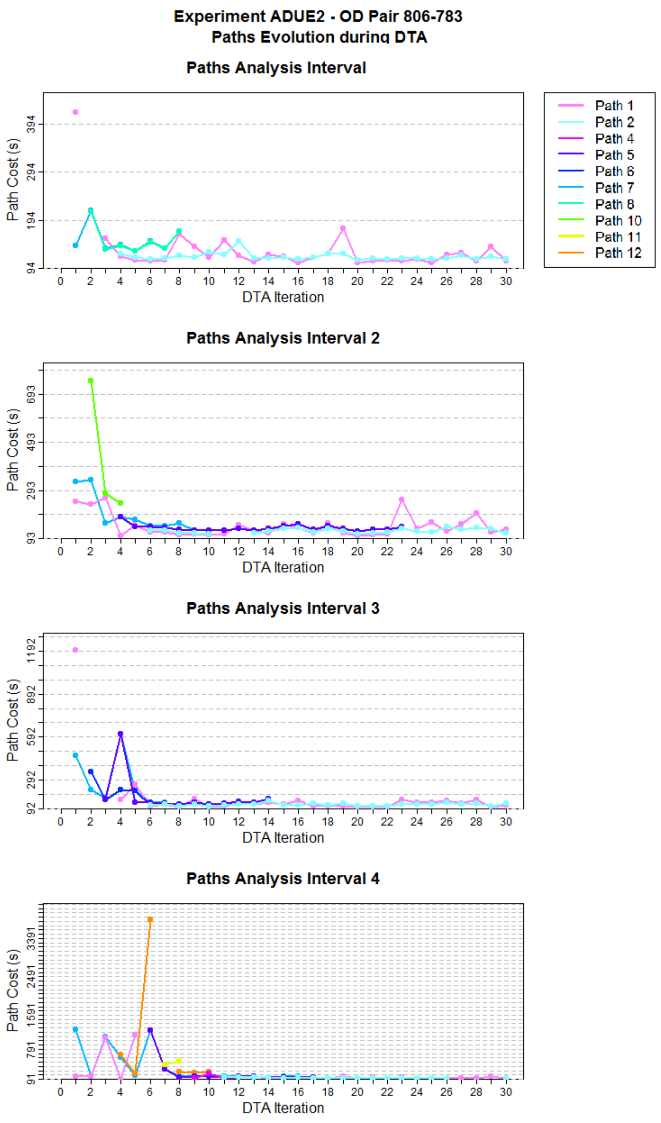


Figure 7.3.8: Example of OD pair path evolution during the DTA process with experiment ADUE2.

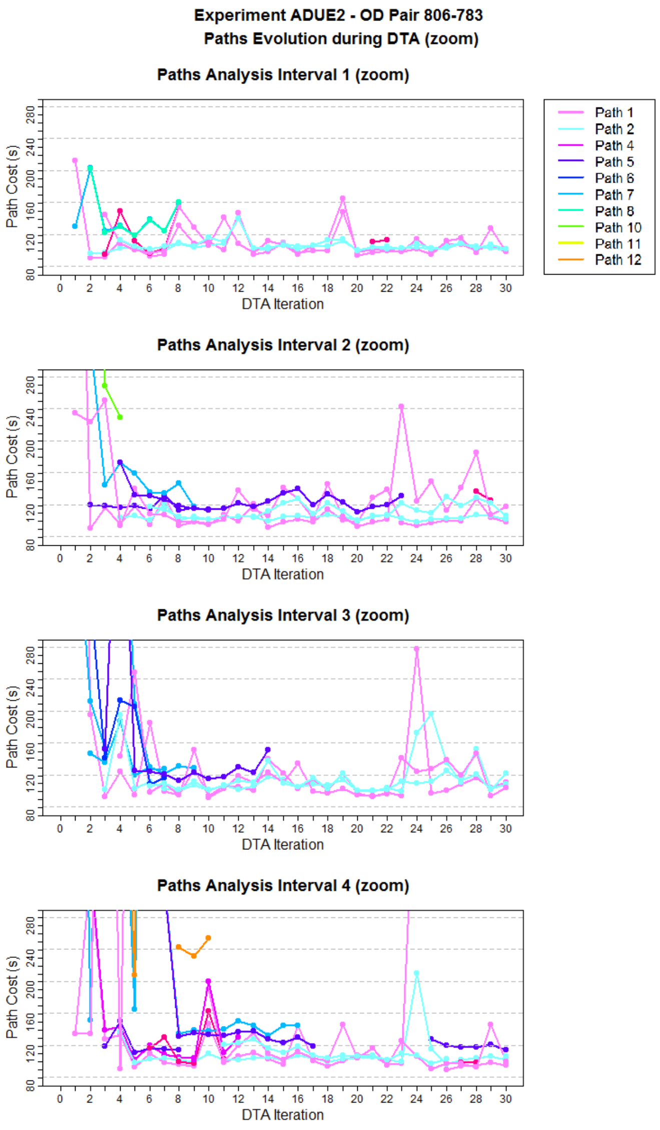


Figure 7.3.9: Example of OD pair path evolution during the DTA process with experiment ADUE2 (zoom).

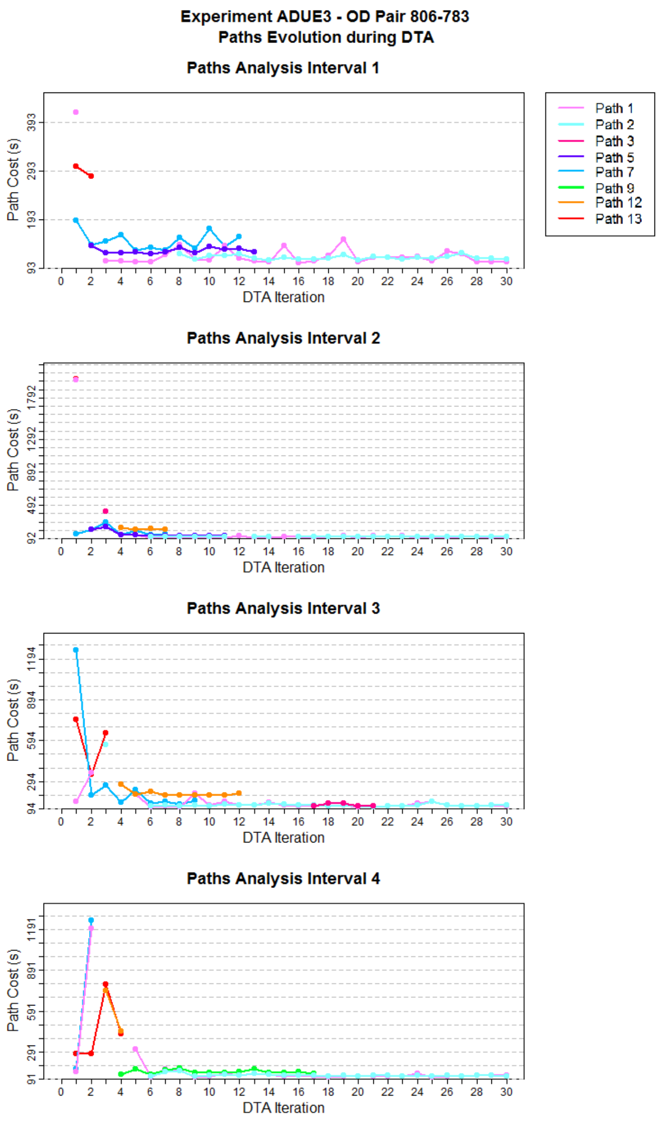


Figure 7.3.10: Example of OD pair path evolution during the DTA process with experiment ADUE3.

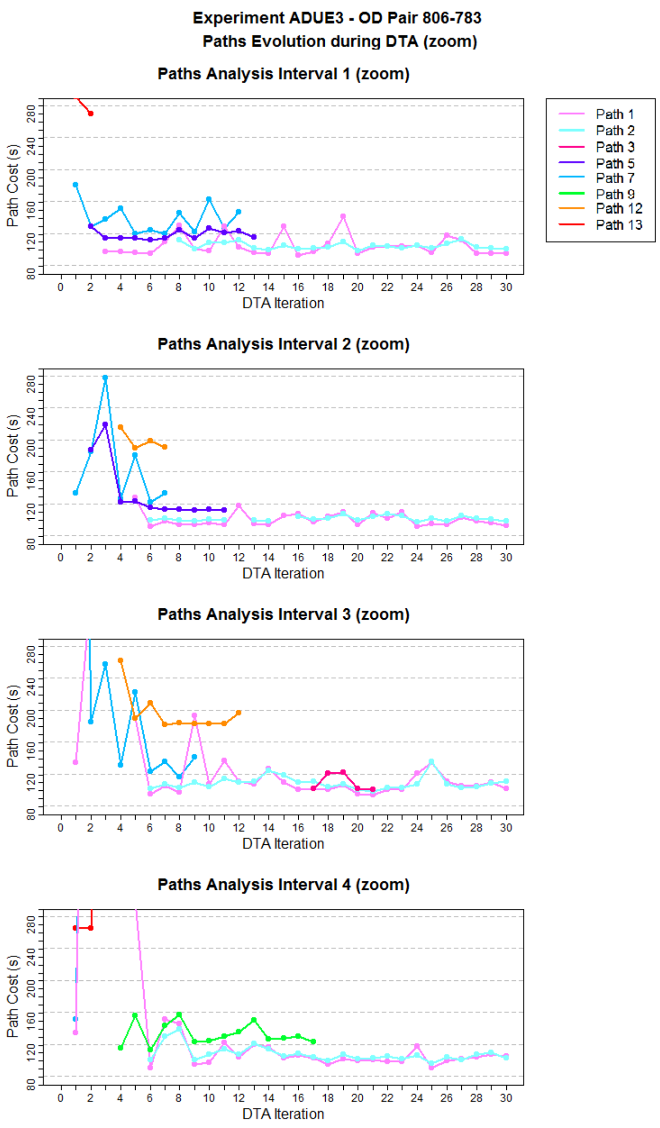


Figure 7.3.11: Example of OD pair path evolution during the DTA process with experiment ADUE3 (zoom).

Figure 7.3.8 (zoom in Figure 7.3.9) describes in detail the evolution of the paths from origin 806 to destination 783 during the DTA process in experiment $ADUE2$. The process starts by using two paths ($Path\,1$and $Path\,7$). As in the previously mentioned experiment, the addition of paths depends on the network situation, which is not the same at every interval. Moreover, in this experiment we need to add more paths throughout the procedure. The number of paths rises to 6 paths at Interval 4. However, at the end of the DTA process, the number of paths used is two for all the intervals.

Figure 7.3.10 (zoom in Figure 7.3.11) describes in detail the evolution of the paths from origin 806 to destination 783 during the DTA process in experiment $ADUE3$. The process starts by using three paths ($Path\,1$, $Path\,7$ and $Path\,3$). As in the previously mentioned experiments, the addition of paths depends on the network situation, which is not the same at every interval. Moreover, in this experiment we need to add more paths throughout the DTA procedure. In this experiment we can see that the process converges to only two paths ($Path\,1$ and $Path\,2$) and that it happens quickly in comparison to the other experiments.

The presented graphics reflect once again the results obtained for relative gap (standard and refined) measures. It should be remembered that the DUE solution is expected to use all the paths with similar travel costs. In these computational experiments, Experiment ADUE1 uses only two paths with similar costs faster than the other two experiments. However, we detect that the process is a bit unstable, particularly at Interval 4. On the other hand, Experiment ADUE2 presents the worst results of the three proposed experiments. Finally, we can see that the experiment that starts with three paths for each OD pair for each interval is the best. This presents a stable solution which uses two paths with very similar travel costs from the 20th iteration.

Finally, we can observe that the three proposed experiments use almost the same number of paths in achieving DUE. Moreover, at the end of the DTA process they use practically the same paths. In this particular case, $Path\,1$ and $Path\,2$ are both included in the three proposed solutions.

7.4 DUE vs One-Shot Simulation

In this section, we want to study the traffic assignment that results from a DTA model based on the DUE approach. The objective is to quantify the improvements in the traffic network if we base the DTA on a specific equilibrium (DUE), instead of doing the traffic assignment through a one-shot simulation.

It should be recalled here that the DUE approach is a time-dependent generalization of the First Wardrop Principle: If, for each OD pair at each instant of time, current travel times experienced by vehicles that have departure in the same time interval are equal and minimal, then the dynamic traffic flow through the network is in a DUE state based on travel times.

In order to attain the objective, the proposed model is run over the presented test network of Amara. In addition, a one-shot simulation is executed and both results are compared qualitatively and quantitatively.

7.4.1 Experiment Design

The performed experiments take into account two different assignment situations. The first one corresponds to the proposed DTA model whose objective is to converge towards DUE. So, in this case the method must iterate as many times as necessary until DUE is achieved. The second option is commonly referred to as the "one-shot" assignment; it is a simulation approach commonly used in some microsimulators.

In order to perform the one-shot assignment in the proposed experiments, we force the method to stop after the first iteration. Thus, the demand assignment will correspond to the first iteration assignment, which is performed differently than those in the remaining DTA iterations. As we have shown in Section 4.3, the flow assignment is inversely proportional to the path cost. For each OD pair od, for each departure time interval t, and for all paths $p \in P_{odt}$, the flow is assigned by following the Equation 7.4.1.

$$f^k_{odpt} = \frac{1/c_{odpt}}{\sum\limits_{p \in P_{odt}} (1/c_{odpt})} \cdot q_{odt} \tag{7.4.1}$$

where:

f^k_{odpt} is the flow assigned at iteration k to the path p from origin o to the destination d departing at time interval t.

c_{odpt} is the cost of the path p from origin o to the destination d departing at time interval t.

q_{odt} is the number of trips of the corresponding time dependent OD matrix.

In the case of the DUE assignment, the DTA process is allowed to iterate until reaching a pre-established maximum number of iterations. Perhaps equilibrium is achieved before, when the corresponding refined relative gap measure is acceptable (around 5%), but we prefer to extend the procedure in order to verify the stable behaviour of the developed DTA model. In these experiments, we consider the maximum number of DTA iterations to be equal to 30. In addition, we consider the initial number of paths for each OD pair for each interval to be equal to three (experiment $ADUE3$).

7.4.2 Computational Results

In this section, we want to illustrate the importance of the improvements achieved with DUE. In order to attain this objective, we execute the proposed experiments over the presented real network of Amara. First of all, Figures 7.4.1 and 7.4.2 show the link density at the end of the first and the fourth 15-min intervals of the simulation, respectively. For an easy visual comparison, we show the results obtained through the proposed DTA model next to the results obtained through one-shot, all of them at the same simulation interval. We consider that the other two 15-min simulation intervals do not provide additional information for the proposed objective of the experiment.

We can observe that the results obtained for both experiments (One-shot Simulation vs DUE) are totally different. The one-shot simulation congests the network in the first 15-min simulation interval. In addition, at the end of the simulation, we can consider that the network situation is critical. On the other hand, at the right side of the figures, we observe the simulation results obtained by using DUE paths and their corresponding flows. It is obvious that the obtained results show practically a free-flow situation. So, the assignment result of our DTA model allows us to load into the network all the demand (12.116 trips) without generating congestion.

Secondly, we want to show other results that confirm the previously detected improvements in the network. For each vehicle, we plot a dot whose x-coordinate is the departure time of the vehicle (the time when the vehicle enters the network) and whose y-coordinate is the total time that this vehicle spends in the network. Figure 7.4.3 shows the results for the two proposed experiments: one-shot simulation vs DUE. In addition, Table 7.6 shows a quantitative summary of the obtained results.

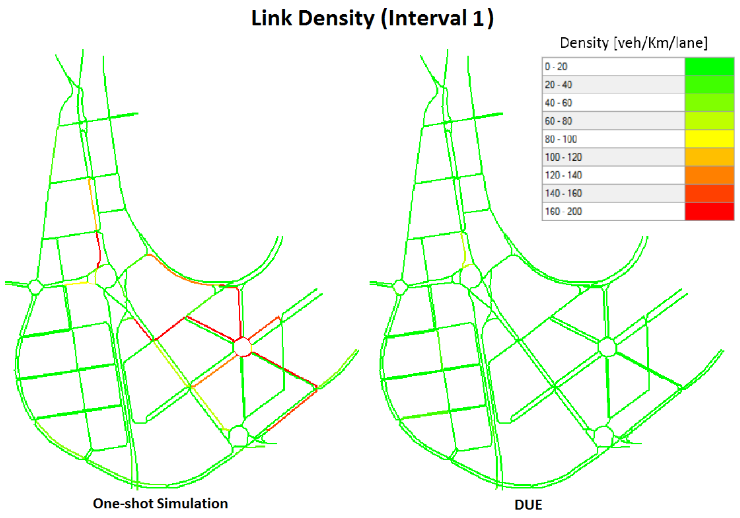


Figure 7.4.1: DUE vs One-Shot Simulations: Average link densities at the end of 15-min simulation interval 1.

Experiment	Consumed Time in Network (seconds for vehicle)				Accumulated Consumed Time (seconds)
	Min	Median	Mean	Max	
One-Shot	19,08	158,70	224,90	1.181,00	2.482.045
DUE	19,08	92,59	91,06	234,90	1.031.604

Table 7.6: Summary of the obtained results for one-shot vs DUE simulations.

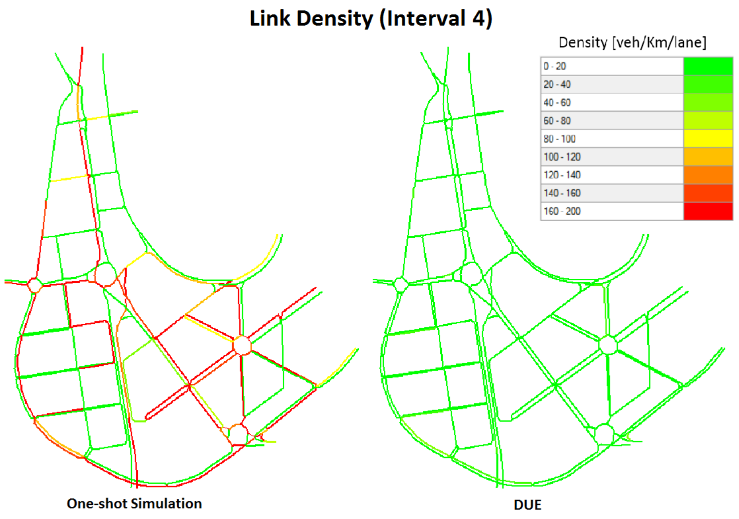


Figure 7.4.2: DUE vs One-Shot Simulations: Average link densities at the end of 15-min simulation interval 4.

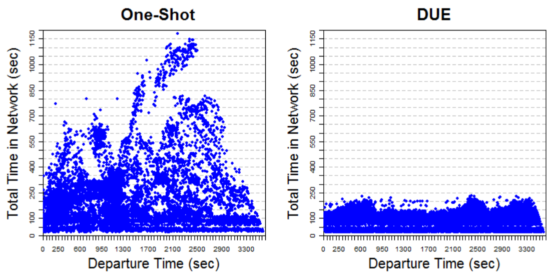


Figure 7.4.3: Total Time in Network One-Shot Simulation vs DUE.

In the case of one-shot simulation, we can observe that the total time used for each vehicle varies from 19.08 seconds to 1,181 seconds. The mean is 224.9 seconds, which is very high if we consider the lengths of the links in the Amara scenario. However, if we observe the total time used for each vehicle in the case of the DUE experiment, we see that it varies from 19.08 seconds to 234.90 seconds. And its mean is 91.06 seconds. Thus, we can conclude that if we use the assignment result of our DTA model (under a DUE approach), the achieved average improvement for each vehicle of the network is of 59.51% compared to the time spent in the proposed one-shot simulation.

7.5 MSA Parameter Impact

The objective of this computational experiment is to prove the importance of the MSA parameter in the DTA process, particularly in achieving convergence. We suspect that when the traffic demand is low, whatever MSA parameter is used makes no difference. The DTA process is able to find the DUE traffic flow assignment through the MSA procedure, no matter what the MSA parameter.

7.5.1 Experiment Design

The performed experiments consider the following three different MSA parameters (λ_k):

- $\lambda_k^A = \frac{1}{k+1}$
- $\lambda_k^B = \frac{1}{2}$
- $\lambda_k^C = \frac{k}{k+1}$

The first one, λ_k^A, is a standard option used by most methods in the literature. It is an MSA parameter that depends on the current iteration of the method and that satisfies Equations 6.2.8. These two conditions ensure the theoretical MSA convergence. With the second proposal, λ_k^B, we want to test what happens if the parameter does not depend of the current iteration, i.e., it is a fixed diversion parameter within the new path and the rest of the paths belonging to the path set. So, we propose a surprising parameter which does not satisfy Equation 6.2.8 and that diverts a half of the total demand to the new path. It

EXPERIMENTS		
Experiment	**No. Initial Paths**	λ_k
ADUE3	3	A
BDUE3	3	B
ADUE3	3	C

Table 7.7: Set of the proposed experiments.

also distributes the other half of the total demand among all the other paths of the set. Finally, we want to test the importance of using an MSA parameter that satisfies Equations 6.2.8. With this aim, we propose λ_k^C, which does not satisfy these two conditions. We also want to test the "bad"convergence of the MSA in this case.

Taking into account the results obtained in the previous Section 7.3, we have decided to use three paths for each OD pair for each interval at the beginning of the DTA process. In this case, we run three different experiments, one of them with one of the presented proposals for the MSA parameter (the same as in Section 6.4). Table 7.7 summarizes the proposed experiments.

In order to accomplish the objective of the computational experiment, we execute each proposed experiment for two different demands. The first one is the demand used in the previous computational experiments, which is detailed in Section 7.2. In this case, we assign the total flow of 12,116 trips through four 15-min matrices. The other used demand is a reduction of the first one. We consider 40% of the previous demand. So, in this case we assign the total flow of 4,847 trips that follow the same distribution as the first assigned demand.

After the calibration, we execute our DTA for a one-hour long demand. In the following, all the obtained results are summarized and discussed through different convergence measures.

7.5.2 Computational Results

In this section we present the results of the proposed experiments executed over the presented test network of Amara. First, we present the relative gap results in order to have a global idea of the performance of the processes. Then, using the relative gap proposed by Mahut, we try to refine the conclusions by taking

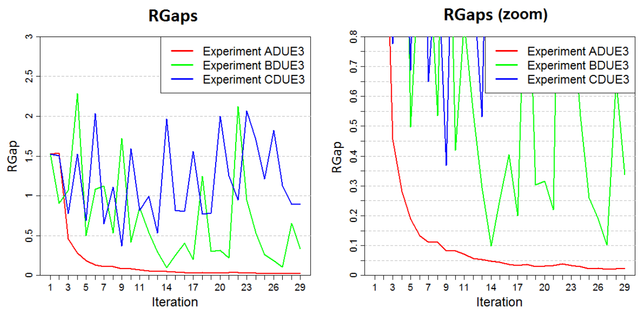


Figure 7.5.1: Relative Gap for each experiment after each DTA iteration.

into account the different departure time intervals in the analysis of the obtained results.

Relative Gap
The proposed DTA experiments are run for 30 iterations. The 60-min loading interval is divided into four 15-min time intervals for the proposed DTA model. After each iteration, the gap proposed by Janson in 1991 (Section 4.3.6) is calculated.

Demand : 12.116 trips Figure 7.5.1 shows the results for the three experiments executed with a demand of 12,116 trips.

Demand : 4.847 trips Figure 7.5.2 shows the results for the three experiments executed with a demand of 4,847 trips.

All these experiments combine an initial number of paths for each OD pair for each interval equal to three, with the proposed options of the MSA parameter. In the case of higher demand, we can observe that the developed DTA process obtains reasonable results only in the case of experiment $ADUE3$. The experiments that use $\lambda_k^B = \frac{1}{2}$ or $\lambda_k^C = \frac{k}{k+1}$ provide the expected bad results, because neither of these parameters satisfies Equations 6.2.8.

In contrast, we observe very different results for the same experiments with low demand. Surprisingly, all the experiments provide very good and stable results.

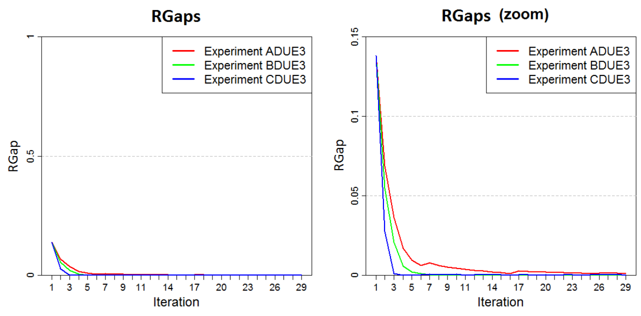


Figure 7.5.2: Relative Gap for each experiment after each Low Demand DTA iteration .

In fact, in the first iteration the relative gaps of the proposed DTA procedure are around 0.15, and they fall rapidly (within a few iterations) to practically zero.

Refined Relative Gap

The proposed DTA experiments are run for 30 iterations. The 60-min loading interval is divided into four 15-min time intervals for the proposed DTA algorithm. After each iteration, the refined relative gap proposed by Mahut in 2003 (see Section 4.3.6) is calculated for the vehicles departing from the origin during each of these intervals.

Demand : 12,116 trips In Figure 7.5.3, graphics $ADUE3$, $BDUE3$ and $CDUE3$ show the results for the three different proposed MSA parameters, combined with a number of initial paths equal to three. To better analyze the results of the experiments, the zooms of the $ADUE3$ results are shown in graphic $ADUE3(zoom)$.

Demand : 4,847 trips In Figure 7.5.2, graphics $ADUE3$, $BDUE3$ and $CDUE3$ show the results for the three different proposed MSA parameters combined with a number of initial paths equal to three.

With respect to the refined RGap results, we can observe the same surprising results as in the case of a standard RGap measure. For the demand of 12,116

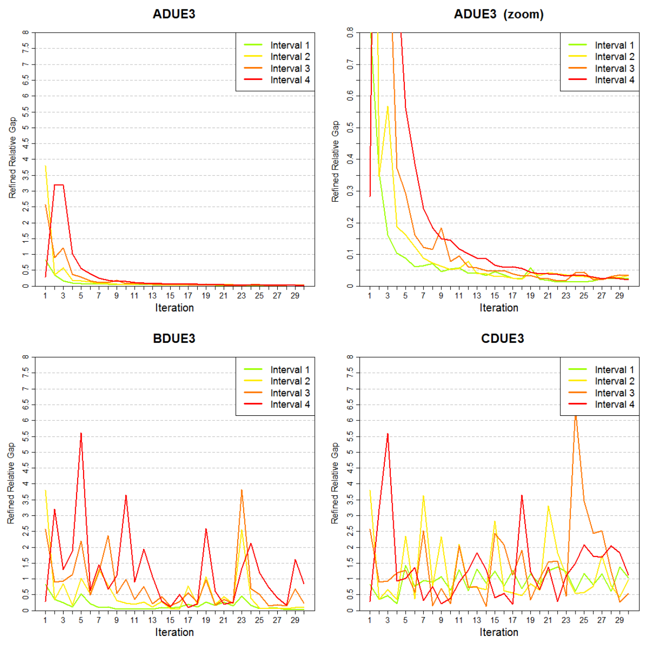


Figure 7.5.3: Refined Relative Gap experiments with initial number of paths equal to three.

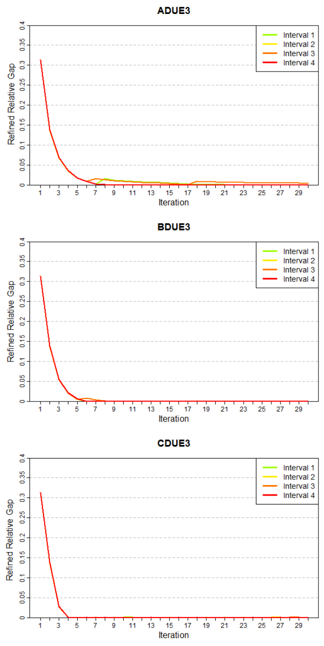


Figure 7.5.4: Refined Relative Gap experiments with initial number of paths equal to three (Low Demand).

trips, only the third proposed experiment, which uses MSA parameter $\lambda_k^A = \frac{1}{k+1}$, achieves good results. The other two experiments, which use $\lambda_k^B = \frac{1}{2}$ and $\lambda_k^C = \frac{k}{k+1}$, respectively, provide the expected bad results. The results for $CDUE3$ are even worse than the constant option assumed in experiment $BDUE3$.

In contrast, for the low demand executed experiments, we observe very different results. The three experiments start with a refined relative gap of around 0.3, but they fall very fast to practically zero. The best experiment is CDUE3, which uses $\lambda_k^C = \frac{k}{k+1}$ and which does not satisfy the convergence Equation 6.2.8.

As we suspected, the MSA flow reassignment converges independently of the proposed MSA parameter in low-demand scenarios. This is true even for options with theoretically bad behaviour.

7.6 Summary and Conclusions

In this chapter we have performed a computational experiments that test the feasibility of the proposed DTA model on the real network of Amara in Spain.

The first experiment has investigated the influence of deciding the number of paths for each OD pair for each interval, specifically in the way the proposed DTA model achieves DUE convergence . At the initialization step of the developed model, it requires a set of shortest paths for each origin to each destination for each departure time interval. These sets of paths are determined using a k-static shortest path algorithm (Section 4.3). Thus, before beginning the DTA process, we must decide the initial number of paths for each OD (k).

During the network calibration process, and depending on the network characteristics, we select the most convenient number of paths. In certain cases, a bad decision may result in false congestion at the beginning of the process, rendering the DUE convergence more difficult. In the first proposed experiment, we considered this parameter as a design factor in order to test its real significance in the global convergence process. In this particular case, as we have justified in the corresponding section, we took into account one, two or three paths for each OD pair for each interval at the start of the DTA process.

In this experiment we the graphically compared Relative Gap and refined Relative Gap measures obtained for each of the proposed experiments. In this case, we observed that the experiment starting with only one path ($ADUE1$)

seems better than the other two experiments. It achieves convergence at the 8th iteration. However, this experiment is not stable because it presents convergence oscillations during approximately 30 DTA iterations. On the other hand, the third proposed experiment ($ADUE3$) improves the stability in the other obtained results. In this case, it needs 20 iterations to achieve equilibrium; but in the next ten iterations the refined RGap measure remains under 0.05. The observed behaviour for the RGap measure is very close to the previously mentioned refined RGap.

In addition, we have analyzed in detail the paths used throughout the DTA process in this first computational experiment. First, we studied the evolution of the total number of paths used during the 30 iterations of the proposed DTA experiments. Throughout the procedure, we observed a different number of paths for each of the proposed experiments, depending on the initial number of paths. However, we paid attention to the last iterations of the DTA (when all the experiments achieved DUE or were very close to this equilibrium), and we observed that the total numbers of paths used at each 15-min simulation interval were similar for all the proposed experiments.

Secondly, we have decided to select one particular OD pair and to study its path evolution throughout the iterative DTA process. We have observed that the results obtained for RGap and refined RGap measures are reflected again in this particular case. The main conclusion is that the three proposed experiments use almost the same number of paths to achieve DUE, independently of the initial number. In addition, the three experiments used practically the same paths ($Path\,1$ and $Path\,2$ are both included in the three proposed solutions).

In summary, we have concluded through this first computational experiment that the experiment starting with the largest excess of paths provides more stable results in terms of RGap and refined RGap convergence measures. This is the $ADUE3$ experiment, which uses three paths at the beginning of the DTA process. This is possible because the developed DTA method can remove paths during the process. If some path flow becomes close to zero during a certain iteration of the global process, then this path is removed from the corresponding path set. So, in the next DTA iteration the total demand will be distributed among the remaining paths of the set.

The aim of the second proposed computational experiment was to verify the expected behaviour of the developed DTA model. Our proposed model is based on the DUE approach; so we hoped that the results obtained through a simulation based on our DTA approach would be better than other results without a DUE assignment. In order to attain the objective of this computational experiment, the proposed model was run over the test network of Amara, and the results were compared with those obtained through a previously proposed one-shot simulation.

As expected, the results obtained from both experiments were totally different. The one-shot simulation congests the network rapidly, finalizing the simulation with a critical network situation. In contrast, the assignment result of our DTA model allows loading all the proposed demand (12,116 trips) into the network without generating congestion. After qualitatively and quantitatively comparing the results, we conclude that if we use the assignment result of our DTA model, the average consumed time in the network drops from 224.9 seconds for each vehicle to 91.06 seconds.

In the last computational experiment, the objective was to prove the importance of the MSA parameter in the DTA process, particularly in its achievement of DUE convergence using the MSA flow reassignment. Before this computational experiment, we suspected that the MSA parameter was not relevant to the procedure when the considered demand is low. In order to accomplish the objective, we considered three different MSA parameters (the same as in the computational experiments of Section 6.4). Moreover, we executed each proposed experiment for two different demands: the demand used in computational experiment which had 12,116 trips; and a low demand with 4,847 trips, which followed the same distribution as the first assigned demand.

In this experiment we graphically compared RGap and refined RGap convergence measures for all the proposed experiments. Both RGap and refined RGap results show the same expected results. For the demand of 12,116 trips, only the third proposed experiment, which used MSA parameter $\lambda_k^A = \frac{1}{k+1}$, provided good results. The other two experiments that uses $\lambda_k^B = \frac{1}{2}$ and $\lambda_k^C = \frac{k}{k+1}$, respectively, provided the expected bad results. In contrast, for the low demand experiments, the three MSA options start with very good results and the gaps fall very fast to practically zero. The best experiment is CDUE3, which uses $\lambda_k^C = \frac{k}{k+1}$ and which does not satisfy the convergence Equation 6.2.8.

As we suspected, the MSA flow reassignment converges independently of the proposed MSA parameter in the case of low demand scenarios. So, it is important to note here that this conclusion must be taken into account in the previously presented results of Chapter 6.

Finally, we conclude that the computational experiments carried out in this chapter have demonstrated the good behaviour of our DTA model. We have shown that the initial number of paths for each OD pair for each interval at the beginning of the process has a key role in the stability of the convergence. All the proposed options achieve DUE convergence using almost the same number of paths, but not with the same level of stability. Starting with an excess of paths and removing them when it becomes necessary appears to be the most stable option. In addition, we were able to test the importance of the MSA parameter in the reassignment flow process on its way to convergence. In the computational experiments of Chapter 6, we observed surprising results with the MSA parameters that, intuitively, seemed they would provide bad convergence. Now, we have shown that the demand assigned to the network must be taken into account: very low demand scenarios will accept any MSA parameter.

Chapter 8

Conclusions

In this final chapter, we look back on the main achievements of this dissertation. The summary of the main research objectives and the approach used to achieve these objectives are presented. After this, we discuss the main research findings. And, finally, some recommendation for further research are formulated.

8.1 Summary

This dissertation deals with a problem that has been extensively studied for decades: the dynamic traffic assignment (DTA) problem. This area of research has particularly accelerated in the last twenty years, since the emergence of Intelligent Transport Systems (ITS). In an operational context, the objective of the DTA model is to represent traffic evolution in the roadway network when traffic conditions change. These models describe the demand assignment on the different paths connecting every OD pair corresponding to an equilibrium traffic state. In particular, we studied the problem when DTA is based on dynamic user equilibrium (DUE), where no users can decrease their travel time by changing paths.

Thus, the main objective of the research presented in this thesis was to develop a DTA model based on the dynamic extension of the Wardrop Principle (DUE). The main motivation for developing this model was the need to develop a decision support tool for assisting operators (sometimes our own research group) in making strategic and operational decisions, particularly in an urban context. In addition, for some time now our research group has been addressing other traffic problems, and while resolving them we realized the importance of having our own DTA model. In this way, we could easily integrate the DTA results into the resolution approaches of other traffic problems related to strategic and operational transport planning.

For this purpose, we solved a variational inequalities formulation by employing an iterative solution algorithm based on a preventive approach, which is a modification of the well-known Method of Successive Averages. In particular, this thesis develops a DTA model that uses a mesoscopic traffic simulator to reproduce complex traffic flow dynamics. In fact, in order for the developed DTA to achieve its functional objective in an urban context, we realized that it was fundamental that the dynamic network loading component be able to reproduce traffic light controls, lane changes and different vehicle classes. Therefore, a new multiclass, multilane, mesoscopic simulation model was developed with these specific characteristics.

8.2 Main Contributions

This section briefly summarizes the main contributions presented in this thesis.

First Contribution: Dynamic Traffic Assignment Model

The first and main contribution of this thesis is a mesoscopic simulation-based DTA model based on DUE behavior.

As shown in Figure 8.2.1, the proposed DTA scheme iterates between the two main components until the convergence criterion is satisfied. These two components are referred to as dynamic network loading and path flow reassignment. In addition, this proposal includes a time-dependent shortest path component, in order to add new paths throughout the procedure when it becomes necessary. A K-static shortest path algorithm that determines an initial path set is also used in the first iteration of the process.

It is important to note here that the proposed scheme of the developed DTA model was designed in such a way that it is relatively easy to substitute different components for others with the same associated. For example, in Chapter

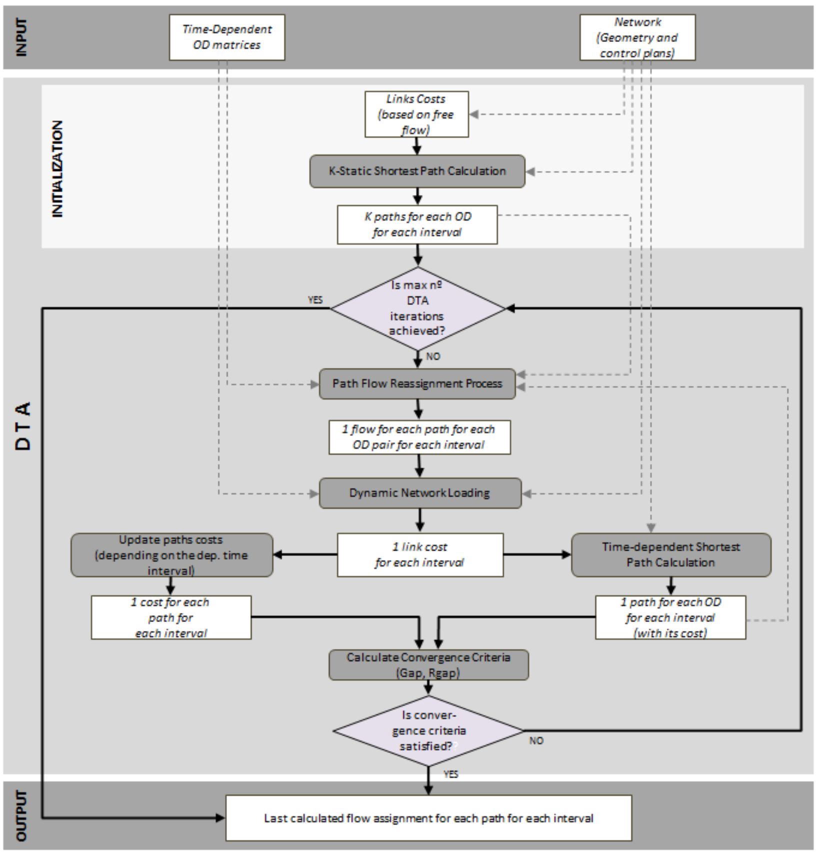

INPUT
Time-Dependent OD matrices
Network (Geometry and control plans)
INITIALIZATION
Links Costs (based on free flow)
K-Static Shortest Path Calculation
K paths for each OD for each interval
DTA
Is max nº DTA iterations achieved?
YES
NO
Path Flow Reassignment Process
1 flow for each path for each OD pair for each interval
Dynamic Network Loading
Update paths costs (depending on the dep. time interval)
1 link cost for each interval
Time-dependent Shortest Path Calculation
1 cost for each path for each interval
1 path for each OD for each interval (with its cost)
Calculate Convergence Criteria (Gap, Rgap)
Is convergence criteria satisfied?
NO
YES
OUTPUT
Last calculated flow assignment for each path for each interval

Figure 8.2.1: Structure of the proposed DTA model.

6 we validate our proposed flow reassignment method with external dynamic network loading, time-dependent and K-static shortest path algorithms. In this particular case, we adopted all these methods from the mesoscopic software MEZZO.

Second Contribution: Mesoscopic Traffic Simulator

DTA models based on simulation use a traffic simulator to mimic the complex traffic flow dynamics, which is the dynamic network loading component of the previously commented scheme. As a result, the second main contribution of this research is a mesoscopic traffic simulation model that is well suited for embedding into the developed DTA model.

This mesoscopic simulator considers a continuous-time link-based approach with complete demand discretization. Considering the disaggregated treatment of each individual vehicle allowed us to use different classes of vehicles in the DTA problem. Moreover, the proposed model allows longitudinal discretization of links in lanes in order to reproduce the traversal movements described by vehicles performing lanes changes, which can considerably augment congestion in the network links. Thus, a complete and new multiclass, multilane, mesoscopic traffic simulator was developed in this dissertation.

Furthermore, the computational experiment performed in Chapter 5 concluded that the developed simulation model was able:

- To reproduce the fundamental diagram,
- To respect the propagation of congestion, ensuring correct temporal and spatial location of the congestion,
- To reproduce traffic behavior in a similar way as a microscopic simulator, except for the case of the roundabout behavior.

Third Contribution: Modification of the MSA Flow Reassignment

One of the main components of the proposed DTA scheme is the flow reassignment method used to distribute the flow at each iteration of the global process. This dissertation proposes a modification of the well known Method of Successive Averages (MSA) for this flow reassignment process. During the flow diversion process, this developed extension takes into account the cost of the alternative paths, unlike the classical MSA which does it indiscriminately regardless of whether or not the path is the worst or only slightly worse than the new optimal path.

So, in order to perform a more realistic flow reassignment among the alternative paths throughout the iterative scheme, we developed a new MSA that uses a diversion factor based on a logit distribution according to travel times. In this case, we adapted the idea of Varia and Dhingra about using this logit distribution in accordance with instantaneous travel time. In our case, we used actual travel times instead of instantaneous ones in order to take advantage of the information provided by the previous dynamic network loading in the global DTA scheme. This travel time information on all the links at each time interval was a considerable improvement on the classical approach. Moreover, in order to reduce the computational storage needed to save all the paths used in the original MSA, the proposed method also combined the proposed diversion factor with the popular limitation on the maximum number of available paths for each OD pair for each departure time interval.

The computational experiments performed in Chapter 6 showed that this proposal produced similar or sometimes better results than other proposals in the literature. In this case, we took advantage of the previously mentioned flexible design of the proposed DTA scheme, and we carried out these experiments by integrating the new proposed MSA with other external components from MEZZO.

Fourth Contribution: Literature Review of Simulation-based DTA Models

In order to accomplish the thesis objectives, we first needed to investigate in detail the relevant DTA models based on simulations in the literature. To the best of our knowledge, there did not exist a state of the art for simulation-based DTA models, at least not at the level of detail we needed. So, the first challenge of this dissertation was to create a specific one.

Therefore, another contribution of this thesis was a complete literature review of the three types of DTA models based on simulation. They were distinguished based on the level of detail with which they represent the studied system, from low to high fidelity: macroscopic, mesoscopic and microscopic simulation models. For each examined DTA model, the in-depth review analyzed not only the approach used to reassign the flow, but also the dynamic network loading component.

The thirteen DTA models discussed are (in alphabetical order):

Aimsun
CONTRAM
DRACULA
Dynameq
DynaMIT
DYNASMART
DynusT
INDY
INTEGRATION
METANET
METROPOLIS
MEZZO
VISTA

Fifth Contribution:

Aside from these main contributions, we will answer the **research questions** presented in Chapter 1.

- *The original MSA indiscriminately diverts traffic from the paths used in the last iteration to the current optimum paths. The process extracts the same amount of flow from each used path, regardless of whether the path is the worst or only slightly worse than the optimal. Does this diversion process affect the achievement of DUE?And, if so, in which way?*

  One of the computational experiments in Chapter 6 compared classical MSA with our developed MSA modification. We remind the reader here that our proposal, in contrast to the original MSA, took into account the cost of the alternative paths when it diverted the flow. Thus, the main part of the flow assigned to the new shortest path came from the worst paths, i.e., from the paths with higher costs.

  It is important to note that we performed this special flow reassignment by using a diversion factor which was based on a logit distribution in accordance with the actual travel times from the DTA scheme's previous dynamic network loading. In this way, we took advantage of the information collected throughout the global process.

  The experiment showed that the modified MSA produced better results than the original approach. Thus, we can conclude that an indiscriminate standard flow reassignment may have effects on achieving DUE, particularly in the number of DTA iterations needed.

- *From a transportation modeling standpoint, is it realistic to accept unlimited growth in the number of OD paths? And, if not, how does an upper bound in the number of paths for each OD pair for each departure time interval affect the quality of the reassignment process and its convergence?*

  In the computational experiments in Chapter 6, we observed that the flow reassignment process had good behaviour, regardless of whether or not it used limitations in the number of paths for each OD pair for each interval.

  In the global Gap measure case, all the experiments generated solutions whose quality was independent of the limited number of paths. So, the limitation solution was able to reduce the computational storage needed in the classical MSA without reducing the good performance of the process.

  Regarding the results obtained for the Relative Gap convergence measure, all the experiments obtained better results by limiting the maximum number of available paths for each OD pair for each interval.

- *In the MSA approach, how does the initial number of paths through each OD pair for each interval influence convergence towards dynamic user equilibrium?*

  One of the experiments in Chapter 7 investigated how deciding on the initial number of paths for each OD pair for each interval influences DUE convergence in the developed DTA model. At the initialization phase of this model, a set of shortest paths is required from each origin to each destination for each departure time interval. These path sets are calculated using a K-static shortest path algorithm (Section 4.3). Depending on the network characteristics, we must decide on the most convenient number of paths during the network calibration process. In fact, in certain cases, a bad decision may result in false congestion at the beginning of the process, making DUE convergence more difficult.

  In the performed computational experiment, we graphically compared convergence measures obtained from the three experiments, in which each of them had one different parameter option. We concluded that the experiment that began with the largest excess of paths was the experiment with more stable results for Gap and RGap measures. This was possible because the developed DTA model can remove paths during the process. If some path flow becomes close to zero at certain iterations of the global process, then this path is removed from the corresponding path set. So, in the next DTA iteration the total demand will be distributed among the remaining paths of the set.

All the proposed options achieved DUE convergence by using almost the same number of paths, but not with the same level of stability. Starting with an excess of paths and removing them when it becomes necessary, appeared to be the most stable option. In conclusion, the initial number of paths for each OD pair for each interval at the beginning of the DTA process has a key role in obtaining a stable convergence.

- *The convergence of the MSA approach depends on the values assigned to the structural parameters of the algorithm (MSA parameter). What is the influence of the different alternatives on convergence to dynamic user equilibrium?*

  In the computational experiment of Chapter 6, we observed surprising results when we used MSA parameters which intuitively seemed to be bad for achieving DUE convergence. We suspected that the MSA parameter was not relevant to the procedure when the considered demand is low.

  In the last computational experiment of Chapter 7, the objective was to test the real importance of the MSA parameter in the DTA process, particularly in achieving DUE convergence with the MSA flow reassignment. We considered three different MSA parameters (the same as in the computational experiments in Section 6.4), and we executed each experiment with two different demands: normal and low.

  For the standard demand, only the experiment that used MSA parameter $\lambda_k = \frac{1}{k+1}$ (where k was the DTA iteration) achieved good results. The other two options, which intuitively seemed unsuitable, provided the expected bad results. In contrast, all three MSA options in the low-demand experiments began with very good convergence measures and quickly fell to practically zero. Thus, the obtained Gap and RGap measures clearly show that the volume of the demand assigned to the network must be taken into account when selecting the MSA parameter. Furthermore, when we need some computational experiment to test a new proposal of the MSA parameter, it would be prudent to test the network with a high volume demand, because very low demand scenarios will accept any MSA parameter.

- *Can this approach adequately reproduce the phenomena observed in macroscopic traffic flow behavior? (For example, shock waves and the traffic fundamental diagram?) In addition, can the model that represents the node behaviour accurately reproduce delays and queue spill-backs?*

  In the computational experiment of Chapter 5, we demonstrated that the macroscopic fundamental diagram can be accomplished with the proposed mesoscopic simulation model. The results obtained show that traffic behavior on the links of the test network follow theoretical macroscopic relationships.

  Furthermore, in the second experiment of the same Chapter, we studied whether or not the developed model properly reproduces the propagation of congestion. The results obtained show that the proposed approach correctly reproduces shock waves, ensuring the correct temporal and spatial location of congestion at the link level.

  We conclud that the proposed approach adequately reproduces the phenomena observed in macroscopic traffic flow behavior, including delays and queue spill-backs.

- *Which is the most appropriate characterization to describe these temporal dynamics? Time-based or event simulation?*

  In Section 5.4, we explained that there are generally two primary approaches to building simulation models: time-step and event-based. These two paradigms are fundamentally different approaches in how they manage time. In a time-step model, time is the independent variable; while in an event-based model, time is a dependent variable.

  Because the proposed mesoscopic simulation model considers the flow propagation process from vehicle to vehicle, and because it is formulated from a space-dependent time relationship, the most suitable approach is the event-based paradigm, which coincides with the idea of considering time as a dependent variable.

  In addition, throughout this thesis we have particularly demonstrated the importance of efficiency in DTA. Adopting an event-based simulator in our DNL component can potentially be more computationally efficient than using the other, time-based simulation approach.

- *How important is the model representation of the network topology? For instance, the experiment shows that roundabouts and short-length links could affect the results of the urban traffic simulation.*

  During the validation of the developed mesoscopic traffic simulator, we detected special traffic behaviour at roundabouts and their adjacent links. One of the computational experiments in Chapter 5 tested the developed model against a microscopic traffic simulator using the real urban network of Amara (Spain). In general, the results obtained looked promising. Although the proposed model took into account considerably fewer parameters than the microsimulator, the reproduced traffic behavior was very similar in both cases, except for roundabout behavior. Moreover, the developed mesoscopic simulator overestimated density on short links (less than 20 meters) when congestion appeared in the network (compared to benchmark microsimulator measurements).

  In conclusion, the developed mesoscopic model encountered some problems when executed on networks that included roundabouts and short links in their topology. These scenarios are very typical in European cities, thus we could not avoid this problem. However, it will be handled promptly. This is the reason we include in the following section, Further Research, a suggestion to improve the model by including a specific treatment of this problematic topology: roundabouts and short links.

8.3 Further Research

In this final section we discuss directions for future research which naturally follow the research described in this thesis.

One of the issues that need further investigation is the treatment of roundabouts and short links (less than 20 meters) in the developed mesoscopic traffic simulation model. As we explained before, in the computational experiments which validated the dynamic network loading component, the only observed significant difference between our results and those of a microscopic simulator was the overestimated density that our simulator produces on short links and roundabouts. Although our results were reasonably similar to those of the benchmark microscopic simulator, we think there is an opportunity for improved investigation of the special behaviour of the proposed simulation for different network topologies.

A further step in this research is to full validation the developed model against real traffic data (counts, speeds, travel times, etc.).

In addition, test the developed models on larger networks (e.g., the cities of Barcelona or Vitoria in Spain) in order to clearly highlight the effects of some of the developed proposals. For instance, our proposed modification of the MSA flow reassignment process improves computational storage by limiting the number of paths for each OD pair for each interval. This improvement should be further evaluated on larger networks.

Another issue that should be focus in a future work is a comparison between models based on different behaviour hypothesis: DUE vs. Stochastic DUE (preventive vs. reactive methods). This would be relevant in the current state of the art of the dynamic traffic assignment models. In the DUE case, the objetive is to achieve a situation where all the vehicles for certain OD pair departing in the same interval have experienced the same and minimal travel times. While in the case of SDUE, its equilibrium is defined in terms of a fixed point in terms of route flows, where the path may have different and non-minimal costs. It is for that reason that I consider that this comparison could be useful for the state of the art if we use real traffic data to evaluate the correctness of the two different behaviour approaches.

Moreover, this computational experience would have significant value if it would be done on different scenarios and with different DTA models. If the experiment would be a comparison of only two models or on only one scenario, the obtained conclusions could not be generalized. The used models and the scenarios can bias the obtained results. For instance, comparing two DUE models we often observe that the resuls are not equal. And moreover, the location of the network and the used demand could concern the decision of what behaviour hypothesis best fits the real behavior of the system.

Another direction of research following from this thesis is to restart one of the flow reassignment methods summarized in the literature review section of Chapter 6: projection methods. For some time now, our research group has addressed asymmetric static traffic assignment problems by using methods based on projection procedures. So, it seems logical that we should try to yield synergies between both problems: DTA and asymmetric assignment. Since the analytical formulation of DUE generalizes variational inequalities with time discretization, approximating this problem with projection procedures in the space of the paths seems to be a suitable approach with promising prospects.

A good starting point would be the algorithm proposed by Wu (1991) in his dissertation, and his corresponding follow-up work in 1998 (Wu et al. (1998)). In addition, we would study the most recent work of Mahut et al. (2007), in which they present an algorithm that operates in the path flow space, which is very attractive for adapting the equivalent projected gradient and reduced gradient algorithms.

Another line of research to be explored using our proposed DTA scheme is the joint calibration of the OD estimation process and the DTA model. The value of DTA models depends on their ability to accurately replicate conditions for the specific network being studied. The DTA model outputs are governed by a set of parameters that must be estimated before the models are applied, in particular the OD matrices that represent traffic demand. The calibration of both components, demand and supply, has been attempted generally through a sequential procedure.

One possible starting point would be the dissertation of Balakrishna (2006). In this work he proposed the calibration problem as a minimization problem whose objective non-linear function is a goodness-of-fit function of the parameters used. He proposed three algorithms to solve the problem: box-complex, SNOBFIT and SPSA. He concluded that the latter is the most efficient. For some time now, our research group has addressed the dynamic OD matrix estimation problem by using methods based on the SPSA algorithm. So, following this line of research is clearly needed.

Finally, although it was far beyond the scope of this dissertation, the experience of this work has shown us that there are great opportunities for reducing the computational efforts of the developed tools. All the developments included in this thesis are academic works, so the computational efficiency is less than we would like. However, they are usually used as research tools and, since the results look promising, they are strong candidates for improvements in more efficient and sophisticated tools.

8.4 Related Presentations and Publications

- Comparison on path flow reassignment methods for dynamic traffic assignment based in mesosimulation. M.P. Linares, J. Barceló. LATSIS 2012 (1st European Symposium on Quantitative Methods in Transportation Systems), September 2012, EPFL, Lausanne, Switzerland.

- An approach to multiclass mesoscopic simulation based on individual vehicles for dynamic network loading. M.P. Linares, C. Carmona, J. Barceló, O. Serch, O. Mazariegos. 16th International IEEE Conference on Intelligent Transport Systems (ITS for All Transportation Modes), October 2013, The Hague, The Netherlands.

- An approach to multiclass mesoscopic simulation based on individual vehicles for dynamic network loading. M.P. Linares, C. Carmona, J. Barceló, O. Serch, O. Mazariegos. Accepted for publication in the Proceedings of 16th International IEEE Conference on Intelligent Transport Systems (ITS for All Transportation Modes), October 2013, The Hague, The Netherlands.

- A mesoscopic simulation based dynamic traffic assignment model. M.P. Linares, C. Carmona, J. Barceló, O. Serch. Accepted for presentation at the 93rd TRB Annual Meeting, January 2014, included in the Compendium of Papers.

- A mesoscopic simulation based dynamic traffic assignment model. M.P. Linares, C. Carmona, J. Barceló, O. Serch. Accepted for presentation at the IFORS 2014, Barcelona, July 2014.

Bibliography

Ahuja, R. K., Orlin, J. B., Pallottino, S., and Scutella, M. G. (2003). Dynamic shortest paths minimizing travel times and costs. *Networks*, 41(4):197–205.

Akcelik, R., Besley, M., and Roper, R. (1999). *Fundamental relationships for traffic flows at signalised intersections*. Number ARR 340.

Akiva, M. E. B. and Lerman, S. R. (1985). *Discrete choice analysis: theory and application to predict travel demand*, volume 9. MIT press.

Ashok, K. and Ben-Akiva, M. E. (1993). Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems. In *International Symposium on the Theory of Traffic Flow and Transportation (12th: 1993: Berkeley, Calif.). Transportation and traffic theory.*

Astarita, V. (2002). Node and link models for network traffic flow simulation. *Mathematical and computer modelling*, 35(5):643–656.

Balakrishna, R. (2006). *Off-line calibration of dynamic traffic assignment models*. PhD thesis, Massachusetts Institute of Technology.

Barceló, J. (2010). *Fundamentals of traffic simulation*, volume 145. Springer.

Barceló, J. and Casas, J. (2002). Heuristic dynamic assignment based on microscopic traffic simulation. In *Proceedings of the 9th Meeting of the Euro Working Group on Transportation.*

Barceló, J. and Casas, J. (2004). Heuristic dynamic assignment based on AIMSUN microscopic traffic simulator. In *5th Triennial Symposium on Transportation Analysis, Guadeloupe.*

Barceló, J. and Casas, J. (2006). Stochastic heuristic dynamic assignment based on Aimsun microscopic traffic simulator. *Transportation Research Record: Journal of the Transportation Research Board*, 1964(1):70–80.

Barceló, J., Casas, J., García, D., and Perarnau, J. (2006). A hybrid simulation framework for advanced transportation analysis. In *Control in Transportation Systems*, volume 11, pages 497–502.

Barcelo, J., Casas, J., Garcia, D., and Perarnau, J. (2007). A methodological approach combining macro, meso and micro simulation models for transportation analysis. In *11th World Conference on Transport Research*.

Barceló, J., Montero, L., Bullejos, M., Serch, O., and Carmona, C. (2012). A Kalman Filter approach for the estimation of time dependent od matrices exploiting bluetooth traffic data collection. In *Transportation Research Board 91st Annual Meeting*, number 12-3843.

Beckmann, M., McGuire, C., and Winsten, C. B. (1956). Studies in the economics of transportation. Technical report, Rand Corporation.

Ben-Akiva, M. and Bierlaire, M. (1999). Discrete choice methods and their applications to short term travel decisions. In *Handbook of transportation science*, pages 5–33. Springer.

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

Biham, O., Middleton, A. A., and Levine, D. (1992). Self-organization and a dynamical transition in traffic-flow models. *Physical Review A*, 46(10):R6124–R6127.

Bliemer, M., Versteegt, H., and Castenmiller, R. (2004). Indy: a new analytical multiclass dynamic traffic assignment model. In *Proceedings of the TRISTAN V conference, Guadeloupe*.

Bliemer, M. C. and Bovy, P. H. (2003). Quasi-variational inequality formulation of the multiclass dynamic traffic assignment problem. *Transportation Research Part B: Methodological*, 37(6):501–519.

Boyce, D. E., Ran, B., and Leblanc, L. J. (1995). Solving an instantaneous dynamic user-optimal route choice model. *Transportation Science*, 29(2):128–142.

Brackstone, M. and McDonald, M. (1999). Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):181–196.

Branston, D. (1976). Models of single lane time headway distributions. *Transportation Science*, 10(2):125–148.

Buckley, D. (1968). A semi-poisson model of traffic flow. *Transportation Science*, 2(2):107–133.

Burghout, W. (2004). *Hybrid microscopic-mesoscopic traffic simulation.* PhD thesis, Department of Infrastructure, Division of Transportation and Logistics, Royal Institute of Technology.

Burghout, W., Koutsopoulos, H. N., and Andreasson, I. (2005). Hybrid mesoscopic-microscopic traffic simulation. *Transportation Research Record: Journal of the Transportation Research Board*, 1934(1):218–255.

Carey, M. (1987). Optimal time-varying flows on congested networks. *Operations research*, 35(1):58–69.

Carey, M. (1992). Nonconvexity of the dynamic traffic assignment problem. *Transportation Research Part B: Methodological*, 26(2):127–133.

Carey, M. and Ge, Y. (2012). Comparison of methods for path flow reassignment for dynamic user equilibrium. *Networks and Spatial Economics*, 12(3):337–376.

Carey, M. and Subrahmanian, E. (2000). An approach to modelling time-varying flows on congested networks. *Transportation Research Part B: Methodological*, 34(3):157–183.

Cascetta, E. (2001). *Transportation systems engineering: theory and methods*, volume 49. Springer.

Cascetta, E., Nuzzolo, A., Russo, F., and Vitetta, A. (1996). A modified logit route choice model overcoming path overlapping problems: Specification and some calibration results for interurban networks. In *Proceedings of the 13th International Symposium on Transportation and Traffic Theory*, pages 697–711. Lyon, France, Elsevier Science.

Chabini, I. (1998). Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record: Journal of the Transportation Research Board*, 1645(1):170–175.

Chabini, I. (2001). Analytical dynamic network loading problem: Formulation, solution algorithms, and computer implementations. *Transportation Research Record: Journal of the Transportation Research Board*, 1771(1):191–200.

Chen, H.-K. and Hsueh, C.-F. (1998). A model and an algorithm for the dynamic user-optimal route choice problem. *Transportation Research Part B: Methodological*, 32(3):219–234.

Chiu, Y.-C., Bottom, J., Mahut, M., Paz, A., Balakrishna, R., Waller, T., and Hicks, J. (2010). A primer for dynamic traffic assignment. *Transportation Research Board.*

Codina, E. and Barcelo, J. (1995). Dynamic traffic assignment: Considerations on some deterministic modelling approaches. *Annals of Operations Research*, 60(1):1–58.

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

Cremer, M. and Ludwig, J. (1986). A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, 28(4):297–303.

Dafermos, S. C. (1971). An extended traffic assignment model with applications to two-way traffic. *Transportation Science*, 5(4):366–389.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287.

de Palma, A. and Marchal, F. (1996). METROPOLIS: un outil de simulation du trafic urbain. Technical report, THEMA (THéorie Economique, Modélisation et Applications), Université de Cergy-Pontoise.

De Palma, A. and Marchal, F. (2002). Real cases applications of the fully dynamic METROPOLIS tool-box: an advocacy for large-scale mesoscopic transportation systems. *Networks and spatial economics*, 2(4):347–369.

Dean, B. C. (2004). Algorithms for minimum-cost paths in time-dependent networks with waiting policies. *Networks*, 44(1):41–46.

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412.

Dussault, J.-P., Ferland, J. A., and Lemaire, B. (1986). Convex quadratic programming with one constraint and bounded variables. *Mathematical Programming*, 36(1):90–104.

Elloumi, N., Haj-Salem, H., and Papageorgiou, M. (1994). METACOR: A macroscopic modeling tool for urban corridors. In *Triennal Symposium on Transportation Analysis*, volume 1, pages 135–150.

Florian, M. and Hearn, D. (1995). Network equilibrium models and algorithms. *Handbooks in Operations Research and Management Science*, 8:485–550.

Florian, M., Mahut, M., and Tremblay, N. (2001). A hybrid optimization-mesoscopic simulation dynamic traffic assignment model. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 118–121. IEEE.

Friesz, T. L., Bernstein, D., Smith, T. E., Tobin, R. L., and Wie, B. (1993). A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, 41(1):179–191.

Friesz, T. L., Luque, J., Tobin, R. L., and Wie, B.-W. (1989). Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research*, 37(6):893–901.

Gabay, D. (1983). Chapter in applications of the method of multipliers to variational inequalities. *Studies in mathematics and its applications*, 15:299–331.

Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40.

Gawron, C. (1998). Simulation-based traffic assignment. Technical report, University of Cologne.

Gerlough, D. L. (1956). Simulation of freeway traffic by an electronic computer. *Highway Research Board Proceedings*, page 543.

Gerlough, D. L. and Huber, M. J. (1976). Traffic flow theory. Technical report.

Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111.

Greenshields, B. D., Bibbins, J., Channing, W., and Miller, H. (1935). A study of traffic capacity. In *Highway research board proceedings*.

Han, D. and Lo, H. (2002). New alternating direction method for a class of nonlinear variational inequality problems. *Journal of optimization theory and applications*, 112(3):549–560.

He, B. and Zhou, J. (2000). A modified alternating direction method for convex minimization problems. *Applied Mathematics Letters*, 13(2):123–130.

Helbing, D. (1996). Gas-kinetic derivation of navier-stokes-like traffic equations. *Physical Review E*, 53(3):2366.

Helbing, D. (1997). Modeling multi-lane traffic flow with queuing effects. *Physica A: Statistical Mechanics and its Applications*, 242(1):175–194.

Herman, R., Lam, T. N., and Prigogine, I. (1971). *Kinetic theory of vehicular traffic: comparison with data*. Research Laboratories, General Motors Corporation.

Herrey, E. M. and Herrey, H. (1945). Principles of physics applied to traffic movements and road conditions. *American Journal of Physics*, 13:1.

Hidas, P. (1998). A car-following model for urban traffic simulation. *Traffic engineering & control*, 39(5):300–305.

Hidas, P. (2005). Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C: Emerging Technologies*, 13(1):37–62.

Hoogendoorn, S. and Bovy, P. (2001a). State-of-the-art of vehicular traffic flow modelling, special issue on road traffic modelling and control of the journal of systems and control eng. *Proceedings of the IME I*.

Hoogendoorn, S. P. (1999). *Multiclass continuum modelling of multilane traffic flow*. PhD thesis, TU Delft, Delft University of Technology.

Hoogendoorn, S. P. and Bovy, P. H. (1998). New estimation technique for vehicle-type-specific headway distributions. *Transportation Research Record: Journal of the Transportation Research Board*, 1646(1):18–28.

Hoogendoorn, S. P. and Bovy, P. H. (2001b). Generic gas-kinetic traffic systems modeling with applications to vehicular traffic flow. *Transportation Research Part B: Methodological*, 35(4):317–336.

Janson, B. N. (1991). Dynamic traffic assignment for urban road networks. *Transportation Research Part B: Methodological*, 25(2):143–161.

Kaufman, D. E. and Smith, R. L. (1993). Fastest paths in time-dependent networks for intelligent vehicle-higway systems applications. *Journal of Intelligent Transportation Systems*, 1(1):1–11.

Kennington, J. L. and Helgason, R. V. (1980). *Algorithms for network programming*. John Wiley & Sons, Inc.

Kerner, B. S. and Konhäuser, P. (1994). Structure and parameters of clusters in traffic flow. *Physical Review E*, 50(1):54.

Kometani, E. and Sasaki, T. (1959). Dynamic behaviour of traffic with a non-linear spacing-speed relationship. In *Proceedings of the Symposium on Theory of Traffic Flow*, pages 105–119. Research Laboratories, General Motors, Elsevier.

Kühne, R. D. and Rödiger, M. B. (1991). Macroscopic simulation model for freeway traffic with jams and stop-start waves. In *Proceedings of the 23rd conference on Winter simulation*, pages 762–770. IEEE Computer Society.

Leonard, D., Gower, P., and Taylor, N. B. (1989). CONTRAM: structure of the model. *Research report-Transport and Road Research Laboratory*, (178).

Leventhal, T., Nemhauser, G., and Trotter, L. (1973). A column generation algorithm for optimal traffic assignment. *Transportation Science*, 7(2):168–176.

Lighthill, M. J. and Whitham, G. B. (1955). On kinematic waves. ii. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345.

Lim, Y. and Kim, H. (2005). A shortest path algorithm for real road network based on path overlap. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1426–1438.

Liu, R. (2010). Traffic simulation with dracula. In *Fundamentals of Traffic Simulation*, pages 295–322. Springer.

Lo, H. K. and Szeto, W. (2002). A cell-based variational inequality formulation of the dynamic user optimal assignment problem. *Transportation Research Part B: Methodological*, 36(5):421–443.

Maerivoet, S. (2006). Modelling traffic on motorways: state-of-the-art, numerical data analysis, and dynamic traffic assignment. *Departement elektrotechniek Esat-SCD (SISTA), Katholieke Universiteit Leuven.*

Mahmassani, H., Peeta, S., Hu, T.-Y., and Ziliaskopoulos, A. (1993). Dynamic traffic assignment with multiple user classes for real-time atis/atms applications. In *Large Urban Systems. Proceedings of the Advanced Traffic Management Conference.*

Mahmassani, H. S. (2001). Dynamic network traffic assignment and simulation methodology for advanced system management applications. *Networks and Spatial Economics*, 1(3-4):267–292.

Mahmassani, H. S. and Jayakrishnan, R. (1991). System performance and user response under real-time information in a congested traffic corridor. *Transportation Research Part A: General*, 25(5):293–307.

Mahmassani, H. S. and Liu, Y.-H. (1999). Dynamics of commuting decision behaviour under advanced traveller information systems. *Transportation Research Part C: Emerging Technologies*, 7(2):91–107.

Mahmassani, H. S. and Peeta, S. (1995). System optimal dynamic assignment for electronic route guidance in a congested traffic network. In *Urban Traffic Networks*, pages 3–37. Springer.

Mahmassani, H. S. and Stephan, D. G. (1988). Experimental investigation of route and departure time choice dynamics of urban commuters. *Transportation Research Record: Journal of the Transportation Research Board*, 1203(1203):69–84.

Mahut, M. (1999). Behavioural car following models. Technical report, Centre for Research on Transportation. University of Montreal.

Mahut, M. (2000). *A discrete flow model for dynamic network loading.* PhD thesis, Universite de Montreal.

Mahut, M. and Florian, M. (2010). Traffic simulation with dynameq. In *Fundamentals of Traffic Simulation*, pages 323–361. Springer.

Mahut, M., Florian, M., and Tremblay, N. (2003). Space-time queues and dynamic traffic assignment: A model, algorithm and applications. In *National Research Council (US). Transportation Research Board. Meeting (82nd: 2003: Washington, DC). Compendium of papers CD-ROM.*

Mahut, M., Florian, M., and Tremblay, N. (2007). Comparison of assignment methods for simulation-based dynamic-equilibrium traffic assignment. In *th Triennial Symposium on Transportation Analysis.* Citeseer.

Mahut, M., Florian, M., Tremblay, N., Campbell, M., Patman, D., and McDaniel, Z. K. (2004). Calibration and application of a simulation-based dynamic traffic assignment model. *Transportation Research Record: Journal of the Transportation Research Board*, 1876(1):101–111.

May, A. (1990). Traffic flow fundamentals. *University of California, Berkeley, Prentice Hall*, page 338.

Merchant, D. K. and Nemhauser, G. L. (1978a). A model and an algorithm for the dynamic traffic assignment problem. *Transportation Science*, 12(3):183–199.

Merchant, D. K. and Nemhauser, G. L. (1978b). Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(3):200–207.

Messner, A. and Papageorgiou, M. (1990). Metanet: A macroscopic simulation program for motorway networks. *Traffic Engineering & Control*, 31(8-9):466–470.

Mouskos, K. C., Izadmehr, B., and Ziliaskopoulos, A. K. (2003). Implementation of the visual interactive system for transportation algorithms (vista) in the united states. *Application of Technology in Urban Development.*

Nagel, K., Beckman, R. J., and Barrett, C. L. (1999). Transims for urban planning. In *6th International Conference on Computers in Urban Planning and Urban Management, Venice, Italy.* Citeseer.

Nagel, K. and Herrmann, H. J. (1993). Deterministic models for traffic jams. *Physica A: Statistical Mechanics and its Applications*, 199(2):254–269.

Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de Physique I*, 2(12):2221–2229.

Newell, G. F. (1995). Theory of highway traffic flow, 1945 to 1965. Technical report, Institute of Transportation Studies. University of California.

Newell, G. F. (2002). A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205.

Orda, A. and Rom, R. (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625.

Panwai, S. and Dia, H. (2005). Comparative evaluation of microscopic car-following behavior. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):314–325.

Papageorgiou, M. (1990). Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research Part B: Methodological*, 24(6):471–495.

Papageorgiou, M., Papamichail, I., Messmer, A., and Wang, Y. (2010). Traffic simulation with metanet. In *Fundamentals of Traffic Simulation*, pages 399–430. Springer.

Pardalos, P. M. and Kovoor, N. (1990). An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1-3):321–328.

Parker, M. (1996). The effect of heavy goods vehicles and following behaviour on capacity at motorway roadwork sites. *Traffic engineering & control*, 37(9):524–531.

Patriksson, P. (1994). *The traffic assignment problem: models and methods.* Topics in Transportation.

Paveri-Fontana, S. (1975). On boltzmann-like treatments for traffic flow: a critical review of the basic model and an alternative proposal for dilute traffic analysis. *Transportation Research*, 9(4):225–235.

Payne, H. J. (1971). Models of freeway traffic and control. *Mathematical models of public systems.*

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.

Peeta, S. and Ziliaskopoulos, A. K. (2001). Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3-4):233–265.

Pipes, L. A. (1953). An operational analysis of traffic dynamics. *Journal of applied physics*, 24(3):274–281.

Powell, W. B. and Sheffi, Y. (1982). The convergence of equilibrium algorithms with predetermined step sizes. *Transportation Science*, 16(1):45–55.

Rahme, N. A., White, J., and Stewart, R. (2002). Journey time estimation using midas loop data.

Ran, B. and Boyce, D. E. (1996). Modeling dynamic transportation networks: an intelligent transportation systems oriented approach. *Springer-Verlag, Heidelberg.*

Ran, B. and Shimazaki, T. (1989a). Dynamic user equilibrium traffic assignment for congested transportation networks. In *Fifth World Conference on Transport Research, Yokohama, Japan.*

Ran, B. and Shimazaki, T. (1989b). A general model and algorithm for the dynamic traffic assignment problems. In *Transport Policy, Management & Technology towards 2001: Selected Proceedings of the Fifth World Conference on Transport Rsearch*, volume 4.

Ranney, T. A. (1999). Psychological factors that influence car-following and car-following model development. *Transportation research part F: traffic psychology and behaviour*, 2(4):213–219.

Reuschel, A. (1950). Fahrzeugbewegungen in der Kolonne. *euschel, A. (1950a). Fahrzeugbewegungen in der Kolonne. Wardrop, J. G. (1952). Some Theoretical Aspects of Road Oesterreichisches Ingenieur-Aarchiv*, 4, no.3/4:193–215.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

Sbayti, H., Lu, C.-C., and Mahmassani, H. S. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record: Journal of the Transportation Research Board*, 2029(1):22–30.

Sheffi, Y. (1985). Urban transportation networks: equilibrium analysis with mathematical programming methods. *Englewood Cliffs, NJ.*

Smith, M. (1979). The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, 13(4):295–304.

Smith, M. and Ghali, M. (1992). Two new models for assessing urban traffic control and road pricing strategies. *Traffic Engineering and Control*, 33(4).

Snelder, M. (2009). A comparison between dynameq and indy. Technical report, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport.

Taylor, N. B. (2003). The contram dynamic traffic assignment model. *Networks and Spatial Economics*, 3(3):297–322.

Tong, C. and Wong, S. (2000). A predictive dynamic traffic assignment model in congested capacity-constrained road networks. *Transportation Research Part B: Methodological*, 34(8):625–644.

Underwood, R. (1961). Speed, volume and density relationships. Quality and theory of traffic flow. *Bureau of Highway Traffic.*

Van Aerde, M., Hellinga, B., Baker, M., and Rakha, H. (1996). Integration: An overview of traffic simulation features. *Transportation Research Records.*

Varia, H. and Dhingra, S. (2004). Dynamic user equilibrium traffic assignment on congested multidestination network. *Journal of transportation engineering*, 130(2):211–221.

Wang, J. (2005). A simulation model for motorway merging behaviour. In *Transportation and Traffic Theory. Flow, Dynamics and Human Interaction. 16th International Symposium on Transportation and Traffic Theory.*

Wang, Y., Messmer, A., and Papageorgiou, M. (2001). Freeway network simulation and dynamic traffic assignment with METANET tools. *Transportation Research Record: Journal of the Transportation Research Board*, 1776(1):178–188.

Wardell, W. and Ziliaskopoulos, A. (2000). A intermodal optimum path algorithm for dynamic multimodal networks. *European Journal of Operational Research*, 125:486–502.

Wie, B.-W. (1989). An application of optimal control theory to dynamic user equilibrium traffic assignment. *Transportation Research Record: Journal of the Transportation Research Board*, 1251(1251):66–73.

Wie, B.-W., Tobin, R. L., Bernstein, D., and Friesz, T. L. (1995). A comparison of system optimum and user equilibrium dynamic traffic assignments with schedule delays. *Transportation Research Part C: Emerging Technologies*, 3(6):389–411.

Wilson, R. E. (2001). An analysis of Gipps's car-following model of highway traffic. *IMA journal of applied mathematics*, 66(5):509–537.

Wisten, M. and Smith, M. (1997). Distributed computation of dynamic traffic equilibria. *Transportation Research Part C: Emerging Technologies*, 5(2):77–93.

Wu, J. H. (1991). *A study of monotone variational inequalities and their application to network equilibrium problems*. PhD thesis, Centre de Recherche sur les Transports, Université de Montréal.

Wu, J. H., Chen, Y., and Florian, M. (1998). The continuous dynamic network loading problem: a mathematical formulation and solution method. *Transportation Research Part B: Methodological*, 32(3):173–187.

Yeo, H., Skabardonis, A., Halkias, J., Colyar, J., and Alexiadis, V. (2008). Oversaturated freeway flow algorithm for use in next generation simulation. *Transportation Research Record: Journal of the Transportation Research Board*, 2088(1):68–79.

Zhu, D. L. and Marcotte, P. (1996). Co-coercivity and its role in the convergence of iterative schemes for solving variational inequalities. *SIAM Journal on Optimization*, 6(3):714–726.

Ziliaskopoulos, A. K. (2000). A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science*, 34(1):37–49.

Ziliaskopoulos, A. K. and Mahmassani, H. S. (1993). Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record: Journal of the Transportation Research Board*, 1408(1408):94–100.