Title
Routing strategies for the reliable and efficient utilization of road networks

Author
Samaranayake, Samitha

Publication Date
2014

Peer reviewed|Thesis/dissertation

Routing strategies for the reliable and efficient utilization of road networks

by

Samitha Samaranayake

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

Graduate Division
of the
University of California, Berkeley

Committee in charge:
Professor Alexandre M. Bayen, Chair
Professor Roberto Horowitz
Professor Satish Rao
Professor Raja Sengupta

Fall 2014

Routing strategies for the reliable and efficient utilization of road networks

Abstract

Routing strategies for the reliable and efficient utilization of road networks

by

Samitha Samaranayake
Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Alexandre M. Bayen, Chair

The research presented in this dissertation aims to develop computationally tractable models and algorithms for the reliable and efficient utilization of capacity restricted transportation networks via route selection and demand redistribution, motivated by the fact that traffic congestion in road networks is a major problem in urban communities. Three related topics are considered, 1) route planning with reliability guarantees, 2) system optimal dynamic traffic assignment, and 3) controlling user equilibrium departure times.

Route planning can in many practical settings require finding a route that is both fast and reliable. However, in most operational settings, only deterministic shortest paths are considered, and even when the link travel-times are known to be stochastic the common approach is to simply minimize the expected travel-time. This approach does not account for the variance of the travel-time and gives no reliability guarantees. In many cases, travelers have hard deadlines or are willing to sacrifice some extra travel-time for increased travel-time reliability, such as in commercial routing applications where delivery guarantees need to be met and perishables need to be delivered on time. The research presented in this dissertation develops fast computation techniques for the reliable routing problem known as the stochastic on-time arrival (SOTA) problem, which provides a routing strategy that maximizes the probability of arriving at the destination within a fixed time budget.

Selfish user optimal routing strategies can, however, lead to very inefficient traffic equilibria in congested traffic networks. This "Price of Anarchy" can be mitigated using system optimal coordinated routing algorithms. The dissertation considers the system optimal dynamic traffic assignment problem when only a subset of the network agents can be centrally coordinated. A road traffic dynamics model is developed based on the Lighthill-Williams-Richards partial differential equation and a corresponding multi-commodity junction solver. Full Lagrangian paths are assumed to be known for the controllable agents, while only the aggregate split ratios are required for the non-controllable (selfish) agents. The resulting non-linear optimal control problem is solved efficiently using the discrete adjoint method.

Spill-back from under-capacitated off-ramps is one of the major causes of congestion during the morning commute. This spill-back induces a capacity drop on the freeway, which then creates a bottleneck for the mainline traffic that is passing by the off-ramp. There-

fore, influencing the flow distribution of the vehicles that exit the freeway at the off-ramp can improve the throughput of freeway vehicles that pass this junction. The dissertation studies the generalized morning commute problem where vehicles exiting the freeway at the under-capacitated off-ramp have a fixed desired arrival time and a corresponding equilibrium departure time schedule, and presents strategies to manipulated this equilibrium to maximize throughput on the freeway via incentives or tolls.

To my parents, Sriya and Kithsiri Samaranayake

Contents

List of Figures

List of Tables

Acknowledgments

The research presented in this dissertation would not have been possible without the help, support and contributions of many people during the last five years. I would like to take this opportunity to express my deepest gratitude to everyone involved.

I would like to start by thanking my advisor Professor Alexandre Bayen for giving me the opportunity to work in his research group and all the support he has given throughout my time at Berkeley. He guided and oriented my PhD research with his vision, inspired me with his boundless energy and work ethic, enabled collaborations by helping build research partnerships, and made the whole experience an immensely enjoyable one by accommodating a good work-life balance. I am greatly indebted to him for all his advice, support and camaraderie, both on a professional and personal level.

A major benefit of being at Berkeley has been the ability to interact with and learn from many domain experts. I was fortunate to have the opportunity to work with Professor Roberto Horowitz and his research group on traffic control problems. Discussing research problems with Professor Horowitz has always been a great experience, both due to the deep knowledge he has on the intricacies of the subject and the enthusiasm with which he engages in research problems.

I would like to thank Professor Raja Sengupta for chairing my qualifying exam committee and the time spent discussing a variety of topics and problems. His insights based on his vast knowledge of a wide range of disciplines has helped me look at research problems from many different angles. I would also like to express my gratitude to Professor Steven Glaser for pioneering the Systems Program at Berkeley and giving me the opportunity to be part of it, and for his valuable guidance during the ClearSky project.

Professor Satish Rao has been a great resource for discussing research problems related to network flows. I would like to thank him for sharing his insights on these research problems and the discussions related to pedagogy, all of which I have enjoyed thoroughly. It is not a surprise that Professor Rao's course on Parallel Algorithms was also one of the most engaging courses I have taken.

Professor Alex Skarbadonis is a walking encyclopedia of transportation research and practice. I would like to thank him for the openness with which he always shared his insights and for helping make sure that my research efforts had the right balance between theory and practical relevance.

I had the privilege to visit Professor Paola Goatin's research group at INRIA Sophia-

Antipolis in the Fall of 2012. This visit led to a very fruitful research collaboration with Professor Goatin and her graduate student Maria-Laura Delle Monache. The interactions with Professor Goatin have helped deepen my knowledge in the mathematical aspects of traffic modeling. I would also like to thank Professor Peter Sanders for the opportunity to visit his research group in Karlsruhe and both Moritz Kobitzsch and Dennis Schieferdecker for the subsequent collaborations on stochastic route planning.

During the PhD, I have had the opportunity to work with a number of industry collaborators and partners on the variety of topics. I would like to thank Mihai Stroe, Lucien Pech, and Mathilde Hurand for all their support during my internship with the Google Transit team in Zurich and their help with subsequent research collaborations. Thank you to Mark Dilman and his group for the opportunity to apply some of my PhD research work to database load balancing problems at Oracle, and also Cris Martin for the discussions that led to this. I would also like to thank Ken Tracton from Nokia and Felix Konig from TomTom for their support with the ClearSky and stochastic routing projects.

I would like to take this opportunity to also acknowledge the influence of Dr. Rohit Kapur at Synopsys, who was my first research mentor as a raw undergraduate and masters student. I am deeply appreciative of his mentorship and guidance during those formative years as a researcher, and for all his subsequent support.

During my first semester at Berkeley, I was extremely fortunate to meet and be able to join forces with Sebastien Blandin, who was the perfect student mentor as I started my PhD. I have learned a great deal from his methodical and detailed approach to research, tireless work ethic and humility, which are qualities I have tried to emulate. Working with Sebestien, throughout the years (and hopefully for years to come), has been both a very productive and extremely enjoyable experience.

I have also had a pleasure and privilege to work with a number of other PhD student collaborators during the PhD. A big thanks to Jack Reilly for the countless hours spent discussing traffic models, optimization problems, systems design and of course all the sports, music and other fun activities. I would like to thank Walid Krichene for all the interesting discussions related to the experts algorithm and other optimization problems. Thanks also to Tim Hunter, Dan Work, Andrew Tinka, Ryan Herring, Eloi Pereira, Aude Hofleitner, Chris Claudel, Paul Borokhov, Jerome Thai and Leah Anderson for all your help and insights.

I would like to gratefully acknowledge all the interns and undergraduate researchers that I had the pleasure of mentoring and collaborating with during my PhD. Working with Axel Parmentier, Guillaume Sabran, Jean-Baptiste Lespuai, Mehrdad Niknami, Manuel Jakob and Maleen Abeydeera has been a great experience that I have learned a great deal from and which has been a major influence in my desire to continue in academia. I would like to especially thank Axel Parmentier for the long-standing collaboration that led to the final chapters of this dissertation.

The implementation of the systems described in this dissertation would not have been possible without the contributions from the staff at the California Center for Innovative Transportation (CCIT) and California Partners for the Advanced Transportation Technology (PATH). I would like to thank Tom West for all his support as director at CCIT and PATH,

Joe Butler for keeping the ship afloat and moving in the right direction even when the waters got rough, Gabriel Gomes for all the discussions on traffic modeling/control and help with implementation, and Ethan Xuan for instrumenting the research on the morning commute problem. I would also like to individually thank Thomas Schrieter, Anthony Patire, Dimitrios Triantafyllos, Saneesh Apte, Bill Sappington, Luis Torres and Javier Hernandez for their contributions and Shelley Okimoto, Rosita Alvarez and Helen Bassham for all their help with navigating the system.

Thanks to Charith, Nishad and Yusuf for all the good times in Berkeley, Monterey and Fremont, Tim, Quentin, Charline and Gary, for being the lead social instigators, Justin, Jennifer and Joyce, for keeping the team intact across state lines, and Buddhika, DC, Hasitha, Jon and Nodari, for being the go to guys. A very special thanks to Nina for all the great adventures and patiently keeping me on track in these last few months.

Finally, I would like to thank my parents and my brother Nayana, without whom none of this would have been possible. Thank you for all your love, support, countless sacrifices through the years.

Chapter 1

Introduction

1.1 Motivation

Traffic congestion in road networks is a major problem in cities worldwide. The amount of time wasted due to congestion in the United States alone is estimated to be over 5.5 billion hours per year, which amounts to an economic loss of over 121 billion dollars or roughly 1 percent of the entire U.S. Gross Domestic Product (GDP) [124]. Furthermore, this cost does not account for the other negative externalities of congestion, such as the estimated 56 billion pounds of additional carbon dioxide and other greenhouse gases released into the atmosphere during urban congested conditions [124].

It is estimated that the 600 largest cities will account for 60 percent of the global GDP and 25 percent of the world population by 2025 [44]. With the continuing migration of rural populations towards cities and the rapid urbanization of developing countries, there are huge economic, environmental and societal gains that can be attained by more efficiently utilizing the urban transportation network. Furthermore, roughly 37 percent of the congestion in the United States was estimated to be due to non-recurrent events [124], which illustrates that a large portion of congestion is not caused exclusively by systemic capacity limitations of the network, and emphasizes the need for real-time traffic information systems and adaptive traffic control strategies.

One approach to managing traffic congestion is to provide commuters with reliable real-time traffic estimates and dynamic routing strategies, so that commuters can alter their journeys en-route or plan their routes to avoid non-recurrent congestion as it occurs. The rapid proliferation of GPS-enabled smart phones and navigation systems has made it easier to both obtain real-time traffic conditions and pass this information back to commuters.

However, even with perfect information, the selfish nature in which drivers select routes can lead to very inefficient equilibrium traffic distributions [21]. Therefore, while providing accurate real-time information can lead to reductions in congestion during non-recurrent events, active traffic control mechanisms are also necessary. Thus, another traffic management strategy is to intelligently manage the traffic network via control strategies such as

traffic signal control, freeway ramp-metering, variable speed limit control, tolling and incentives. Such strategies, also known as Intelligent Transportation Systems (ITS), can lead to more efficient utilization of existing network infrastructure.

The research presented in this dissertation is motivated by the need for computationally tractable traffic management and control strategies for efficient utilization existing capacity in transportation networks, with a focus on practical applicability and a pathways for their real-world deployment. To this effect, the models, optimization techniques and algorithms developed have been implemented in the Connected Corridors traffic management system [28] at UC Berkeley and its predecessor the Mobile Millennium traffic information system [95]. The following sections provide some background information on the demand management techniques that are utilized.

1.2 Traffic information systems and reliable routing

One approach to mitigating traffic congestion is to provide commuters with reliable and timely traffic information. This includes both predictive information based on historical demand patters, event information, road closures, weather, etc., and real-time information as the state of the network changes due to unforeseen events such as accidents. Making such information available allows commuters to make informed decisions and increase the efficiency of the road network [27].

The increasing availability of cheap GPS-based traffic information due to the proliferation of smart-phones and GPS-enabled navigation devices has revolutionized the ability to provide such information. Commuters can now easily obtain traffic information via applications provided by Google, Apple, TomTom, Garmin, Waze etc. and via organizations such as 511.org. One of the main features of these applications is the ability to do path planning and route vehicles to their destination according to some shortest path metric. Current state of the art systems provide real-time traffic information and have the ability to dynamically reroute vehicles as the state of the network changes. However, route selection in many practical settings can require both a fast and reliable route, and current navigation systems do not have the ability to provide reliability information on the recommended route.

When the link travel-times are known to be stochastic the simple approach is to find the route with the least expected travel-time (LET) [85]. The LET problem has been well researched and many efficient algorithms exist for different variants of the problem [52, 93, 133]. When the link travel-times are independent and time-invariant distributions, the LET problem can be reduced to the standard deterministic shortest path problem by setting each link weight to its expected value. This problem has been studied extensively since Dijkstra's seminal algorithm from 1959 [43]. Current state of the art solutions to the deterministic shortest path problem can now run in logarithmic query time with polynomial pre-processing [2, 55], and in practice provide sub millisecond query times for continental U.S. sized networks. However, Dijkstra's algorithm and its variants do not provide an optimal solution when the link weights are time-varying [63]. If the network satisfies the first-in first-out (FIFO)

condition [6], the problem can be solved using dynamic programming using time-dependent weights with the original graph [36]. However, solutions to the time-varying shortest path problem are considerably slower than their time-invariant counterparts. A comprehensive summary of deterministic route planning algorithms is provided in [39].

While, as mentioned above, efficient algorithms exist for solving the deterministic shortest path problem, the stochastic shortest path problem turn out to be considerably harder to solve, when accounting for the reliability (variance) of the solution. There are many possible definitions for a stochastic shortest path based on the trade-off between the expected travel-time and the variance [48, 105]. For example, one might want to minimize variance subject to a constraint on the expected travel-time, minimize expected travel-time subject to some constraint on the variance or minimize a weighted sum of the expected travel-time and variance. Another natural definition, is to maximize the probability of reaching the destination subject to a travel-time budget [50], which eliminates the need to consider multiple objectives. This formulation is known as the stochastic on-time arrival (SOTA) problem.

The SOTA problem can be solved as a stochastic optimal control [15] problem, which results in an adaptive optimal policy as opposed to an optimal path. An optimal policy generates a node-based decision rule that defines the optimal path from a given node to the destination conditioned on the realized travel-time, and it is clear that such an adaptive policy is necessarily better than a static a-priori solution. With most commuters obtaining navigation instructions in real-time as they drive, as opposed to for example printing direction prior to the trip, an adaptive solution is also a practical approach. For situations where a fixed path is required, a static route can be obtained by solving the related path-based SOTA problem.

This SOTA formulation can be extended to solve for more general objectives by simply modifying the cost function at the destination [49]. As a result, the solution methods for the SOTA problem discussed in this dissertation can be applied to these generalized utility functions without requiring any modification, further motivating the search for efficient solutions.

The first part of this dissertation (Chapters 2 and 3) is devoted to speedup techniques for efficiently computing the optimal policy for the SOTA problem, so that the solution methods are fast enough for practical implementation in commercial navigation systems.

1.3 Dynamic traffic assignment

Dynamic traffic assignment (DTA) is the process of allocating time-varying origin-destination based traffic demand to a set of paths on a road network. This problem has been studied extensively over the last 35 years, since the seminal works of Merchant and Nemhauser [90, 91]. There are two types of traffic assignment that are generally considered in the literature: the user equilibrium or Wardrop equilibrium allocation (UE-DTA), in which users minimize individual travel-time in a selfish manner, and the system optimal allocation (SO-DTA) where a central authority picks the route for each user and seeks

to minimize the aggregate total travel-time over all users. These principles were first presented by Wardrop [134] (in the context of static traffic assignment) and expanded on by Beckman [12]. See [108] for a broad overview of dynamic traffic assignment.

User equilibrium traffic assignment can lead to inefficient network utilization, highlighted by the Braess Paradox [21], where adding capacity to the network can actually result in longer travel times for all users. It has been shown that this paradox can occur in real road networks [72] and that it is hard to design networks that are immune to it [115]. In fact, it can be shown that the price of anarchy (PoA) [74], the worst-case ratio of the system delay caused by the selfish behavior over the system optimal solution, may be arbitrarily large even in simple networks [116, 130]. System optimal traffic assignment on the other hand leads to optimal utilization of the network resources, but is hard to achieve in practice since the overriding objective for individual drivers in a road network is to minimize their own travel-time. It is well known that setting a toll on each road segment corresponding to the marginal delay of the demand moves the user equilibrium towards a system optimal allocation [114, 131]. However, imposing time-varying tolls on each road segment is not practical and tolling in general is difficult to implement in many settings due to both infrastructure and political considerations.

An alternative approach is to attempt to control a fraction of the drivers (via direct control or some incentive scheme) and assign routes via a central authority that tries to minimize system-wide total travel time. This has been studied in the context of Stackelberg routing games [73, 113, 130], where the goal of the central controller is to assign routes to a fraction of the demand using a strategy that minimizes the system wide cost, while anticipating selfish behavior from the demand that is not being controlled. Most of this work has been in the area of communication networks and assumes a non-decreasing latency function and vertical queues. However, these assumptions are generally not satisfied in road traffic networks, with horizontal queuing, because of congestion propagation and more complex latency functions, due to the physics of flows and driver behavior [82, 111, 31, 34]. Therefore, the literature on partial control in traffic assignment is sparse and usually makes simplifying assumptions, such as vertical queues and non-decreasing latency functions [7], or simple networks such as parallel networks [75].

The second part of this dissertation (Chapter 4) considers the SO-DTA problem with partial control for networks with horizontal queuing and latency functions that satisfy the complex traffic dynamics. It should, however, be noted that this work currently does not model the response of the selfish users. It is clear that a change in the network state will result in a response from the selfish users as in a Stackelberg game. Finding the optimal control for a Stackelberg game is NP-Hard in the size of the network for the class of increasing latency functions even in the static case [113] and it is common to use approximate strategies [113, 130]. Efficiently computing the SO-DTA solution with partial control while modeling the response of the selfish users is currently an open problem.

1.4 The morning commute problem

The morning commute problem is the task of finding the user equilibrium (UE) departure times for a set of commuters that travel through a network with bottlenecks and need to reach their destination at a fixed time. The commuters incur a penalty due to both queuing delays in the network and not arriving at the destination on time (being early or late). The case of a single bottleneck with homogeneous flow was studied by Vickrey [131] in his seminal paper from 1969. The Vickrey model is elegant and simple, and has been widely adopted in many settings from equilibrium analysis to time-dependent toll pricing. The key assumptions of the model are that: (1) travelers have identical and piecewise linear cost functions; (2) there is only a single bottleneck and a single route; (3) the adoption of user equilibrium assumes perfect information, rationality, and perfect decision; (4) the queue is vertical and spatial extent of the queue is not considered. Hendrikson and Kocur [64] showed the existence and uniqueness of the equilibrium with linear cost functions and subsequent studies have gradually relaxed the assumptions of the Vickrey model. For example, Smith [127] and Daganzo [32] proved the existence and uniqueness of equilibrium with convex cost functions, Newell [99] and Lindsey [83] relaxed the identical cost functions assumption and introduced groups of commuters, and a numerical solution method was given by Zijpp and Koolstra [40]. Furthermore, the assumption of a single bottleneck and a single route was relaxed by Kuwahara [76] who studied the case of two tandem bottlenecks, and Arnott et al. [4] who studied the case of parallel routes with bottlenecks. Mahmassani and Chang [25, 88] studied the day to day variations of the equilibrium to relax assumption (3). Finally, Mahmassani and Herman [87] (with some corrections by Newell [100]) relaxed assumption (4) by studying the horizontal queuing both upstream and downstream of the bottleneck.

Spill-back from under-capacitated off-ramps is one of the major causes of congestion during the morning commute. Such a bottleneck is a common occurrence at commuter heavy exits close to large corporations, schools, industrial parks etc. An off-ramp bottleneck causes congestion to spill back onto the highway, and thereby reduces the available capacity for vehicles that are continuing on the highway, which then creates an additional bottleneck for the mainline traffic that is passing by the off-ramp [23]. Therefore, influencing the flow distribution of the vehicles that exit the freeway at the off-ramp can improve the throughput for all freeway vehicles that pass this junction. However, there have been limited efforts to study the equilibrium behavior in the case of junctions with bottlenecks such as the case of the congested off-ramp. Lago and Daganzo [77] considers the Vickrey equilibrium at a merge between two highways, and Yperman et al. [139] consider a highway diverge (although the source of congestion is not at the highway diverge).

The off-ramp bottleneck problem can be mitigated either by building additional capacity at the off-ramp to accommodate the peak flow or by altering the demand of vehicles that are exiting at the off-ramp during the peak period. Adding new capacity at a highway off-ramp is extremely disruptive in the short term and incurs a large monetary cost. Furthermore, the appropriate capacity requirement needs to be known in advance and can not be modified easily. Therefore, the use of incentives and tolls to manipulate the equilibrium departure

times of the exiting vehicles can be a more effective mechanism of demand management in this context.

The research presented in third and final part of this dissertation (Chapters 5 and 6) generalizes the morning commute problem to a freeway network with an off-ramp exit and proves existence and uniqueness properties of the equilibrium for a more general set of arrival time cost functions that enable the analysis of incentives and tolls.

1.5 Integrated corridor management

The research presented in this dissertation was conducted in the context of the *Connected Corridors* [28] Integrated Corridor Management (ICM) initiative. Congested freeway corridors are a primary source of traffic delays during peak congestion periods. Decentralized and uncoordinated traffic management strategies across the different stakeholders such as multiple cities and counties that manage the arterial roadways and the Department of Transportation that manages freeways can lead to inefficient control strategies and responses to incidents. Integrated Corridor Management attempts to combine the application of technology with a commitment from the network stakeholders to transform the management of freeway corridors [30, 92], enabled by the recent advancements in intelligent transportation systems (ITS), such as real-time traffic and traveler information, cheap sensing technologies, and cooperation between traffic management agencies.

The *Connected Corridors* project, which is managed by the California Partners for Advanced Transportation Technology (PATH) at the University of California Berkeley, is an ICM project aimed at fundamentally changing the way in which California operates its freeway corridors to improve the safety, efficiency, robustness and reliability of urban transportation systems [28]. The *Connected Corridors* system consists of the following subsystems.

- **Estimation**: Traffic estimation is the basis for many simulation, control and traffic information applications within the system. The lack of accurate estimates of the traffic state will lead to inefficient control strategies and incorrect information being passed on to users. Therefore, it is a critical component of the system. A number of specialized estimation algorithms are used to estimate both velocity and density for freeways and arterials [68, 137].

- **Prediction**: Accurate traffic predictions [65, 138] are important for many applications, such as providing advanced traffic information for commuters and the routing subsystem. They are also a key component in solving control problems, such as optimal ramp metering, which are solved as finite horizon optimal control problems and depend on accurate boundary flow predictions during the control horizon [110].

- **Simulation**: A macroscopic traffic simulator driven by specialized traffic models [69, 97, 137] and model calibration [41] allows for the simulation and analysis of different traffic scenarios.

- **Control**: The models described above are then used to compute control strategies that maximizing the efficiency of the corridor, such as coordinated ramp metering [110], incident based traffic rerouting [122] and traffic signal control.

- **Traffic information**: Finally, the system will provide real-time traffic conditions and traffic routing information directly to commuters via changeable message signs (CMS), a web interface and smart-phone application.

The estimation and traffic information subsystems build upon the *Mobile Millennium* traffic information system [95]. The goal of the *Connected Corridors* system is to eventually integrate these submodules such that the control problems are solved in a coordinated manner instead of solving each problem as an independent optimization problem. Figure 1.1 illustrates as example where traffic flow rerouting from the freeway due to an accident is integrated with the arterial traffic signal control, ramp metering and express lane control systems to best utilize the network capacity.

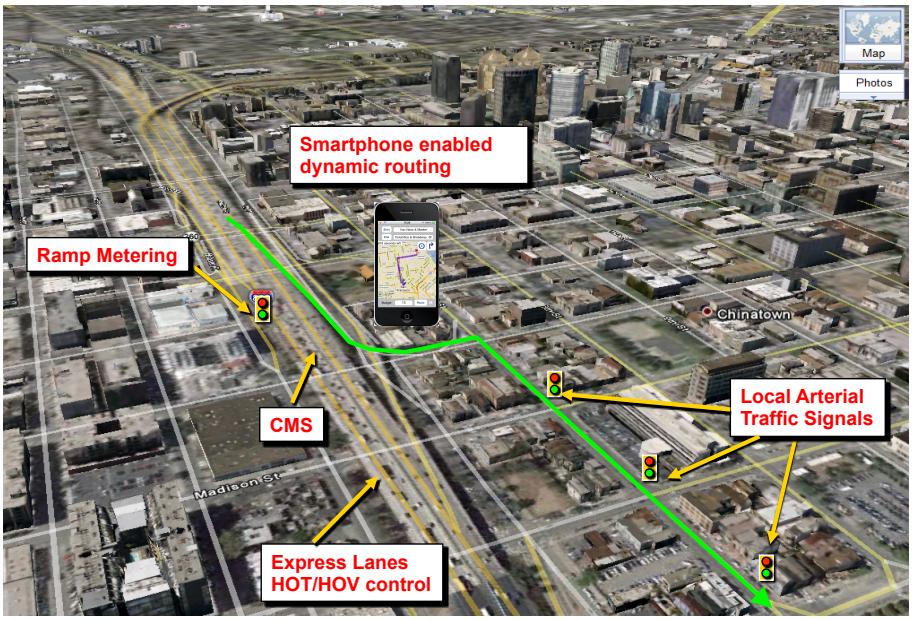


Figure 1.1: An illustration of traffic flow rerouting in the *Connected Corridors* traffic management system. The reroute control strategy is integrated with the arterial traffic signal control, ramp metering and express lane control systems to best utilize the network capacity.

1.6 Summary of contributions

1.6.1 Contributions in the dissertation

This dissertation includes contributions in the areas of mathematical modeling, algorithm design, optimal control and optimization in the context of efficient and reliable flow routing in traffic networks.

The stochastic on-time arrival problem (Chapters 2 and 3)

- Label-setting algorithm for the continuous time stochastic on-time arrival (SOTA) problem, even in cyclic networks, when there is an uniform strictly positive minimum link travel-time [118].
- Fast discrete time solution to the SOTA problem using an optimal ordering algorithm that determines the best order in which to solve the underlying dynamic program [118].
- Method that combines the ideas of the existence of a minimum strictly positive link travel-time and optimal ordering, and the idea of zero-delay convolution [37, 54] to further improve our previous solution to the SOTA problem [119].
- Extensions for time-varying and correlated link travel-time distributions [119].
- Implementation of the above described speedup techniques within the Mobile Millennium traffic information system [118].
- Pre-processing techniques for the SOTA problem that reduce computation time by an order of magnitude [117].

System optimal dynamic traffic assignment with partial control (Chapter 4)

- Formulation of the system optimal dynamic traffic assignment with partial control (SO-DTA-PC) problem as a multi-commodity non-linear optimal control problem using a framework that minimizes the OD data requirements, so that it can be applied in an operational setting [122].
- Adaptation of a multi-commodity traffic dynamics model [53] for the SO-DTA-PC problem with the appropriate junction solver [122].
- Indentification of how the system dynamics of the traffic dynamics model and the discrete adjoint method [46, 56, 57, 109] can be exploited to compute the gradient of the system efficiently.

Controlling user equilibrium departure times (Chapters 5 and 6)

- Mathematical framework for modeling delays in a single-source multi-destination network with analytical expressions of delay.
- Analysis of existence and uniqueness properties of the solution to the delay model [107].
- Extending the solution to the Vickrey morning commute problem of finding a departure time equilibrium [131] to more general cost functions and a diverge junction [123].
- Proof of existence of the equilibrium and requirements for uniqueness of the equilibrium with discontinuous and non-convex arrival time cost functions [123].
- Analysis of optimal incentive and tolling mechanisms for maximizing the equilibrium throughput [123].

1.6.2 Additional contributions

In addition to the main contributions that are presented in this dissertation, several related research collaborations led to significant results that are presented in associated publications.

The stochastic on time arrival problem

- Fast heuristic solution to the path-based SOTA problem using the optimal policy as a heuristic [104].
- Routing system for providing adaptive routing directions on location-aware mobile devices and a corresponding iPhone application [19].
- GPU parallalization of the stochastic on-time arrival problem [1].
- Experimental study of heuristic pruning techniques for the stochastic on-time arrival problem [86].

Optimal control of networked dynamical systems

- Continuous time traffic dynamics model for finite-horizon optimal control problems that require strong boundary conditions [38].
- Adjoint-based optimization on a network of discrete scalar conservation law PDEs with applications to coordinated ramp metering [110].
- Solving the Dynamic User Equilibrium Problem (DUE) via sequential convex optimization for parallel horizontal queuing networks [80].

Traffic forecasting

- Learning the dependency structure of highway networks for traffic forecast [120]

Emissions estimation and dispersion modeling

- Real-time emissions estimation and dispersion modeling along the transportation network [121]

1.7 Organization of the dissertation

The organization of the rest of the dissertation is as follows. The first part (Chapters 2 and 3) is devoted to the stochastic on-time arrival (SOTA) problem, the second part (Chapter 4) considers system optimal dynamic traffic assignment problem with partial control, and the third part (Chapters 5 and 6) focus on controlling user equilibrium departure times during the morning commute. A summary of the individual chapters is given below.

Chapter 2 studies the stochastic on time arrival (SOTA) problem of finding a routing policy that maximizes the probability reaching the destination within a fixed time budget for stochastic networks. A number of strategies are explored for computing the solution efficiently and their complexity is analyzed. Implementation details and numerical results are also presented from the Mobile Millennium traffic information system.

Chapter 3 explores the use of pre-computation strategies to speed up the query time for the SOTA problem inspired by the techniques used for the deterministic shortest path problem. The difficulties in applying these ideas to the stochastic setting are discussed and two speedup techniques that can be adapted are explained. Numerical results are presented to illustrated the effectiveness of these techniques.

Chapter 4 discusses the system optimal dynamic traffic assignment problem (SO-DTA) for general traffic networks with horizontal queuing. A traffic dynamics model based on a Godunov discretization of the the Lighthill-Willams-Richards (LWR) partial differential equation and corresponding multi-commodity junction solver are presented. The problem is posed as a finite horizon optimal control problem and solved via the discrete adjoint method. Numerical results are presented for a synthetic network and an example scenario on interstate I-210 in Southern California.

Chapter 5 presents a mathematical framework for modeling flows in a single source multiple destination network with vertical queuing. The model provides an analytical expression for the delays at each node as a function of the flows at the source of the network and the queuing state at each node. The existence and uniqueness of the solution proved and example applications of the model are presented.

Chapter 6 analyzes the equilibrium departure time problem where a set of vehicles has to travel through a network with capacity restrictions and reach the destination at a fixed time, and the vehicles incur a penalty for both any queuing delays due to the capacity restrictions and arriving at the destination early or late. The problem is studied in the

context of a off-ramp bottleneck. Existence and uniqueness properties of the departure time equilibrium are presented for a general class of delay and arrival time cost functions, which allow for discontinuities in the arrival cost functions. This enables the implementation of step incentives or tolls that are more practical to implement than their continuous counterparts.

The dissertation concludes with some final remarks and a discussion of possible extensions in Chapter 7.

Part I

Reliable routing

Chapter 2

Stochastic on-time arrival (SOTA) problem

2.1 Introduction

The research presented in this chapter aims to improve the computational tractability of solutions to the stochastic on-time arrival (SOTA) problem, where the goal is to determine a routing policy that maximizes the probability of on-time arrival, given an origin destination pair and a desired travel-time budget. Fan et al. [47] formulated the SOTA problem as a stochastic dynamic programming problem and solved it using a standard *successive approximation* (SA) algorithm. In an acyclic network, the SA algorithm converges in a number of steps no greater than the maximum number of links in the optimal path. However, in a network that contains cycles, as is the case with all road networks, the maximum number of iterations required for the algorithm to converge can be unbounded [47]. This is due to the fact that the optimal solution can contain loops, as will be illustrated later. As an alternative, Nie et al. [103] propose a discrete approximation algorithm for the SOTA problem that converges in a finite number of steps and runs in pseudo-polynomial time.

This chapter presents a number of theoretical and numerical results for improving the tractability of the SOTA problem over existing methods. We first solve the unbounded convergence problem, by developing a new label-setting algorithm that gives an exact solution to the SOTA problem and has a provable convergence bound for networks in which the minimum realizable travel-time is greater than zero. As with [47], this algorithm requires computing a continuous-time convolution product, which is one of the challenges of the method. In general, this convolution cannot be solved analytically when routing in arbitrary networks, and therefore a discrete approximation scheme is required. By exploiting the structure of our algorithm, we are able to solve the convolution more efficiently than the standard (brute force) discrete time approximation algorithm used in [103] and obtain a faster computation time. We show that the order in which the nodes of the graph are considered greatly impacts the running time of our proposed solution and present an optimal ordering

algorithm that minimizes the number of convolutions that need to be computed. We then present an algorithm that combines the existence of a minimum realizable travel-time and the optimal ordering with zero-delay convolution [37, 54] to create an even more efficient solution to the SOTA problem. In addition, we present an analysis of the conditions under which our framework can be extended to work with time-varying travel-time distributions and show that these conditions are satisfied in the commonly used travel-time models for road networks. We also consider the problem of correlated travel-time distributions. Finally, we present a simple network pruning scheme that reduces the search space of the algorithm and thus also improves its computational efficiency.

Complexity results are given for all the optimization techniques presented. We also analyze the computation time of the algorithms as a function of the network topology. The algorithms perform best on networks with long road segments and a limited number of loops. Road networks in general consist of arterial networks with short segments and many loops that are connected via a highway network that contains long segments and fewer loops. The implications of this structure for efficient computation of stochastic shortest paths is discussed. Our goal is to provide the theoretical basis for a tractable implementation of adaptive routing with reliability guarantees in an operational setting. Therefore, experimental results are provided for San Francisco Bay Area highway and arterial networks using the Mobile Millennium [95] traffic information system.

The rest of the chapter is organized as follows. In Section 2.2, we define the stochastic on time arrival (SOTA) problem and discuss its classical solution method. Section 2.3 presents the new label-setting algorithm for the SOTA problem, proves its convergence properties and discusses how the algorithm can be used with both time-varying and correlated travel-time distributions. Next, in Section 2.4, we present an efficient numerical method to compute the discrete-time solution to the SOTA problem using an optimal update ordering for solving the underlying dynamic program. Experimental results are given in Section 2.5. In Section 2.6, we describe a method for improving the computational complexity of the problem using a technique known as Zero-Delay-Convolution, and provide more experimental results to illustrate the performance gains.

2.2 Problem formulation

We consider a directed network $G(N, A)$ with $|N| = n$ nodes and $|A| = m$ links. The weight of each link $(i, j) \in A$ is a random variable with probability density function $p_{ij}(\cdot)$ that represents the travel-time on link (i, j). Given a time budget T, an optimal routing strategy is defined to be a policy that maximizes the probability of arriving at a destination node s within time T. A routing policy is an adaptive solution that determines the optimal path at each node (intersection in the road network) based on the travel-time realized to that point. This is in contrast to a-priori solutions that determine the entire path prior to departure. Given a node $i \in N$ and a time budget t, $u_i(t)$ denotes the probability of reaching node s from node i in less than time t when following the optimal policy. At each node i,

the traveler should pick the link (i, j) that maximizes the probability of arriving on time at the destination. If j is the next node being visited after node i and ω is the time spent on link (i, j), the traveler starting at node i with a time budget t has a time budget of $t - \omega$ to travel from j to the destination, as described in equation (2.1)[1].

Definition 2.1. *The optimal routing policy for the SOTA problem can be formulated as follows:*

$$\begin{aligned} u_i(t) &= \max_j \int_0^t p_{ij}(\omega)\, u_j(t-\omega)\, d\omega \\ &\qquad \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ 0 \leq t \leq T \\ u_s(t) &= 1 \qquad 0 \leq t \leq T \end{aligned} \tag{2.1}$$

where $p_{ij}(\cdot)$ is the travel-time distribution on link (i, j).

The functions $p_{ij}(\cdot)$ are assumed to be known and can for example be obtained using historical data or real-time traffic information.

Fan and Nie [47] presented the *successive approximations* (SA) algorithm described in Algorithm 2.1, which solves the system of equations (2.1) and gives an optimal routing policy.

Algorithm 2.1 Successive approximations algorithm [47]

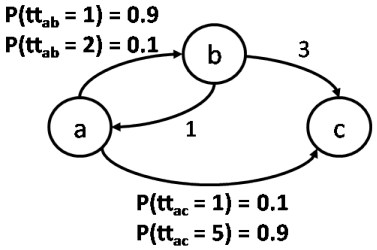
Step 0. Initialization
$k = 0$
$u_i^k(t) = 0,\ \ \forall i \in N,\ \ i \neq s,\ \ 0 \leq t \leq T$ % $u_i^k(t)$ is the approximation of $u_i(t)$
$u_s^k(t) = 1,\ \ 0 \leq t \leq T$ in the k^{th} iteration of the algorithm

Step 1. Update
$k = k + 1$
$u_s^k(t) = 1,\ \ 0 \leq t \leq T$
$u_i^k(t) = \max_j \int_0^t p_{ij}(\omega)\, u_j^{k-1}(t-\omega)\, d\omega,\ \forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ 0 \leq t \leq T$

Step 2. Convergence test
If $\forall (i,t) \in N \times [0,T]\,,\ \ \max_{i,t} |u_i^k(t) - u_i^{k-1}(t)| = 0$ stop;
Otherwise go to Step 1.

[1] In this formulation of the problem, the traveler is not allowed to wait at any of the intermediate nodes. In Section 2.3.2 we state the conditions under which travel-time distributions from traffic information systems satisfy the first-in-first-out (FIFO) condition, and thus waiting at a node cannot improve the on time arrival probability of the modeled traveler.

At each iteration k, $u_i^k(t)$ gives the probability of reaching the destination from node i within a time budget t, using a path with no more than k links, under the optimal policy. The approximation error monotonically decreases with k and the solution eventually reaches an optimal value when k is equal to the number of links in the optimal path. A formal proof of the convergence is given in Section 3 of [47] using the bounded monotone convergence theorem. However, since an optimal routing policy in a stochastic network can have loops (see Example 2.1), the number of iterations required to attain convergence is not known a-priori.



Path	Travel-time	Probability
$\{(a,b),(b,c)\}$	4	0.9
$\{(a,c)\}$	1	0.1
$\{(a,b),(b,a),(a,c)\}$	4	0.01

Figure 2.1: A simple network with an optimal routing policy that may contain a loop. Links (b,c) and (b,a) have deterministic travel-times of respectively 3 and 1 time units. Link (a,b) has a travel-time of 1 with probability 0.9 and a travel-time of 2 with probability 0.1. Link (a,c) has a travel-time of 5 with probability 0.9 and a travel-time of 1 with probability 0.1. The table presents on time arrival probabilities for all possible realizations of optimal paths.

Example 2.1. *Figure 2.1 shows a simple network in which an optimal path can contain a loop. Consider finding the optimal route from node a to node c with a budget of* 4 *time units. There are two choices at the origin, of which it is clear that link* (a,b) *gives the highest probability of reaching the destination on time, since there is a* 0.9 *probability of the travel-time on* (a,b) *being 1 time unit, which leads to a total travel-time of* 4*. However, assume that the realized travel-time on link* (a,b) *is actually 2 time units, which can happen with probability* 0.1*. In this case, taking the path* $\{(a,b),(b,c)\}$ *results in zero probability of reaching the destination on time. The optimal path is therefore* $\{(a,b),(b,a),(a,c)\}$*, which is the only path that has a non zero probability of reaching the destination on time. This optimal path contains a loop. An a-priori algorithm such as the least expected time (LET) algorithm will always route on path* $\{(a,b),(b,c)\}$ *regardless of the realized travel-times and thus have a lower probability of reaching the destination on time.*

As stated in [47], as the network gets arbitrarily complex, it is possible to have an infinite-horizon routing process. This is unlikely to happen in realistic networks [16], but a naive successive approximations strategy can be extremely inefficient and intractable for even moderately large networks.

To solve the problem raised by the unbounded convergence of Algorithm 2.1, Nie and Fan [103] presented a discrete time approximation of the SOTA problem. This algorithm

has a computational complexity of $O(m(Td)^2)$, where m is the number of links in the graph, d is the number of discretization intervals per unit time and T is the time budget. The drawback of this method is the numerical discretization error in the representation of the probability density function. A smaller discretization interval leads to a more accurate approximation, but increases the computation time quadratically. In this chapter, we present both a continuous time exact solution that does not require successive approximations and a discrete time solution with a computation time to approximation error trade off that is lower than in [103].

2.3 Continuous time formulation of the SOTA problem

In this section, we present an algorithm that finds the optimal solution to the continuous time SOTA problem in a single iteration through time space domain of the problem. The complexity of the algorithm does not depend on the number of links in the optimal path.

2.3.1 Solution algorithm for single iteration convergence

The key observation used in this algorithm is that there exists a minimum physically realizable travel-time on each link of the network. Let β be the minimum realizable link travel-time across the entire network. β is strictly positive since speeds of vehicles have a finite uniform bound, and the network contains a finite number of links with strictly positive length. Therefore, given $\epsilon \in (0, \beta)$, $\delta = \beta - \epsilon$ is a strictly positive travel-time such that $p_{ij}(t) = 0 \ \forall\, t \leq \delta, (i,j) \in A$.

Given a time budget T discretized in intervals of size δ, let $L = \lceil T/\delta \rceil$. We propose the solution given in Algorithm 2.2. In this formulation of the SOTA problem, the functions $u_i^k(\cdot)$ are computed on $[0, T]$ by increments of size δ. The proposed algorithm relies on the fact that for $t \in (\tau^k - \delta, \tau^k]$, $u_i^k(t)$ can be computed exactly using only $u_j^{k-1}(\cdot)$, $(i,j) \in A$, on $(\tau^k - 2\,\delta, \tau^k - \delta]$, where τ^k is the budget up to which $u_i^k(\cdot)$ is computed at the k^{th} iteration of Step 1.

Proposition 2.1. *Algorithm 2.2 finds the optimal policy for the SOTA problem in a single iteration.*

Proof. Proof by induction over the sub-steps $\tau = \delta$ to $L\delta$.

Base case: When $k = 1$ the convolution product is computed on the interval $(0, \delta]$. From the definition of δ, we know that there does not exist a realizable travel-time that is less than or equal to δ. Therefore, $p_{ij}(\omega) = 0$ on the interval $(0, \delta]$ of the convolution product, and $u_i^1(t) = u_i^0(t)\ \forall i \in N$. This is indeed the correct solution $\forall\, t$ such that $0 \leq t \leq \delta$, since no feasible path to the destination exists in this time interval.

Algorithm 2.2 Single iteration SOTA algorithm

Step 0. Initialization.
$k = 0$
$u_i^k(t) = 0, \ \forall i \in N, \ i \neq s, \ t \in [0, T)$
$u_s^k(t) = 1, \ \forall\, t \in [0, T)$

Step 1. Update
For $k = 1, 2, \ldots, L$
  $\tau^k = k\delta$
  $u_s^k(t) = 1, \ \forall\, t \in [0, T)$
  $u_i^k(t) = u_i^{k-1}(t)$
    $\forall i \in N, \ i \neq s, \ t \in [0, \tau^k - \delta]$
  $u_i^k(t) = \max_j \int_0^t p_{ij}(\omega)\, u_j^{k-1}(t - \omega)\, d\omega$ % computation of the convolution product
    $\forall i \in N, \ i \neq s, \ (i, j) \in A, \ t \in (\tau^k - \delta, \tau^k]$

Induction step: Assume that the algorithm generates an optimal policy for $k < L$. We show that the algorithm also provides an optimal policy at step $k + 1$. When $t = (k + 1)\delta$, the convolution product is computed on the interval $(k\delta, (k + 1)\delta]$. Since $p_{ij}(\omega) = 0, \ \forall\, \omega$ such that $\omega \leq \delta$, to find the optimal policy for $u_i^{k+1}(t), \ \forall\, t$ such that $k\delta < t \leq (k + 1)\delta$, we only need to know the optimal policy for $u_j^k(t), \ \forall\, t$ such that $0 \leq t \leq k\delta$, $(i, j) \in A$. By the induction hypothesis we know that $u_i^k(t)$ gives the optimal policy $\forall\, t$ such that $0 \leq t \leq k\delta$ for all nodes and it is known. This implies that the optimal policy is computed for the range $0 \leq t \leq (k + 1)\delta$ at the end of the $(k + 1)^{th}$ step. □

The most compute-intensive step of this algorithm is the computation of the convolution product, which is represented algebraically in the above algorithm. If the link travel-time distributions $p_{ij}(\cdot)$, $(i, j) \in A$, and the cumulative optimal travel-time distributions $u_i(\cdot)$, $i \in N$, belong to a parametric distribution family which is closed under convolution (e.g. Gaussian, Erlang, see [17, 5]), the convolution product can be computed analytically. However, since $u_i(\cdot)$ is the point wise maximum of the convolution products of the link travel-time distributions $p_{ij}(\cdot)$ and the cumulative distributions $u_j(\cdot)$, $(i, j) \in A$, it does not have an analytical expression in general.

Numerical approximations of the distributions involved in the convolution product have been proposed in the literature. Fan et.al. [48] argue that since $u_i(\cdot)$ is a continuous monotone increasing function, it can be approximated by a low degree polynomial. When the approximating polynomial is of degree $2n$, the convolution integral can be solved exactly with n evaluation points using the Gaussian quadrature method (see [13]). However, the applicability of these methods is highly dependent on the shape of the travel-time distributions, which can be very complex, depending on the traffic conditions and the topology of the net-

work. Therefore, we propose solving the convolution product via a time discretization of the distributions involved, which results in a computational complexity that is independent of the shape of the optimal cumulative travel-time distributions $u_i(\cdot)$. An incremental update scheme that exploits the structure of the proposed SOTA algorithm is used to efficiently compute the discrete convolution product. This solution is shown to be computationally less expensive than than the existing convolution methods for the SOTA problem (e.g [103]). A detailed explanation is given in Sections 2.4 and 2.6.

2.3.2 Extended algorithm for time-varying link travel-times

The solution algorithm proposed in the previous section makes the assumption that link travel-time distributions are static. However, in real transportation networks, it is clear that link travel-time distributions are time-varying. In this section, we present an extension to Algorithm 2.2 that accounts for time-varying distributions. A common approach when solving shortest path problems on graphs with time-dependent distributions is to consider the corresponding time expanded graph with static weights. See [36] for a discussion on the various flavors of this problem. If the *first-in-first-out* (FIFO) condition holds, the problem can be solved without time-expanding the graph using a trivially modified version of Dijkstra's algorithm and indexing the link weights by time [45]. We propose a similar algorithm based on the fact that waiting at a node is never optimal when the FIFO conditions holds. First we show that most commonly used travel-time estimates satisfy the the FIFO condition and then show that waiting at a node is never optimal in such a model.

Definition 2.2. *In a deterministic setting, let α_P^t denote the travel-time on path[2] P when departing at time t. The graph satisfies the FIFO condition if and only if:*

$$\alpha_P^{t_1} \leq \alpha_P^{t_2} + (t_2 - t_1) \;\; \forall P \in \mathcal{P} \;\; \textit{and} \;\; \forall\, t_1, t_2 \;\; \textit{such that} \;\; 0 \leq t_1 \leq t_2 \tag{2.2}$$

This definition states that on a given path P, the travel-time $\alpha_P^{t_1}$ when leaving at t_1 is lower than the travel-time $\alpha_P^{t_2} + (t_2 - t_1)$ obtained by waiting at the departure node for $t_2 - t_1$ and departing at t_2.

In the time-varying setting, the link travel-time estimates given by a travel-time model can change as a vehicle moves through a link. We assume an elastic vehicle travel-time model, where the vehicle link travel-time is calculated based on all the link travel-time estimates the vehicle might encounter as it moves through a link. This is in contrast to a frozen vehicle travel-time model, where the travel-time is calculated simply based on the link travel-time estimate when the vehicle enters the link[3].

[2]The FIFO condition is typically defined on a link. Here, we use a path-based definition to make the subsequent explanations and proofs more intuitive. This leads to equivalent results under the assumption that the network topology is static.

[3]Please see [106] for further discussion on the elastic and frozen travel-time models.

Proposition 2.2. *Under an elastic vehicle travel-time model, a deterministic discrete-time traffic estimate such that the link travel-time is single-valued on each time-discretization yields a deterministic FIFO path.*

Proof. Consider two vehicles traveling along the same path P from node i to node k departing at times t_1 and t_2 respectively for $0 \leq t_1 \leq t_2$. Their respective travel-times are denoted $\alpha_P^{t_1}$ and $\alpha_P^{t_2}$. For the vehicle departing node i at time t_2 to arrive at node k before the vehicle departing at time t_1, it must overtake the vehicle that departed first. Overtaking can only occur when both vehicles are in the same space-time cell. However, since we assume that the model gives a single-valued speed in each space-time cell, both vehicles will travel at the same speed when in the same cell and no overtaking can occur. Therefore, a single-valued speed in each space-time cell implies that the vehicle that departed first will always arrive first. □

This guarantees that the shortest path problem on transportation networks, with time-varying link travel-times generated by a traffic information system, can be solved with the same complexity as in the static case by time-indexing the link weights [36]. In the case of transportation networks with stochastic link travel-times and the SOTA problem (equation (2.1)), a similar stochastic FIFO condition is needed to guarantee correctness of the algorithm with time-indexed link travel-times.

Definition 2.3. *Let $u_P^t(\cdot)$ denote the cumulative travel-time distribution on path P when departing at time t. The graph satisfies the stochastic FIFO condition if and only if:*

$$u_P^{t_1}(T) \geq u_P^{t_2}(T - (t_2 - t_1)) \;\; \forall P \in \mathcal{P}, 0 \leq t_1 \leq t_2, \;\; t_2 - t_1 \leq T \tag{2.3}$$

This definition states that at any given time and on any given path on the network, departing as soon as possible yields a greater probability of arriving on time than delaying the departure. The stochastic FIFO property is obtained if the conditions defined by Proposition 2.3 are satisfied.

Proposition 2.3. *A stochastic discrete-time traffic estimate such that link travel-time distributions are fixed for each time discretization yields a stochastic FIFO path.*

Proof. Proof by induction over the length of the path $(v_n, \ldots, v_1)$.

Base case ($n = 2$): From definition 2.3, a stochastic FIFO network satisfies the following condition:

$$u_{v_2 v_1}^{t_1}(T) \geq u_{v_2 v_1}^{t_2}(T - (t_2 - t_1)) \;\; \forall \;\; 0 \leq t_1 \leq t_2$$

where $u_{ij}^t(T)$ is the probability of arriving at node j in time T when departing from node i at time t. We want to show that a delayed departure cannot improve the probability of on time arrival when traveling from node v_2 to node v_1 on link (v_2, v_1). Without loss of generality,

let vehicle w_1 depart from node v_2 at time t_1 and vehicle w_2 depart from node v_2 at time t_2, where w_1 is in cell $c_1 = (v_2, v_1) \times [t_{c_1}, t_{c_2}]$ and w_2 is in cell $c_2 = (v_2, v_1) \times [t_{c_2}, t_{c_3}]$. Also, let $X_w(t_a, t_b)$ be the distribution of the distance traveled by vehicle w in the interval (t_a, t_b). We have the following:

$$X_{w_2}(t_1, t_2) = 0$$
$$X_{w_1}(t_1, t_2) \geq 0$$

Let x be the length of link (v_2, v_1). We want to show that:

$$\begin{aligned} & u^{t_1}_{v_2 v_1}(T) \geq u^{t_2}_{v_2 v_1}(T - (t_2 - t_1)) \\ \iff \quad & P(X_{w_1}(t_1, t_1 + T) \geq x) \geq P(X_{w_2}(t_2, t_1 + T) \geq x) \\ \iff \quad & P(X_{w_1}(t_1, t_{c_2}) + X_{w_1}(t_{c_2}, t_2) + X_{w_1}(t_2, t_1 + T) \geq x) \geq P(X_{w_2}(t_2, t_1 + T)) \geq x) \end{aligned}$$

This clearly holds because:

$$X_{w_1}(t_2, t_1 + T) = X_{w_2}(t_2, t_1 + T)$$

since both vehicles are in the same space-time cells from time t_2.

Induction step ($n = k$): We assume that a single-valued travel-time distribution for each space-time cell implies a stochastic FIFO path $(v_k, \ldots, v_1)$ of k nodes and show that it holds for $k+1$ nodes. The explanation in the base case shows that departing from node v_{k+1} earlier gives a higher probability of reaching node v_k within a given time budget. The induction hypothesis implies that arriving at node v_k earlier increases the probability of reaching the destination v_1 within a given time budget. Therefore, leaving the node v_{k+1} earlier increases the probability of reaching the destination within a given time budget. □

Under the stochastic FIFO condition, we now show that an optimal policy for the SOTA problem does not prescribe waiting at a node. Therefore, the SOTA problem on transportation networks with time-varying link travel-time distributions can be solved by time-indexing the link travel-time distributions.

Proposition 2.4. *In a stochastic FIFO network, waiting at a non-terminal node is never optimal when following the optimal SOTA policy.*

Proof. Proposition 2.3 shows that waiting at a node cannot improve the probability of arriving within a certain budget using the same path. We now show that waiting at a node cannot improve the probability of arriving within a given budget T when using the optimal path for each departure time. Assume that path P_1 is the optimal path when departing at time t_1 and that path P_2 is the optimal path when departing at time t_2. From proposition 2.3 we know that $u^{t_1}_{P_2}(T) \geq u^{t_2}_{P_2}(T - (t_2 - t_1))$. Furthermore, since P_1 is the optimal path when leaving at time t_1 with a budget of T, we have $u^{t_1}_{P_1}(T) \geq u^{t_1}_{P_2}(T)$. Therefore, $u^{t_1}_{P_1}(T) \geq u^{t_2}_{P_2}(T - (t_2 - t_1))$ and waiting cannot improve the probability of on time arrival. □

When the assumptions from Proposition 2.3 are satisfied, according to Proposition 2.4, waiting at a node is not part of the optimal policy. Therefore, the optimal policy in the time-varying case can be defined as follows:

$$\begin{aligned} u_i^\tau(t) &= \max_j \int_0^t p_{ij}^\tau(\omega)\, u_j^{\tau+\omega}(t-\omega)\, d\omega \\ &\quad \forall i \in N,\ i \neq s,\ (i,j) \in A,\ 0 \leq t \leq T,\ 0 \leq \tau \\ u_s^\tau(t) &= 1 \quad \forall\ 0 \leq t \leq T, 0 \leq \tau. \end{aligned} \tag{2.4}$$

where $u_i^\tau(t)$ is the maximum probability of arriving at destination s within time budget t when leaving node i at time τ. Waiting is not allowed in the optimal policy as enforced by the fact that the departure time from node i (superscript of $u_i^\tau(\cdot)$) is the same as the time at which the link (i,j) is traversed (superscript of $p_{ij}^\tau(\cdot)$). This policy is optimal according to Proposition 2.4.

The proposed algorithm uses the same network as the static SOTA problem and simply replaces the travel-time distribution query with a time-indexed version, as defined in Equation 2.4. i.e. it modifies the convolution step to query the appropriate link travel-time distribution based on the current time offset τ. This algorithm has the same computational complexity as the static algorithm because the structure of the graph remains the same and no additional queries are performed. The required memory is larger in the time-varying case than in the static case because there are multiple travel-time distributions for each link.

2.3.3 Generalized algorithm for correlated link travel-times

In this section, we relax another assumption that was made in Section 2.2, that the link travel-times on the network are independent. However, in reality the travel-times of neighboring links can be quite strongly correlated. Assuming that link travel-times across links satisfy the Markov condition, they only depend on their upstream and downstream neighbors. Therefore, the travel-time on each link is a joint distribution over the link and its neighbors. If we do not have any information regarding the travel-times on some of these links, the correct approach is to marginalize them out and use the marginal distribution of the link we are considering. However, in the SOTA formulation each decision could be preceded by conditioning on the travel-time of an upstream link, since the travel-time on the upstream link is known. Assuming independence in this case results in an inaccurate expected travel-time and an overestimation of the variance. Therefore, to minimize such errors one must incorporate this observation and use the conditional probability distribution of the link travel-time. We present a simple extension to our formulation that considers the correlation between a link and the upstream neighbors via which the link is reached. The proposed problem formulation is as follows:

$$
\begin{aligned}
u_i(t,k,y) &= \max_j \int_0^t p_{ij}(tt_{ij} = \omega | tt_{ki} = y)\, u_j(t-\omega, i, \omega)\, d\omega \\
&\quad \forall i \in N,\ i \neq s,\ (i,j) \in A,\ (k,i) \in A, \\
&\quad 0 \leq y \leq (T-t),\ 0 \leq t \leq T \\
u_s(t,k,y) &= 1 \quad \forall\ 0 \leq t \leq T,\ (k,s) \in A,\ 0 \leq y \leq T-t
\end{aligned} \tag{2.5}
$$

where $u_i(t,k,y)$ is the cumulative distribution function (CDF) when t is the remaining time budget, k is the node from which the vehicle is arriving and y is the realized travel-time on this upstream link, and $p_{ij}(tt_{ij} = \omega | tt_{ki} = y)$ is the probability that the travel-time on link (i,j) is ω conditioned on the travel-time on link (k,i) being y. The joint probability density function of a link and its upstream neighbors is assumed to be known. In this case, each CDF $u_i(\cdot,\cdot,\cdot)$ and travel-time distribution $p_{ij}(\cdot)$ is now conditioned on the upstream travel-time, but the structure of the problem remains unchanged. We are simply propagating more information at each step. Therefore, the correctness analysis of our algorithm remains unchanged and the same proof holds. However, the time complexity of the algorithm increases since a new dimension is being added to the problem. Equation (2.5) needs to be solved for each incoming node k and the travel-time on link (k,i), which is in the range $0 \leq y \leq (T-t)$. This increases the runtime of the original formulation (Equation (2.1)) by a factor of ΦT, where Φ is the maximum in-degree of the network.

In most practical cases this is likely to make the algorithm intractable for real-time applications, as the complexity is now cubic in T. Therefore, we propose using a discrete approximation of the conditional distribution function. In the simplest case, the conditioning can be done based on whether the upstream link was in congestion or free flow, which will only increase the complexity by a factor of 2Φ. If upstream travel-time is discretized in to d ranges, the increase in complexity will be a factor of $d\Phi$. The most appropriate value to use for d depends on the quality of the conditional travel-time distributions available and the computing resources that can be utilized.

2.4 Discrete formulation of the SOTA algorithm with a Fast Fourier Transform solution

For practical networks, a numerical approximation of the convolution integral is necessary, with a proper discretization. In the discrete setting, Algorithm 2.2 can be reformulated as shown in Algorithm 2.3 below. The length of a discretization interval is given by Δt and T is the time budget. The functions $u_i(\cdot)$ and $p_{ij}(\cdot)$ are vectors of length $L = \lceil \frac{T}{\Delta t} \rceil$. For notational simplicity, we assume that T is a multiple of Δt. In general, the link travel-time distributions are available as either discrete or continuous time distributions. If the link travel-time distribution is discrete and the length of the discretization interval d is not equal to Δt, the probability mass needs to be redistributed to intervals of Δt. If the distribution

is continuous, the probability mass function $p_{ij}(\cdot)$ is computed as follows:

$$p_{ij}(h + \Delta t) = \int_{h}^{h+\Delta t} p_{ij}(\omega)\ d\omega,\ \ \forall h = 0, \Delta t, \ldots, (L-1)\Delta t \tag{2.6}$$

Algorithm 2.3 Discrete SOTA algorithm

Step 0. Initialization.
$k = 0$
$u_i^k(x) = 0,\ \ \forall i \in N,\ \ i \neq s,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$
$u_s^k(x) = 1,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$

Step 1. Update
For $k = 1, 2, \ldots, L$
  $\tau^k = k\delta$
  $u_s^k(x) = 1,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{T}{\Delta t}$
  $u_i^k(x) = u_i^{k-1}(x)$
    $\forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ x \in \mathbb{N},\ \ 0 \leq x \leq \frac{(\tau^k - \delta)}{\Delta t}$
  $u_i^k(x) = \max_j \sum_{h=0}^{x} p_{ij}(h)\, u_j^{k-1}(x-h)$
    $\forall i \in N,\ \ i \neq s,\ \ (i,j) \in A,\ \ x \in \mathbb{N},\ \ \frac{(\tau^k - \delta)}{\Delta t} + 1 < x \leq \frac{\tau^k}{\Delta t}$
    % Δt is selected such that $\delta > \Delta t$

2.4.1 Complexity analysis

Obtaining the appropriately discretized probability mass functions can be done in time $O(\frac{mT}{\Delta t})$, since there are m links and each link travel-time distribution function is of length $\frac{T}{\Delta t}$. This can also be computed in advance and reused during each call to the algorithm[4]. In step 0, initializing k vectors (one for each node i) of length $\frac{T}{\Delta t}$ takes $O(\frac{kT}{\Delta t})$ time. In step 1, notice that for each link (i,j) the algorithm progressively computes a sum of increasing length from $x = 1$ to $x = \frac{L\delta}{d} = \frac{T}{\Delta t}$. Therefore, the time complexity of the summation for each link is $O((\frac{T}{\Delta t})^2)$. The assignment $u_i^k(x) = u_i^{k-1}(x)$ can be done in constant time by manipulating pointers instead of a memory copy or by simply having one array for all $u_i(\cdot)$ that keeps getting updated at each iteration of the loop. Since there are m links, the total time complexity of step 1 is $O(m(\frac{T}{\Delta t})^2)$. This dominates the complexity of step 0 and therefore is the total time complexity of the entire algorithm. Recall that this is identical to the time complexity of the discrete approximation algorithm proposed in [103].

[4]In the case of time-varying link travel-times, this needs to be recomputed for each time step at which the travel-times differ.

Algorithm 2.3 can perform more efficiently by computing the convolution products via the *Fast Fourier Transform* (FFT). The FFT computes the convolution of two vectors of length n in $O(n\log(n))$ time [29]. Notice however that the proposed algorithm does not compute the entire convolution at once. The computation is required to be done in blocks of length δ to preserve optimality. Therefore, L convolution products of increasing length $\delta, 2\delta, \ldots, L\delta$ have to be computed. One inefficiency of this approach is that successive convolutions recompute the results that have already been obtained. For each link, the time complexity of the sequence of FFTs is $O(\sum_{k=1}^{L} \frac{\delta k}{\Delta t} \log(\frac{\delta k}{\Delta t}))$, where $L = \lceil \frac{T}{\delta} \rceil$. Since there are m links, the total time complexity is:

$$O\left(m \sum_{k=1}^{\lceil \frac{T}{\delta} \rceil} \frac{\delta k}{\Delta t} \log\left(\frac{\delta k}{\Delta t}\right)\right) \tag{2.7}$$

As $T \to \infty$, the complexity of the FFT based approach $O((\frac{T}{\Delta t}^2 \log(\frac{T}{\Delta t}))$ is asymptotically larger than the run-time of the brute force approach $\sum_{k=1}^{\frac{T}{\Delta t}} k = O((\frac{T}{\Delta t})^2)$. However, the running time of the FFT approach is significantly smaller than the brute force approach when the time budget increases and the discretization time step decreases, as shown in Proposition 2.5.

Proposition 2.5. *The travel budget* $t = \Delta t \cdot 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - \delta$ *is a lower bound for the largest budget at which the FFT based approach has a faster run-time than the brute force approach.*

Proof. We compare the runtime of the FFT approach and the brute force approach. In order to compute the optimal solution up to a given time τ, the brute force method needs to be executed for $\frac{\tau}{\Delta t}$ steps and the FFT method needs to be executed for $\frac{\tau}{\delta}$ steps. The runtime of the brute force method is $\sum_{k=1}^{\frac{\tau}{\Delta t}} k$ and the running time of the FFT approach is $\sum_{k=1}^{\frac{\tau}{\delta}} \frac{\delta k}{\Delta t} \log(\frac{\delta k}{\Delta t})$. The FFT approach is faster than the brute force convolution for $t = K\,\delta$ such that:

$$\sum_{k=1}^{K} \frac{\delta k}{\Delta t} \log \frac{\delta k}{\Delta t} \leq \sum_{k=1}^{\frac{K\delta}{\Delta t}} k = \frac{1}{2} \frac{K\delta}{\Delta t} \left(\frac{K\delta}{\Delta t} + 1\right).$$

If we use the fact that the function $x \mapsto x \log x$ is increasing on $[1; +\infty)$, we can bound the left hand side of the above inequality as follows:

$$\begin{aligned} \sum_{k=1}^{K} \frac{\delta k}{\Delta t} \log \frac{\delta k}{\Delta t} &< \int_{z=0}^{K} \frac{\delta(z+1)}{\Delta t} \log \frac{\delta(z+1)}{\Delta t}\, dz \\ &= \frac{1}{2} \frac{\delta}{\Delta t} (K+1)^2 \left(\log \frac{\delta\,(K+1)}{\Delta t} - \frac{1}{2}\right) - \frac{1}{2} \frac{\delta}{\Delta t} \left(\log \frac{\delta}{\Delta t} - \frac{1}{2}\right) \end{aligned}$$

where the inequality is a right Riemann integral bound. A lower bound t on the time up to which the FFT approach is faster than the brute force convolution thus satisfies:

$$\frac{1}{2}\frac{\delta}{\Delta t}(K+1)^2\left(\log\frac{\delta(K+1)}{\Delta t}-\frac{1}{2}\right)-\frac{1}{2}\frac{\delta}{\Delta t}\left(\log\frac{\delta}{\Delta t}-\frac{1}{2}\right)\leq\frac{1}{2}\frac{K\delta}{\Delta t}\left(\frac{K\delta}{\Delta t}+1\right).$$

If we assume $\delta \geq \Delta t \exp\frac{1}{2}$, we have $-\frac{1}{2}\frac{\delta}{\Delta t}\left(\log\frac{\delta}{\Delta t}-\frac{1}{2}\right)\leq 0$ and thus a sufficient condition to have the inequality above satisfied is to have:

$$\frac{1}{2}\frac{\delta}{\Delta t}(K+1)^2\left(\log\frac{\delta(K+1)}{\Delta t}-\frac{1}{2}\right)\leq\frac{1}{2}\frac{K\delta}{\Delta t}\left(\frac{K\delta}{\Delta t}+1\right)$$

We can equivalently rewrite the above inequality as:

$$(K+1)^2\log\frac{\delta(K+1)}{\Delta t}\leq\frac{1}{2}(K+1)^2+\frac{\delta}{\Delta t}K^2+K.$$

We wish to find $\alpha\in\mathbb{R}$ such that $\alpha(K+1)^2\leq\frac{1}{2}(K+1)^2+\frac{\delta}{\Delta t}K^2+K$ for $K\geq 1$. This is satisfied for $\alpha\leq\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)$, which allows us to write that the FFT is faster than the brute force approach when:

$$(K+1)^2\log\frac{\delta(K+1)}{\Delta t}\leq\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)(K+1)^2$$

which is equivalent to:

$$K\leq\frac{\Delta t}{\delta}2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)}-1$$

and thus a lower bound t reads:

$$t\leq\Delta t\,2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)}-\delta$$

□

This lower bound t is typically a large value in road networks where individual links have large travel-times. Table 2.1 shows the value of the lower bound for some sample values of δ and Δt. The exact value of t can be obtained by computing summation (2.7) numerically.

2.4.2 Acceleration of Algorithm 2.3 with localization

As shown in Section 2.4.1, the runtime of the FFT based solution is a function of δ and decreases as the value of δ increases. The value of δ that is used in the algorithm is bounded by the minimum realizable travel-time across the entire network. However, in general, road networks are heterogeneous and contain a large range of minimum realizable travel-times. This section presents an optimization that can significantly improve the runtime of the proposed algorithm by exploiting the disparity of these local δ values.

	$\Delta t = 0.1$	$\Delta t = 0.2$	$\Delta t = 0.5$	$\Delta t = 1$
$\delta = 90$	$1.51 \cdot 10^{65}$	$4.11 \cdot 10^{31}$	$4.93 \cdot 10^{11}$	$1.6 \cdot 10^{5}$
$\delta = 60$	$4.00 \cdot 10^{42}$	$2.11 \cdot 10^{20}$	$1.50 \cdot 10^{7}$	$9.17 \cdot 10^{2}$
$\delta = 30$	$1.06 \cdot 10^{20}$	$1.08 \cdot 10^{9}$	$4.59 \cdot 10^{2}$	4.57
$\delta = 15$	$5.44 \cdot 10^{8}$	$2.47 \cdot 10^{3}$	2.41	$1.27 \cdot 10^{-1}$

Table 2.1: Lower bound $t = \Delta t \cdot 2^{\frac{1}{4}\left(3+\frac{\delta}{\Delta t}\right)} - \delta$ (minutes) for which the FFT approach is faster than the brute force approach for the computation of the convolution product in the main algorithm. Values of δ and Δt are given in seconds.

Proposition 2.6. *Let β_{ij} be the minimum realizable travel-time on link (i, j) with $\delta_{ij} = \beta_{ij} - \epsilon$ $(0 < \epsilon < \beta_{ij})$ and τ_i be the budget up to which the cumulative distribution function $u_i(\cdot)$ has been computed for node i. For correctness, the invariant*

$$\tau_i \leq \min_j(\tau_j + \delta_{ij}) \ \ \forall (i, j) \in A \tag{2.8}$$

must be satisfied throughout the execution of the algorithm.

Proof. Assume that this invariant can be violated. Then, it is possible to compute the cumulative distribution function $u_i(\cdot)$ at some node i such that $\tau_i > \min_j(\tau_j + \delta_{ij})$, which in turn means that $\tau_i - \tau_j > \delta_{ij}$ for at least one node j. This implies that $\exists\ t'$ such that $u_i(t')$ was computed using the product of a downstream cumulative distribution function $u_j(t' - \omega)$ and $p_{ij}(\omega)$, where $u_j(t' - \omega)$ is unknown because $\tau_j < t' - \delta_{ij}$ and $p_{ij}(\omega) > 0$ because $\omega > \delta_{ij}$. This value of the cumulative distribution function $u_i(t')$ is undefined and the SOTA algorithm fails. Therefore, for correctness the invariant should not be violated. □

When computing the cumulative density function $u_i(\cdot)$ using local δ_{ij} values, the growth of τ_i is different across the nodes i, unlike in our original algorithm (Algorithm 2.3) where the τ_i grow at the constant uniform rate δ. Furthermore, when $u_i(\cdot)$ is updated asynchronously using the invariant $\tau_i \leq \min_j(\tau_j + \delta_{ij}),\ \ (i, j) \in A$, the order in which the nodes are updated impacts the runtime of the algorithm, as illustrated in Example 2.2.

Example 2.2. *To illustrate how the order in which the nodes are updated impacts the runtime of Algorithm 2.3, consider the network in Figure 2.2. The value of τ_i and the computation time depends on the order in which the nodes are considered. In the worst case, as a lower bound, we assume that $u_i(\cdot)$ is updated based on the values of its constraint nodes in the previous iteration. Table 2.2 shows the sequence of updates for four iterations when using the constraints τ_i from the previous iteration. Notice that the update pattern is cyclic every four iterations. The highest speedup is achieved when the nodes in the loop are considered in topological order. Table 2.3 shows the sequence of updates when the nodes are considered in the topological order (a, b, c, d). As seen in Table 2.3, the τ_i value for each node i can be incremented by the length of the shortest loop node i belongs to when the nodes are updated in*

this order. The topological order can be determined easily in this simple example, but such an ordering is unlikely to exist in realistic transportation networks. Table 2.4 shows the sequence of updates when the nodes are considered in the order (d, c, b, a). It can be clearly seen that this ordering is much more inefficient than the ordering (a, b, c, d). Furthermore, without the local δ optimization, the algorithm can only update u_i by one step at each iteration, since the minimum δ_i value is 1 in this example. This simple example shows how local δ optimization can provide large improvements in the runtime.

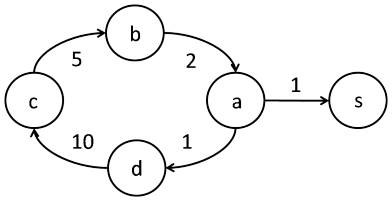


Figure 2.2: The influence of the order of computation with an example of a simple subnetwork with a loop. The δ value for each link is given along the link.

Iter.	a	b	c	d
1	1	2	5	10
2	11	3	7	15
3	16	13	8	17
4	18	18	18	18

Table 2.2: τ_i values when computing u_i constrained on previous iteration.

Iter.	a	b	c	d
1	1	3	8	18
2	19	21	26	36
3	37	39	44	54
4	55	57	62	72

Table 2.3: τ_i values when computing u_i in the order (a, b, c, d).

Iter.	d	c	b	a
1	10	5	2	11
2	15	7	13	16
3	17	18	18	18
4	28	23	20	29

Table 2.4: τ_i values when computing u_i in the order (d, c, b, a).

Table 2.5: The influence of the update order.

Given that the runtime of the SOTA algorithm depends on the update order, we would like to find an optimal ordering that minimizes the runtime of the algorithm. The first step in finding such an optimal ordering is to formalize the runtime of the FFT SOTA algorithm.

Definition 2.4. *The computation time of the cumulative density function $u_i(\cdot)$ can be minimized by finding the ordering that solves the following optimization problem.*

$$\begin{aligned}
&\text{minimize}_{(\tau_i^{k_i}, K_i)} && \sum_{(i,j)\in A}\sum_{k_i=1}^{K_i} \frac{\tau_i^{k_i}}{\Delta t}\log\frac{\tau_i^{k_i}}{\Delta t} && && (2.9)\\
&\text{ject to} && \tau_i^{k_i} \le \tau_j^{k_j} + \delta_{ij} && \forall \tau_i^{k_i}, \tau_j^{k_j} \ \ s.t.\ \ (i,j)\in A,\\
& && && C(i,k_i) < C(j,k_j+1)\\
& && \tau_r^{K_r} \ge T &&\\
& && \tau_s^{1} \ge T &&\\
& && \tau_i^{1} \ge \Delta t && \forall i \in N, i \ne s\\
& && \tau_i^{k+1} > \tau_i^{k} && \forall i \in N
\end{aligned}$$

where $\tau_i^{k_i}$ is the budget up to which $u_i(\cdot)$ has been computed in the k_i^{th} iteration of computing $u_i(\cdot)$, $C(\cdot,\cdot)$ is an index on the order in which nodes are updated such that $C(i,k_i)$ denotes when node i is updated for the k_i^{th} time and K_i is the total number of iterations required for node i.

The optimal order in which $u_i(\cdot)$ is computed might result in updating some set of nodes multiple times before updating another set of nodes.

Proposition 2.7. *The ordering that gives the optimal solution to the optimization problem (2.9) can be obtained using Algorithm 2.4 in $O(\frac{mT}{\Delta t}\log(n))$ time, where n and m are respectively the number of nodes and links in the network, Δt is the time discretization interval and T is the time budget. See*

The optimal order of updates (node and value) that computes the cumulative distribution function $u_r(T)$ of the origin r most efficiently is stored in the stack χ at the termination of the algorithm. Algorithm 2.4 works by taking the source node r and the time budget to which it needs to be updated T, and then recursively updating the set of constraints that need to be satisfied before $u_r(T)$ can be computed. At the first iteration, the source and its terminal value in the algorithm (the budget) are added to the stack, and the constraints that are required for updating the source to that value are stored in the heap ψ. At any given iteration, the largest value in the heap is extracted and added to the stack, since it is the most constrained node in the current working set. A detailed proof is given below.

Proof. Algorithm 2.4 begins at the termination condition of the SOTA problem, the source node being updated to the budget, and recursively builds (in reverse order) the optimal sequence of updates that allow the source node to be updated to the budget. Thus, the algorithm is initialized with the terminal condition of $\tau_r = T$. This is the initial constraint of the optimal ordering algorithm. Reaching this condition must be preceded by all the

Algorithm 2.4 Optimal order for updating $u(\cdot)$

Step 0. Initialization.
$\tau(i) = 0, \ \forall i \in N, \ i \neq r, \ i \neq s$ % where r is the origin and s is the destination
$\tau(s) = \infty, \ \tau(r) = T$
$\psi := \{(r, \tau(r))\}$ % ψ is a priority queue data structure
$\chi := \{(r, \tau(r))\}$ % χ is a stack data structure

Step 1. Update
$(v, \theta) =$ExtractMax(ψ)
$\tau(v) := \theta$
Push$(\chi, (v, \tau(v)))$
$\pi :=$ Children(v)
For $k := 1$ to size(π)
  If $((\pi[k] \neq s)$ and $(\tau(v) - \delta_{v\pi[k]}) > 0)$
    $\tau :=$ max(Extract$(\psi, \pi[k]), \tau(v) - \delta_{v\pi[k]})$
    Insert$(\psi, (\pi[k], \tau))$

Step 2. Termination
If $\psi := \emptyset$ stop;
Otherwise go to Step 1.

downstream nodes $j \in \pi_r$ of the source node r being updated to at least $\tau_r - \delta_{rj}$, since the correctness of the algorithm requires the invariant in equation (2.8),

$$\tau_i \leq \min_j (\tau_j + \delta_{ij}) \ \ \forall (i, j) \in A$$

to hold. Therefore, the initial constraint $\tau_r = T$ is relaxed by adding these new constraints to the constraint list ψ. At the same time, we also add $\tau_r = T$ to the optimal order stack χ. This will be the final update in the optimal ordering, since we are building the list from the last update to the first.

Once we have a set of new constraints, we need to decide which node to relax and how far to update τ_i. The contribution from a given node i to the objective function of the optimization problem (2.9) is minimized when the τ_i value for that node is minimized as much as possible. Furthermore, lowering the τ_i value for a node reduces the new constraints introduced when relaxing that node. Therefore, an optimal update should reduce the τ_i value of a node as much as possible such that invariant (2.8) is not violated.

The next step is to determine which constraint from the constraint list ψ to relax first. We need to show that that the order of relaxation guarantees that the algorithm will not introduce any new constraints that violate any updates done in previous relaxations. Picking the node i with the largest constraint τ_i in ψ guarantees this, since δ_{ij} is strictly positive

and the new constraints $\tau_j\,(\forall j \in \pi_i)$ that are added satisfy the condition $\tau_j \leq \tau_i - \delta_{ij}$, which implies that node i cannot have a new constraint that is greater than its current constraint τ_i at any future point of the algorithm. Therefore, correctness is preserved by relaxing the node i with the largest constraint τ_i in ψ and setting its value to $\tau_i - \delta_{ij}$. Node i is then added to the optimal order stack χ. The new constraints introduced by setting node i to this value are then added to ψ, if $\tau_i - \delta_{ij} > 0$ and $j \neq s$. It is unnecessary to add constraints if these conditions are not satisfied, since $u_i(t) = 0\,(\forall\, i \in N, t \leq 0)$ and $u_s(t) = 1\,(\forall\, t \geq 0)$. This process is performed recursively until the list ψ is empty. The process is guaranteed to terminate because the values of new constraints that are added when relaxing an existing constraint are monotonically decreasing.

The complexity of Algorithm 2.4 is $O(\frac{mT}{\Delta t}\log(n))$. The *Extract* and *Insert* operations of the Algorithm 2.4 can be replaced with a single *IncreaseKey* operation, which runs in $O(log(n))$ time (see [125, 29] for details). The *IncreaseKey* operation will increase the key of a given node if the new key is greater than its existing value. This is exactly what the *Extract* and *Insert* operations are used for. The pseudo-code for Algorithm 2.4 uses the *Extract* and *Insert* operations to improve readability. The *ExtractMax* operation can be performed in constant time. Therefore, each iteration of the algorithm takes $O(log(n))$ time. In the worst case, each link might need to be updated $\frac{T}{\Delta t}$ times. Repeating this over the m links of the network, we obtain a complexity of $O(\frac{mT}{\Delta t}\log(n))$. □

2.4.3 Search space pruning by elimination of infeasible paths

The proposed algorithm requires computing the cumulative distribution function $u_i(t)$ for every node in the graph. This can be prohibitively expensive even in reasonably sized road networks. Therefore, we need to constrain the search space of our algorithm. In this section, the proposed algorithm is extended by adding a pruning algorithm that eliminates infeasible paths by removing unnecessary nodes during a preprocessing step. Consider an instantiation of the SOTA problem with an origin node r, destination node s and a travel-time budget of T. Since every link (i, j) in the network has a minimum realizable travel-time β_{ij} (as defined in Section 2.3.1), it follows that every path in the network must have a minimum realizable travel-time as well. For any path P_{ik}, let the minimum realizable path travel-time be α_{ik}. The value of α_{ik} can be found by running a standard deterministic shortest path algorithm such as Dijkstra's algorithm on the network with the link weights being the minimum realizable link travel-time corresponding to each link.

Proposition 2.8. *Consider some arbitrary node i in the network. Let α_{ri} and α_{is} be respectively the minimum realizable travel-times from the origin to node i and from node i to the destination.*

1. *If $\alpha_{ri} + \alpha_{is} > T$, we can safely remove this node from the network and ignore it when solving the SOTA problem.*
2. *The cumulative distribution function $u_i(\cdot)$ only needs to be computed for the time interval $\alpha_{is} \leq t \leq T - \alpha_{ri}$.*

Proof.

1. If $\alpha_{ri} + \alpha_{is} > T$, the minimum realizable travel-time from the origin to the destination through node i is greater than the travel-time budget. Therefore, no feasible path exists through node i.

2. The minimum realizable travel-time from the origin to node i is α_{ri}. Therefore, no path in the dynamic programming recursion will query $u_i(t)$ for $t > T - \alpha_{ri}$. The minimum realizable travel-time from node i to the destination is α_{is}. Therefore, $u_i(t)$ is zero for $t < \alpha_{is}$.

□

By performing an all destinations shortest path computation from the source and an all sources shortest path computation from the destination, we can significantly prune both the size of the network required when solving the SOTA problem and the time interval for which the cumulative distribution function $u_i(\cdot)$ needs to be computed for each node. For a graph with n nodes and m links, the time complexity of Dijkstra's algorithm is $O(m + n \log n)$ [29], which is dominated by the complexity of the SOTA algorithm. Thus, the cost of pruning is negligible compared to the complexity of the SOTA algorithm. Furthermore, state of the art shortest path algorithms that run in near constant time [2, 55] can be utilized when the minimum realizable travel-time does not vary with time. It should be noted that this is a very conservative pruning algorithm and that further runtime reductions can be achieved using more aggressive pruning methods, which are described in the following chapter.

2.5 Implementation of the algorithm in the Mobile Millennium system

Experimental results are provided by implementing the proposed algorithms in the Mobile Millennium traffic information system. Mobile Millennium is a traffic information system that integrates traffic measurements from a variety of sensors (loop detectors, radars, probe vehicles, tolltag readers) to produce real-time traffic estimates for the San Francisco Bay Area. The system collects on the order of 2 million reports of vehicle counts and time occupancy from loop detectors, count and point speeds from over 100 radars, and between $2 - 10$ million probe vehicle speeds from traffic commuters daily for the San Francisco Bay Area. These data feeds are filtered and fused using multiple techniques (viability tools ([8]), statistical methods, etc.), and integrated as real-time observations by highway and arterial traffic estimation modules. The Mobile Millennium system provides real-time estimation of traffic conditions (travel-time, speed, density) on most of the non residential streets and roads in the Bay Area, on as ongoing basis. Traffic estimates are broadcast via a web interface and a cellphone application, enabling commuters to select the optimal commute route in order to avoid congestion, caused by both recurrent conditions and unexpected events such

as accidents. In this section, we propose to test the performance of the routing algorithm introduced in this chapter on two specific traffic estimates from the system:

- Sample-based representation of the velocity map on the Bay Area highway network (Figure 2.3): this output is produced by the *v-CTM EnKF* algorithm from [136] which fuses loop detectors counts, radars speed and spatially sampled probe speeds into a partial differential equation flow model coupled with an ensemble Kalman filter estimation algorithm. This estimate is updated every 30 seconds and has a space resolution of approximately 400 meters. Link travel-time distributions representing model uncertainty on the travel-time at the mean speed can be directly computed from this output and used by our routing algorithm.

- First two moments representation of link travel-times on the San Francisco arterial network (Figure 2.4): this output is produced by the machine learning algorithm from [66] using a mixture of real-time and historical probe generated travel-times. The link travel-time distributions represent individual commuters travel-time and can be directly used by our routing algorithm.

The practical applicability of the algorithms presented are illustrated by implementing them in the Mobile Millennium system and comparing them to existing solutions. The algorithms are tested on the arterial and highway road networks described above. The highway network from Figure 2.3 is a relatively sparse network with short loops at ramps and intersections, and large loops covering the entire network. This network contains 3639 nodes and 4164 links. The San Francisco arterial network of Figure 2.4 is a dense network with a large number of loops with varying lengths, with 1069 nodes and 2644 links.

The algorithms are implemented in Java and executed on a Windows 7 PC with a 2.67Ghz Dual Core Intel Itanium processor and 4GB of RAM. We use the open source Java libraries JTransforms ([135]) and SSJ ([79]) for FFT computations and manipulating probability distributions. Time-varying link travel-time distributions are obtained a-posteriori from the traffic estimation models described above. Both travel-time models assume that the link travel-times are independent with respect to each other.

We consider the following performance metrics:

- *Runtime*: computation time for the different variants of our algorithm compared to similar existing routing algorithms and specific runtime improvement provided by the speed up techniques introduced in this work.

- *On-time arrival guarantee*: sampling the Mobile Millennium traffic estimates enable the generation of realistic user travel-time realizations. The probability of arriving on time for both the SOTA policy and least expected travel-time path are compared. The performance of the SOTA algorithm under different traffic conditions and route types is also analyzed.

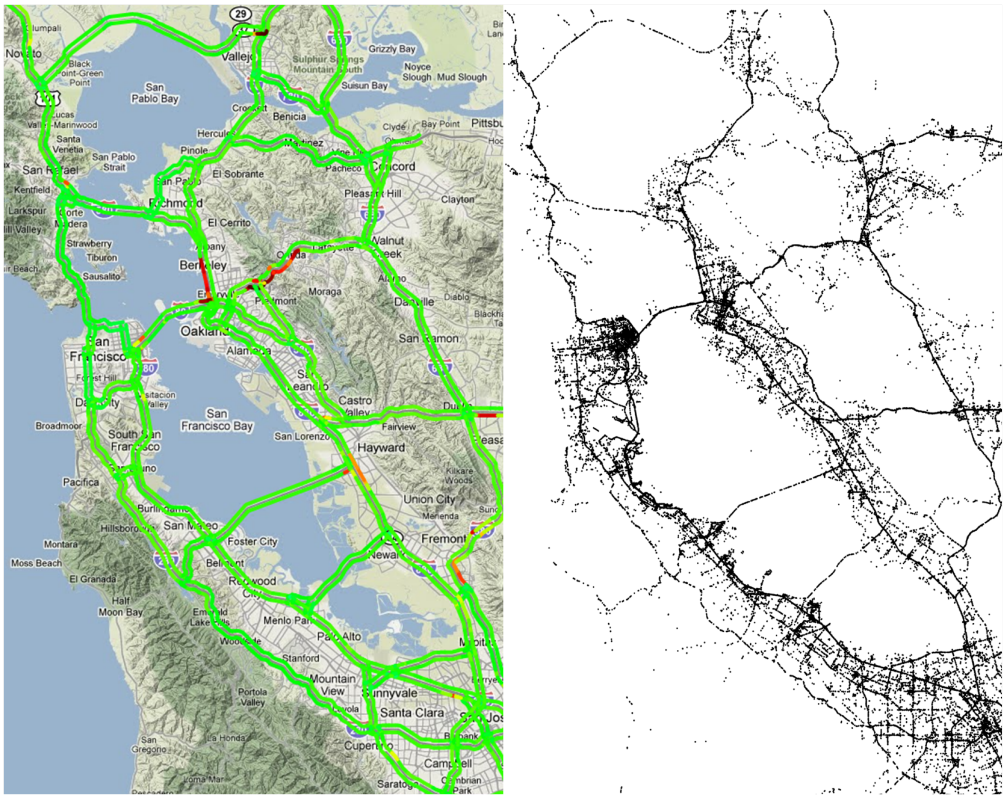


Figure 2.3: Best viewed in color. **Velocity estimates on the Bay Area highway network. Right:** Cumulated probe speed measurements collected on July 29$^{\text{th}}$ by the Mobile Millennium system. **Left:** Velocity estimates on July 29$^{\text{th}}$ at 9 pm; most of the network is in free flow, active bottlenecks correspond to red spots.

- *En route re-routing*: performance of the algorithm on a real test case on which the ability of the routing module to provide adaptive route choices depending on traffic conditions is presented.

2.5.1 Runtime performance

As shown in the complexity analysis from Section 2.4.1, the algorithm introduced in this chapter is linear in the size of the network and pseudo-polynomial in the ratio $T/\Delta t$, where T is the time budget of the user, and Δt is the discretization interval of time. It was argued that the proposed algorithm performs better than the existing solution to the SOTA problem, in theory, for most practical routing problems. In this section we present empirical results to validate this claim. Figure 2.5 shows the actual run-times (in CPU time) for two sample origin-destination (OD) pairs, when computing the optimal policy over a range of travel-time budgets.

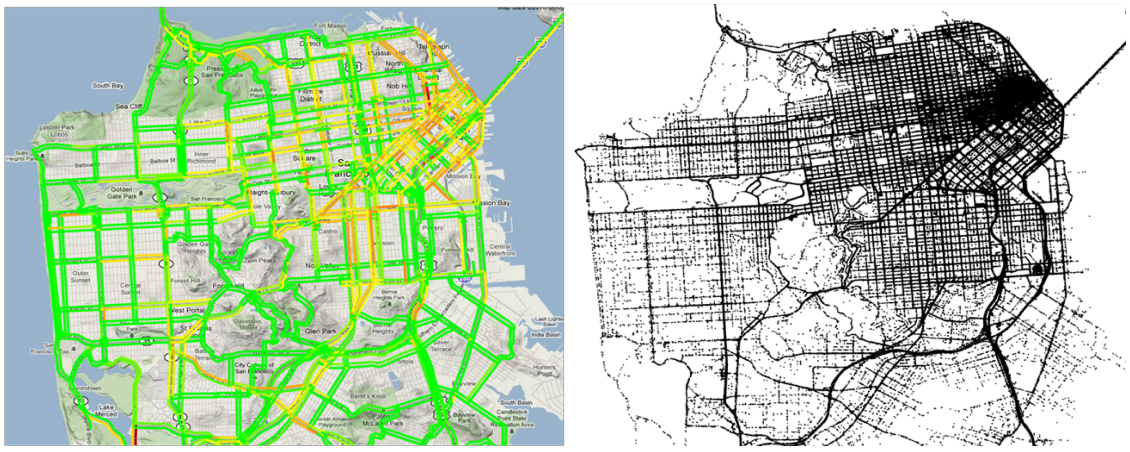


Figure 2.4: Best viewed in color. **Travel-time estimates on the San Francisco arterial network. Right:** Cumulated probe location measurements collected on July 29$^{\text{th}}$ by the Mobile Millennium system. **Left:** Travel-time estimates on July 29$^{\text{th}}$ at 8 pm; green links correspond to free flow travel-times and red spots correspond to congestion.

As illustrated in Figure 2.5, the FFT based algorithm with optimal ordering performs significantly better than the brute force approach[5] in both networks (17 minute gain for a 1 hour policy on the highway network and 10 minute gain for a 30 minute policy on the arterial network) and makes the SOTA problem more tractable for real-time applications. The proposed algorithm performs much better on the highway network, where most of the loops have large minimum travel-times and allow the cumulative distribution function $u_i(\cdot)$ to be updated in larger increments (as explained in Section 2.4.2), making the convolution product more efficient. The FFT algorithm with a random update order performs quite poorly, especially in the case of the arterial network, where it takes approximately 70 minutes to compute a 30 minute policy.[6] This observation agrees with the lower bound in Proposition 2.5 and the example values in Table 2.1, since the relative efficiency of the FFT algorithm increases exponentially with the travel-time of the minimum length loop for each node.

2.5.2 Comparison with classical routing algorithms

Most common routing algorithms rely solely on the knowledge of the travel-time as a deterministic quantity when generating optimal route choices. This deterministic travel-time can be inferred for instance from the speed limitations on the network, from historical

[5] [103] show that their discrete convolution algorithm dominates the method given in [47] in terms of runtime. We refer the reader to Table 3 in [103] for the comparison. Therefore, we compare our algorithm to the algorithm given in [103].

[6] The computation time for the FFT algorithm with a random update order is not displayed in Figure 2.5, since it takes much longer than the other algorithms, to improve the readability of the plot.

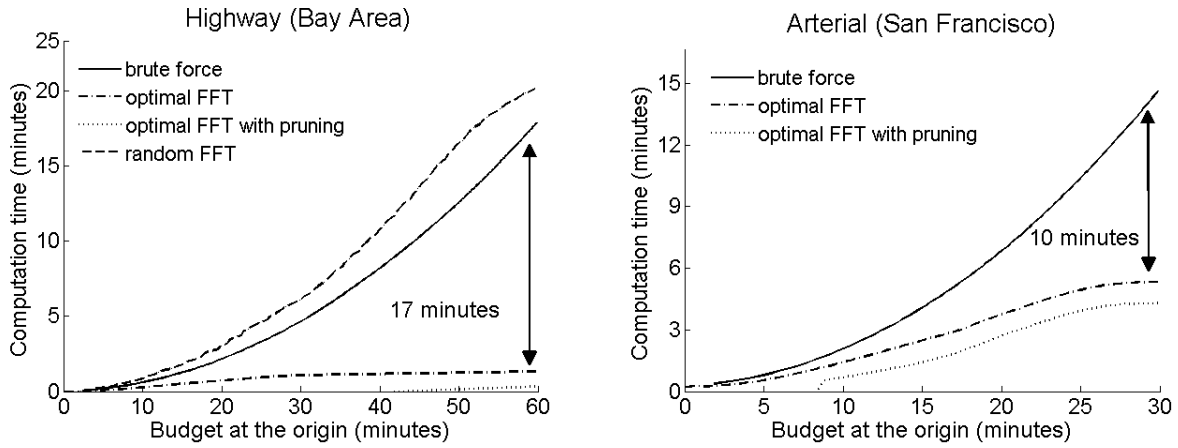


Figure 2.5: **Illustration of the tractability of the problem. Comparison of run-times (CPU time) for the brute force convolution (solid line), randomly ordered FFT (dashed line), optimally ordered FFT (dot-dash line) and optimally ordered FFT with pruning (dotted line): Left: Highway network** Runtime for computing the optimal policy from Berkeley to Palo Alto. The time discretization (Δt) is 0.5 seconds. **Right: Arterial network** Runtime for computing the optimal policy for a route from the Financial District (Columbus and Kearny) to the Golden Gate Park (Lincoln and 9th). The time discretization (Δt) is 0.2 seconds.

realized travel-times (e.g. average, worst case), or from a real-time deterministic output of a traffic information system (e.g. mean travel-time, median travel-time).

We compare the performance of the SOTA algorithm to these classical methods on both the highway and arterial networks defined above. First we instantiate the case of a commuter traveling on the highway network from Berkeley (latitude: 37.87201, longitude: -122.3056) to Palo Alto (latitude: 37.4436, longitude: -122.1176), on July 29^{th}, on two departure times, 6:45 am and 8:00 am. This is a typical Bay Area commute experienced by a large population of the San Francisco Bay Area every day. Different optimal routes are possible; for instance the route with minimum expected travel-time (LET route), the route which minimizes the travel-time at the speed limit (speed limit based route), the route with the shortest distance (distance based route), the route which maximizes the probability of arriving on time (SOTA route). We generate optimal routes for all of the above strategies using traffic estimates from the Mobile Millennium system. The LET and SOTA routes are computed using the time-dependent implementations of these algorithms. Once the routes are determined, we compute the travel-time distributions for each of these routes a posteriori (this is computed by performing a convolution of the individual link travel-time distributions for the links of each route) and determine the probability of arriving within the budget range of 0 to 60 minutes. Figure 2.6 presents the probability of arriving on time for each of the routing strategies during the budget range.

As traffic conditions vary, the time-dependent SOTA and LET routes change accord-

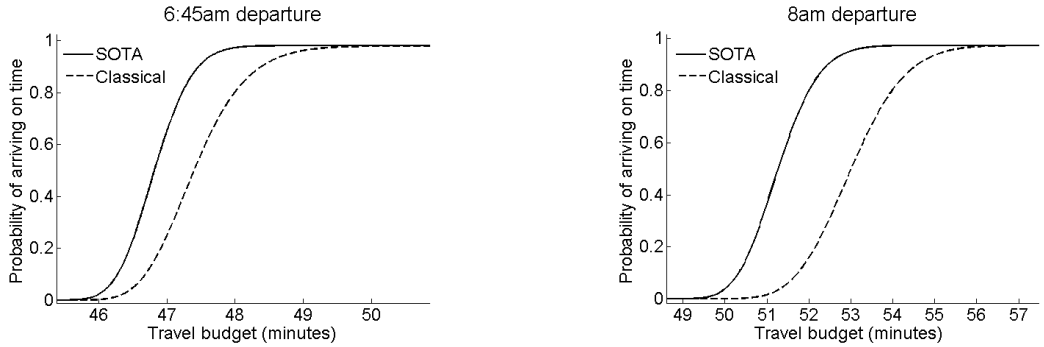


Figure 2.6: **July 29^{th}: Probability of arriving on time at Palo Alto when departing from Berkeley: Left: Departure at** 6:45 **am.** The SOTA policy (solid line) provides a higher probability of arriving on time than the choice of the LET route (dashed line). The distance-based route and the speed limit based route are the same as the LET route at this time (Highway I-880). **Right: Departure at** 8:00 **am.** The SOTA policy and the LET route provide the same probability of arriving on time (solid line). The speed limit based route and the distance based route (dashed line) are inferior for this criterion.

ingly, while the speed limit based route and the distance based route are static. When departing at 6:45 am, the maximal point wise difference between the SOTA route and the LET route is around 0.4, corresponding to a budget of about 47 minutes. For this budget, the commuter has a 0.65 probability of arriving on time on the SOTA route and a 0.25 probability of arriving on time on the LET route. Naturally, for both the SOTA and LET solutions, the risk of not making the destination on time increases as the budget decreases. However, as illustrated in Figure 2.6, the SOTA route always provides a higher probability of arriving on time. Furthermore, the SOTA algorithm can provide the user with the probability of on time arrival (i.e. the risk level) for any range of time budgets the user is interested in when the policy is computed, which allows the user to determine whether the risk is acceptable or not and act accordingly.

One may note that the morning congestion build up is visible in Figure 2.6 since the sharp increase in the cumulative distributions between the left subfigure (6:45 am) and right subfigure (8:00 am) evolves from around 47 minutes to around 52 minutes in this time period. The increase in the area between the SOTA cumulative probability (solid line from Figure 2.6) and a second choice route (dashed line from Figure 2.6) during this time period illustrates that the SOTA policy can dominate classical optimal routes by a higher margin in congestion and non-stationary phases when there is a high uncertainty on the realized travel-time.

We also compare the performance of the SOTA algorithm on the San Francisco arterial network for two routes starting in the Financial District (Columbus and Kearny) and ending at 1) Lincoln and 9th, and 2) Fulton and 2nd. As seen in Figure 2.7, the maximal point wise difference between the SOTA route and the LET route for the first example is around 0.4, corresponding to a budget of about 15 minutes, where the commuter has a 0.75 probability

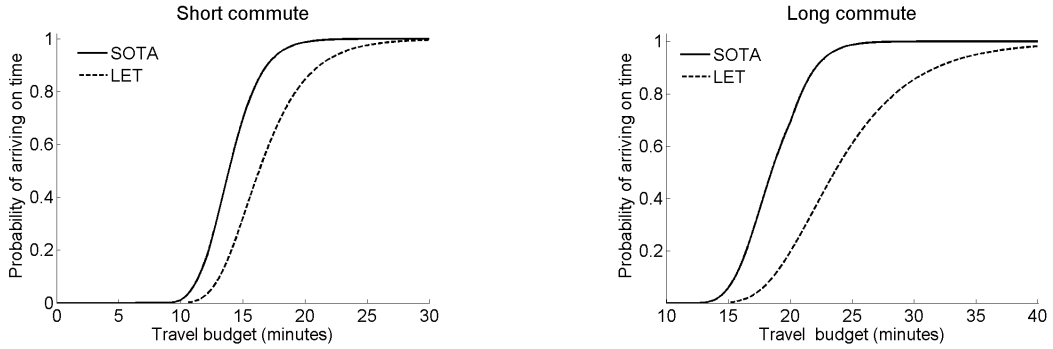


Figure 2.7: **February 1st: Probability of arriving on time at Left: Fulton and 2nd, and Right: Lincoln and 9th when departing from the Financial District (Columbus and Kearny) at** 8**:**50 **pm.** As the graphs imply, the commute to Lincoln and 9th is a longer route than Fulton and 2nd. The relative benefit of using a SOTA policy increases with the route length, since the longer route contains more route choices in the arterial network.

of arriving on time on the SOTA route and a 0.35 probability of arriving on time on the LET route. For the second example, the maximal point wise difference is around 0.5, corresponding to a budget of about 22 minutes, where the commuter has a 0.89 probability of arriving on time on the SOTA route and a 0.39 probability of arriving on time on the LET route. The relative benefit of using a SOTA policy increases with the length of a route, since the longer route contains more route choices (with varying cumulative distribution functions) in the arterial network. In the highway network examples from Figure 2.6, the SOTA policy dominates the LET path for only a 3-5 minute window of the travel-time budget. However, in the arterial network examples, the SOTA policy dominates the LET path for approximately a 10-15 minute window, even though the route lengths were shorter. The reason for this disparity is not limited to these specific examples and is due to the inherent differences of the two networks. The highway network has a limited number of reasonable travel choices from Berkeley to Palo Alto and relatively low variance of the travel-time distributions. Whereas, the arterial network has a large number of route options and highly variable traffic conditions due to the uncertainty introduced by pedestrians, stop signs, traffic lights etc. This results in routes with many distinct cumulative distribution functions and leads to an improved SOTA policy, since the SOTA policy is the upper envelope of all these distinct cumulative distributions functions as illustrated below.

To further illustrate the benefits of the SOTA algorithm when the travel-time distributions are very heterogeneous, we consider the very simple example of two nodes connected by 30 different links each gamma distributed with a mean of 25 minutes, but having different shape and scale parameters. As illustrated in Figure 2.8, the travel-time distributions for the links are vastly different even though they have the same expected travel-time. A LET routing algorithm could pick any of these links as the optimal solution and in the worst case pick the path that is the worst option for the travel-time budget. On the other hand, the

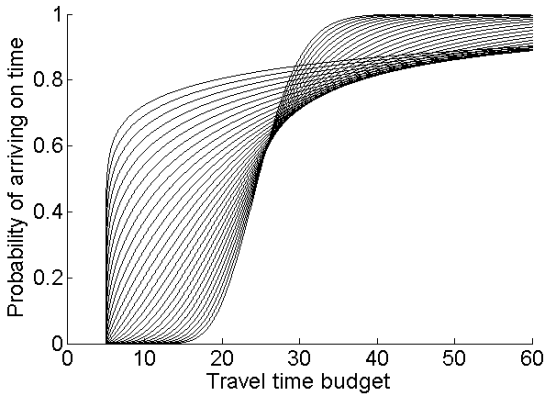


Figure 2.8: **A family of travel-times distributions modeled using 30 shifted gamma distributions, each with the same mean travel time of 25 minutes and a minimum travel time of 5 minutes.** The shape parameter of the distributions ranges from 4 to 0.13 and the scale parameter of the distributions ranges from 5 to 150. The SOTA policy will be the upper envelope of all the curves. The LET path could be any of the curves and in the worst case even be the path that minimizes the probability of arriving on time for a given budget.

SOTA algorithm picks the best path for a given time-budget, which graphically corresponds to the upper envelope of all the curves. As illustrated by this simple example, the SOTA algorithm has the potential for being relatively more superior to a LET path when the number of travel choices increases and their travel-time distributions are not similar.

2.5.3 Test case: evening rush commute within the city of San Francisco

In this section, we illustrate the adaptive nature of the SOTA algorithm presented in this chapter. The output of the algorithm is a policy which accounts for the stochastic nature of link travel-times. Given a budget T, the optimal policy computation encompasses the design of a decision process at each possible intersection of the network; the choice of the optimal route to take from this node depends on the remaining budget.

Here we consider two drivers commuting from point A to point B (see Figure 2.9) on February 1st. They depart from point A at 8:50 pm and desire to reach point B before 9:10 pm; i.e. their travel budget is 20 minutes. We assume that both drivers are equipped with a mobile device on which the output of the SOTA algorithm is available. At each intersection, they follow the turn directions given by the optimal policy. For this test case we sample the drivers travel-time from the output of the real-time arterial traffic estimation module [66] from the Mobile Millennium system.

Because of different driving behaviors, external factors, link travel-time stochasticity, both drivers will experience different travel-times during their commute. The strength of

Figure 2.9: **Commute from point A to point B:** two drivers depart from point A at 8:50 pm on February 1st with a budget of 20 minutes to reach point B. They are routed by the SOTA module. Because their realized travel times differ, their recommended routes differ. The first driver is suggested to turn left at point C, whereas the second driver is suggested to drive straight.

the SOTA algorithm is that the optimal route choice given by the algorithm is given at every intersection in function of the remaining budget. As illustrated in Table 2.6, the optimal route to take from point C includes a left turn for low values of the remaining budget. Because the second driver experienced a larger travel-time on the path from point A to point B, he is advised to take a left turn and to follow a path with more variability, and thus higher risk, which may be more appropriate to his situation. The first driver continues straight at point C.

Remaining budget b at point C	Turn direction from East
$b \leq 12$ minutes	Take a left turn
$b \geq 12$ minutes	Continue straight

Table 2.6: The optimal policy at point C routes on different paths depending on the value of the remaining travel budget with respect to 12 minutes. The probability of arriving on time at B when remaining budget at C is 12 minutes is 0.56.

2.6 Efficient convolutions

As explained in section 2.4.1, the major computational overhead of the SOTA algorithm is the numerical computation of the convolution products in the dynamic program. While the use of localization with the optimal ordering algorithm, as explained above minimizes the total computation time spent on convolutions, the complexity of the FFT based convolution for each link remains $O\left(\frac{T^2}{\delta_i \Delta t}\log\left(\frac{T}{\Delta t}\right)\right)$, with the only change being the minimum link travel-time δ_{ij} being replaced with the minimum loop travel-time of the upstream node δ_i. The asymptotic complexity as a function of T remains the same since each cumulative distribution function $u_i(\cdot)$ is still recomputed at each update step, as described in section 2.3. We assume that $\Delta t = 1$ and denote $\delta = \delta_i$ in the rest of this section for notational simplicity.

Gardner [54] proposed an algorithm called *zero-delay convolution* (ZDC) to compute convolutions more efficiently when the input signal is only available in an online fashion, as is the case in our problem. The complexity of convolving two vectors of length n is reduced from $O(n^2 \log n)$ to $O(n \log^2 n)$ when using this technique. ZDC works by constructing the convolution via a series of smaller block convolutions and thereby eliminating the need to recompute sections of the convolution product that have already been computed. Figure 2.10 illustrates the algorithm. Dean [37] showed that ZDC can be applied to the standard SOTA problem to reduce the computational time complexity of the convolutions in each link from $O(T^2 \log T)$ to $O(T \log^2 T)$.

In our setting, ZDC can be combined with the idea of localization to achieve a computational time complexity of $O\left(T\left(\log^2 T - \log^2 \delta\right)\right)$, which can significantly reduce the computation time for networks with large δ values. We call this algorithm δ-multiple ZDC. The process is as follows. First the optimal ordering algorithm is executed to obtain the update steps for all links. Let τ_i^k be the budget up to which $u_i(\cdot)$ has to be calculated to at the k^{th} update for node i. For ease of explanation, without loss of generality we assume that the update interval δ^k is constant over all updates and that both the budget T and update interval δ are powers of two. Without ZDC, $u(\cdot)$ is updated at each step k by convolving two vectors of length $k\delta$ at a cost of $O(k\delta \log(k\delta))$. This sums to a total time complexity of $O\left(\frac{T^2}{\delta}\log T\right)$ as shown in section 2.3. With δ-multiple ZDC, as with the standard ZDC, the convolution is done in blocks that are reassembled to create the entire convolution. Figure 2.11 shows a simple example with δ=4. Each block is now twice as large as it was with standard ZDC and the computational time complexity is shown to be $O\left(T\left(\log^2 T - \log^2 4\right)\right)$. More generally, the complexity for each link is:

$$\sum_{i=0}^{\lceil(\log T/\delta)\rceil} O\left(T \log\left(2^i \cdot \delta\right)\right) = O\left(T\left(\log^2 T - \log^2 \delta\right)\right). \tag{2.10}$$

Since the optimal ordering algorithms pre-computes the maximum update values for each link, the δ-multiple ZDC algorithm can be run with the most efficient δ value for each link, while preserving correctness invariant given in Equation 2.8.

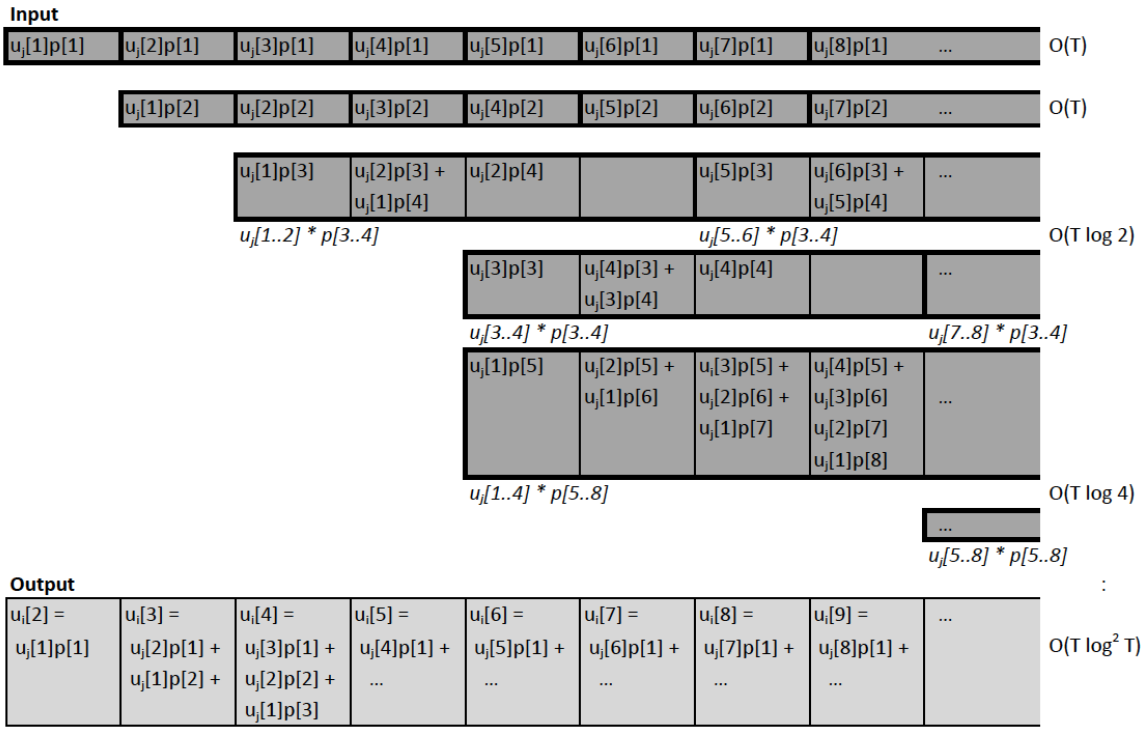


Figure 2.10: Illustration of the zero-delay convolution algorithm from [37]. The ZDC algorithm computes the convolution one column at a time from left to right. The convolutions are computed in blocks and reassembled to avoid recomputation. The functions $u_i[\cdot]$ and $u_j[\cdot]$ are the discrete cumulative distribution functions for the probability of reaching the destination within some time budget from nodes i and j respectively, where $(i, j) \in A$, and $p[\cdot]$ is shorthand for the probability density function $p_{ij}[\cdot]$. We assume that (i, j) is the only outgoing link from node i. Notice that all the components needed to construct $u_i[k + 1]$ are available by the time column k is computed. Some components are computed in advance to exploit the efficiency of block convolutions. The size of the blocks increases exponentially as we proceed through the vectors with the final block having size T. The total computation time is $2T + \sum_{i=1}^{\log T} T \log 2^i = O(T \log^2 T)$.

.

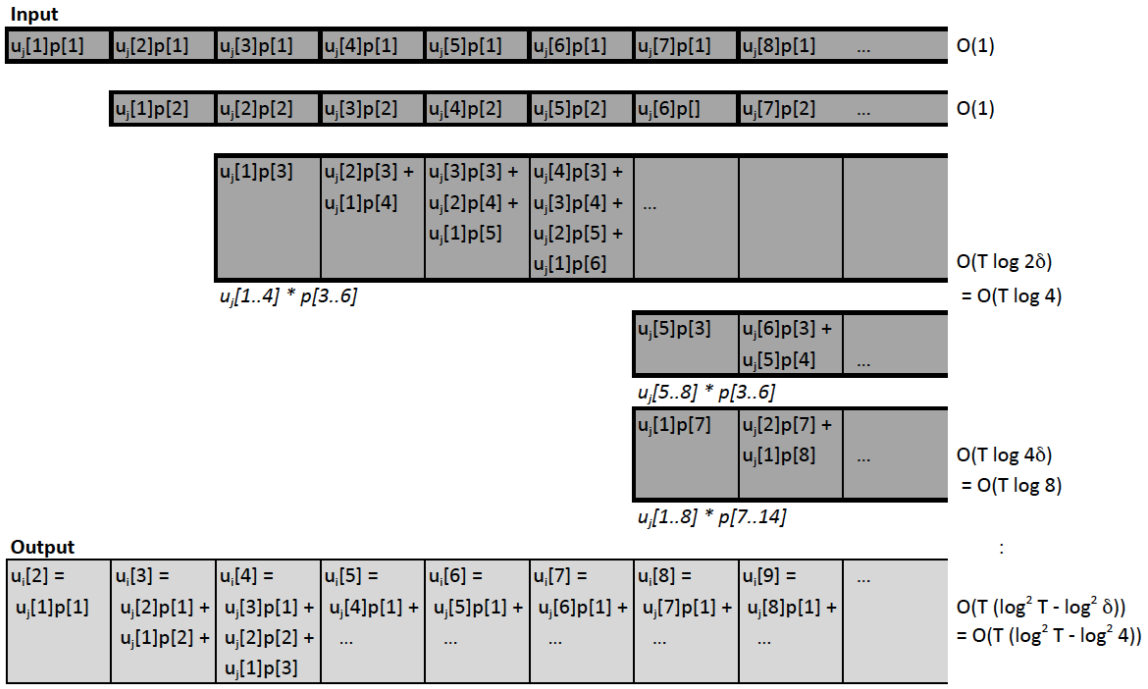


Figure 2.11: Illustration of the δ-multiple zero-delay convolution algorithm for the SOTA problem. We consider a node i with one downstream link (i, j), where $\delta = 4$ and $\delta_{\text{ij}} = 2$. The functions $u_i[\cdot]$ and $u_j[\cdot]$ are the discrete cumulative distribution functions for the probability of reaching the destination within some time budget from nodes i and j respectively, and $p[\cdot]$ is shorthand for the probability density function $p_{ij}[\cdot]$. The first two rows can be computed in constant time, since $\delta_{\text{ij}} = 2$ implies that $p[1], p[2] = 0$. The rest of the convolution is now computed using block sizes that are multiples of δ making the process more efficient than the standard ZDC. Incorporating localization reduces the computational time complexity from $O\left(T \log^2 T\right)$ to $O\left(T\left(\log^2 T - \log^2 \delta\right)\right)$.

2.6.1 Experimental setup

In this section we present numerical results on the performance of the speed-up techniques for the SOTA algorithm (including ZDC) for two types of networks. First we create a set of synthetic networks to illustrate the relative performance of the base algorithm and its optimizations as a function of the structure of the network. Then we provide some numerical results from implementing the algorithms in a traffic information system for the San Francisco Bay Area. The performance of the algorithm is measured as a function of the total budget T. The algorithms are programmed in Java and executed on an Apple Macbook computer with a 2.4Ghz Intel Core 2 Duo processor and 4GB of RAM. We use the open source Java libraries JTransforms [135] and SSJ [79] for FFT computations and manipulating probability distributions. We consider the following combinations of speed-up techniques[7]:

- **SOTA-Brute force**: convolution as a point-wise shifted product.
- **SOTA-FFT**: convolution using the Fast Fourier Transform algorithm.
- **SOTA-FFT-Opt**: convolution using the FFT algorithm, policy updates according to the optimal ordering algorithm.
- **SOTA-FFT-ZeroDelay**: convolution using the Fast Fourier Transform algorithm in a zero delay framework.
- **SOTA-FFT-ZeroDelay-Opt**: convolution using the Fast Fourier Transform algorithm in a zero delay framework, policy updates according to the optimal ordering algorithm.

2.6.2 Synthetic network

In this section we analyze the performances of the speed-up techniques proposed on a Manhattan grid (see Figure 2.12), parameterized by n, the number of arcs on each of the four sides of the grid, δ, the minimal link travel-time, and Δt, the discretization time. We consider the following instantiations of a Manhattan grid:

- Graph A: n=60, $\delta = 5$, $\Delta t = 1$
- Graph B: n=30, $\delta = 10$, $\Delta t = 1$
- Graph C: n=30, $\delta = 20$, $\Delta t = 1$
- Graph D: n=30, $\delta = 5$, $\Delta t = 1$

[7]The successive approximations algorithm from [47] is not considered, since SOTA-Brute force has been shown to outperform it in [103].

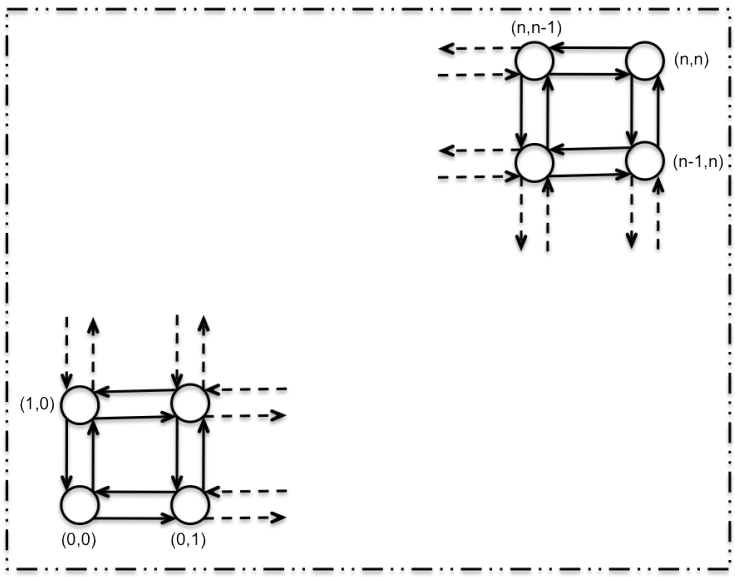


Figure 2.12: **Manhattan Grid** with n arcs along the edge of the grid, and minimal link travel-time δ.

The link travel-times are chosen as shifted Gamma distributions, with the left support boundary at δ, mean travel-time $\mu = 2\,\delta$, and variance $\sigma = 0.5\,\delta$. The origin is defined as the node with coordinates $(0, 0)$ in the grid and the destination is defined as the node with coordinates (n, n) in the grid. Consequently, on algorithm instantiations for which search pruning is used (implicitly via the optimal ordering algorithm in this case), an inflexion point in the runtime can be observed at the budget corresponding to the minimal origin-destination travel-time, corresponding to the fact that the whole graph has been explored by the SOTA policy computation method proposed at this point.

As detailed in the previous section, the runtime of the algorithm depends on the graph size, and on the discretized minimal loop size. In an operational setting, typical nation-wide road networks are composed of two fundamentally different network types, which differ by the inherent structure of their associated graphs, characterized by their minimal graph loop size. Highway networks exhibit large loop sizes, whereas arterial networks are characterized by small loop sizes. Figure 2.13 illustrates the impact of the network structure over the performances of the proposed speed-up techniques for the SOTA algorithm. For a given budget, and fixed discretized loop size, the runtime on a highway network, with large loop travel-times (Figure 2.13, right), is significantly reduced compared to the runtime on a arterial network, with small loop travel-times (Figure 2.13, left). Over hybrid nation-wide networks composed of highway and arterial components, the performance of the algorithm is constrained by the policy computation on arterial networks.

Figure 2.14 illustrates the impact of the network size over the performances of the speed-up techniques. For two graphs with identical network structure, and the larger network

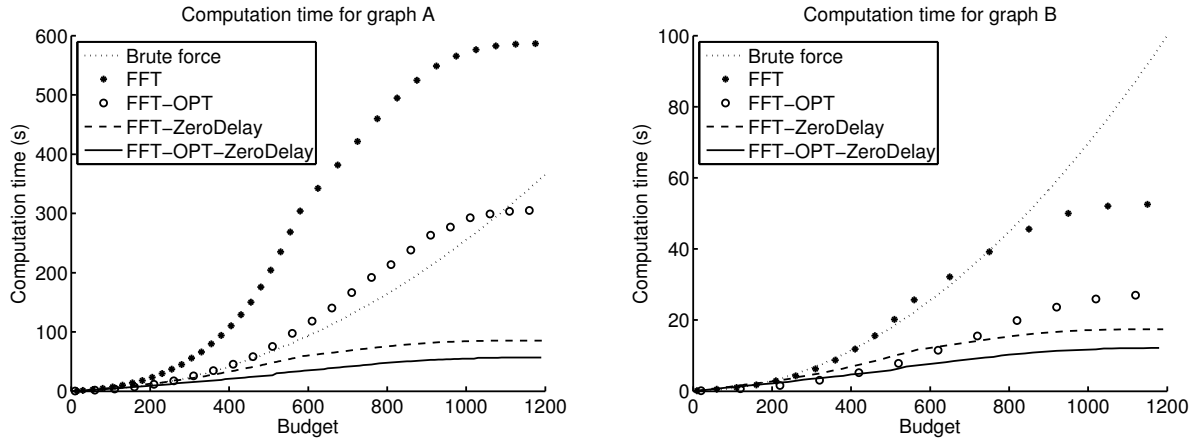


Figure 2.13: **Runtime as a function of budget for different graph structures:** Runtime for computing the optimal policy for graph A, left, with $n = 60$, $\delta = 5$, $\Delta t = 1$, and graph B, right, with $n = 30$, $\delta = 10$, $\Delta t = 1$. The brute force method is represented in dotted line, SOTA-FFT using star markers, SOTA-FFT-OPT using circle markers, SOTA-FFT-ZeroDelay in dashed line, and SOTA-FFT-OPT-ZeroDelay in solid line.

(figure 2.14, left) having travel-times on average four times as long than those of the smaller network (figure 2.14, right), the benefits of the speed-up techniques are more pronounced in the larger network. The SOTA-FFT-ZeroDelay and SOTA-FFT-OPT-ZeroDelay algorithms scale well with the network size, since the computational time complexity is sub-quadratic in the time budget.

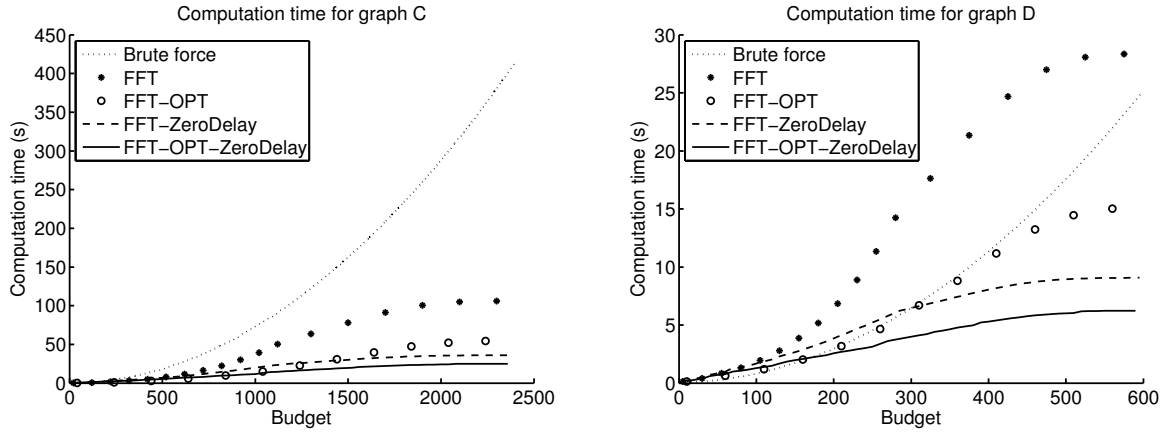


Figure 2.14: **Runtime as a function of budget for different graph size:** Runtime for computing the optimal policy for graph C, left, with $n = 30$, $\delta = 20$, $\Delta t = 1$, and for graph D, right, with $n = 30$, $\delta = 5$, $\Delta t = 1$. The brute force method is represented in dotted line, SOTA-FFT using star markers, SOTA-FFT-OPT using circle markers, SOTA-FFT-ZeroDelay in dashed line, and SOTA-FFT-OPT-ZeroDelay in solid line.

2.6.3 San Francisco Arterial Network

This section presents experimental results comparing the various versions of the SOTA algorithm on a real network from the San Francisco Bay Area. The algorithms are implemented within the Mobile Millennium [95] traffic information system and we test them on the San Francisco arterial sub-network. The network contains 1069 nodes and 2644 links. The travel-time distributions are estimated using the statistical learning algorithm described in [66] using a mixture of real-time and historical probe-generated travel-times. Time-varying link travel-time distributions are obtained a-posteriori from the traffic estimation model. The link travel-time distributions are assumed to be independent. We present the actual runtimes (in CPU time) for a sample origin-destination (OD) pair (see Figure 2.15, right), when computing the optimal policy over a range of travel-time budgets.

As illustrated in table 2.7, the speed-up techniques introduced in this chapter provide a significant gain in runtime for the SOTA algorithm. The consideration of batch computation via FFT-based convolution (SOTA-FFT), presented in section 2.4, increases the runtime compared to the brute force method (SOTA-Brute force) due to the inefficiency of computing multiple convolutions products for the same link, however it allows the use of a localization technique (SOTA-FFT-OPT), introduced in section 2.4.2, providing an order of magnitude speed-up compared to SOTA-FFT overall, and a factor 2 speed-up, for a budget of 30 minutes compared to SOTA-Brute force. Additionally, the zero-delay convolution method, introduced in section 2.6, provides an order of magnitude speed-up (SOTA-FFT-ZeroDelay-OPT) compared to the localized algorithm (SOTA-FFT-OPT). Overall, the combination of the localization technique and the zero-delay convolution bring the runtime on a standard laptop from values comparable to the travel budget, to values below the minute for city-level trips, which fall into the practical range for real-time transportation applications.

Table 2.7: Runtime (in minutes) for different budgets.

Algorithm	Budget 10 minutes	Budget 20 minutes	Budget 30 minutes
SOTA-Brute force	3.3	13.0	29.2
SOTA-FFT	19.1	73.2	154.9
SOTA-FFT-OPT	2.7	9.8	15.0
SOTA-FFT-ZeroDelay	0.8	2.7	5.2
SOTA-FFT-ZeroDelay-OPT	0.3	0.8	1.1

The three best combinations of the speed-up techniques are also illustrated in Figure 2.15, left. The impact of the localization technique, which induces a pruning of the graph and leads to policy updates only for vertices that are feasible given the travel budget, is visible in the typical shape of the curves corresponding to SOTA-FFT-OPT and SOTA-FFT-ZeroDelay-OPT, which illustrate that for large budget, the marginal increase in computation time is limited when using the localization technique because fewer additional vertices

are feasible. On the other hand the computation time curve for SOTA-FFT-ZeroDelay has a convex shape. The complexity reduction provided by the zero-delay method combined with localization is also clearly visible by comparing the computation times for SOTA-FFT-OPT and SOTA-FFT-ZeroDelay-OPT, which decrease from around 15 minutes to around 1 minute for a budget of 30 minutes.

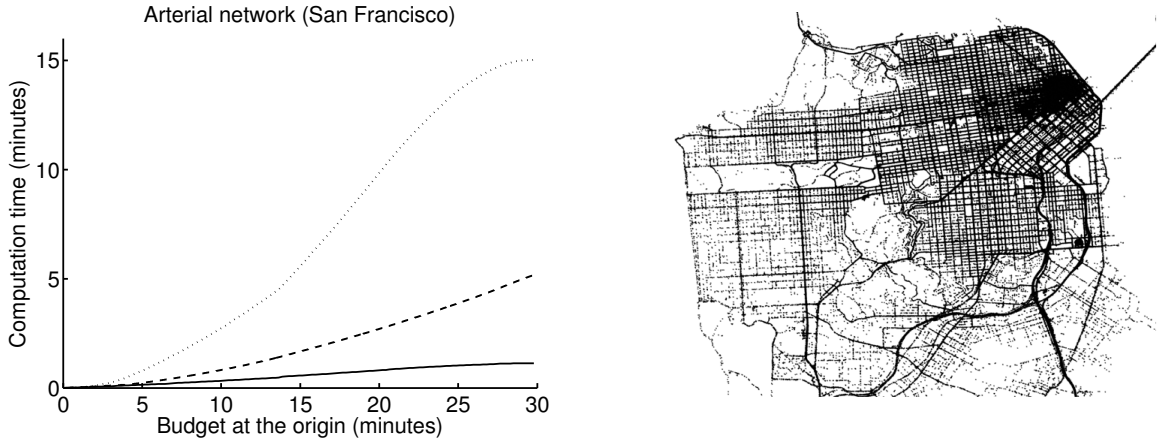


Figure 2.15: **Left: Runtime for computing the optimal policy for a route from the Financial District (Columbus and Kearny) to the Golden Gate Park (Lincoln and 19th)** Comparison of run-times (CPU time) for SOTA-FFT-OPT (dotted line), SOTA-FFT-ZeroDelay (dashed line), SOTA-FFT-ZeroDelay-OPT (solid line). The time discretization (Δt) is 0.4 seconds. **Right: Illustration of the San Francisco Arterial network** Cumulated probe data measurements from the San Francisco arterial network for a single day.

Chapter 3

Precomputation techniques for the stochastic on-time arrival problem

3.1 Introduction

While the previous chapter presented a number of techiques for speeding up the computation of the solution to the stochastic on-time arrival (SOTA) problem, these methods still fall short of the performance requirements for implementation in commercial navigation systems. Therefore, in this chapter, we attempt to further improve the computation time by adapting preprocessing techniques that have been used very successfully in the deterministic shortest path setting. These techniques, which include goal-directed search methods such as $A^\star$, arc-flags [67, 10] and ALT [61], and algorithms that exploit the hierarchy of road networks such as reach [60], contraction hierarchies [55], and transit-node routing [9], can provide speedups of over three orders of magnitude over Dijkstra's algorithm. For example, using transit-node routing, the deterministic shortest path problem can be solved in less than a millisecond for road networks with 20 million nodes and 50 million edges [9].

In this chapter, we explore the use of these preprocessing techniques for improving the query-time of the SOTA problem. We start by identifying some properties of the SOTA problem that limit the types of preprocessing techniques that can be used in this context, and then define the stochastic variants of two deterministic shortest path preprocessing techniques that can be adapted to the SOTA problem, namely reach and arc-flags. We present the preprocessing and query algorithms for each technique, and also present an extension to the standard reach based preprocessing method that provides additional pruning. Finally, we explain the limitations of this approach due to the inefficiency of the preprocessing phase and present a fast heuristic preprocessing scheme. Numerical results for San Francisco, Luxembourg and a synthetic road network show up to an order of magnitude improvement in the query-time for short queries[1], with even larger gains expected for longer queries.

[1]The maximum time budget of our queries is limited by the memory limitations of the hardware and Java implementation.

This work is, to the best of our knowledge, the first attempt at using graph preprocessing techniques to speed up the query-time of the SOTA problem.

Preprocessing constraints of the SOTA problem.

Ideally, we would be able to directly apply the ideas from the deterministic SP problem to the SOTA setting with minimal modifications. However, there are some fundamental differences of the two problems that limit the types of preprocessing techniques that can be used in the SOTA framework. The SOTA solution does not satisfy two important properties that are present in the deterministic SP problem; it cannot be computed in the reverse direction and sub-policy optimality does not hold.

Bidirectional search is not possible. Bidirectional search is a common technique used both in the preprocessing and query stages of fast deterministic routing solutions. For example, Contraction Hierarchies [55] and variants of the arc-flags [67] and reach [59] algorithms rely on the ability to perform bidirectional search. However, speedup techniques that rely on bidirectional search can not be applied to the SOTA problem. As can be seen in equation (2.1), the final and intermediate solutions of the SOTA problem are a function of the remaining time budget, which implies that finding the optimal routing strategy requires this knowledge. When performing a bidirectional search, the reverse search will not have this information.

Lemma 3.1 (Solution on reverse graph)**.** *Let $s, d \in V$ and T be a time budget. The SOTA problem of reaching d from s within T in G is not equivalent to reaching s from d in the reverse graph with the same time budget.*

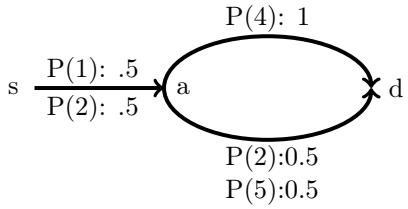


Figure 3.1: A network where the forward and reverse problems are not equivalent

Proof by contradiction. Figure 3.1 depicts a network in which we wish to find the SOTA solution for traveling from the source s to the destination d within 5 time units. The probability of reaching d from s within 5 units of time is 0.75 and the probability of reaching s from d within the same time budget in the reverse graph is 0.5. The reason for this difference is the following. In the forward problem, the path decision from a to d is made using the information about the travel time from s to a and knowing the remaining time budget. However, in the reverse problem, the decision on which path to take from d to a must be

made without any information on the realized travel time between a and s.

Sub-policy optimality does not hold. Sub-path optimality is another commonly utilized property when solving SP problems. If a destination node d has two incoming links a and c (as in figure 3.2), then the optimal path from a source node s to d is either the optimal path from $s \to a$ plus (a, d) or the optimal path from $s \to c$ plus (c, d). However, this basic assumption does not hold in the SOTA setting. To be precise, the optimal SOTA policy from s to d can not be constructed using the optimal policies from s to a and s to c. This prevents us from being able to construct stochastic variants of some the most effective preprocessing techniques such as transit nodes [9] and SHARC [10].

Definition 3.1 (Optimal node set)**.** *Let $V_{sd}(T)$ be the set of nodes that span all realizable optimal paths for reaching a destination d from a source s within a time budget T.*

Lemma 3.2 (Sub-policy sub-optimality)**.** *Let $s, d \in V$, T be a time budget and Φ be a vertex separator of $V_{sd}(T)$. The optimal policy from s to d for a time budget of T can not be constructed using only the optimal policies from s to v and v to d for all $v \in \Phi$.*

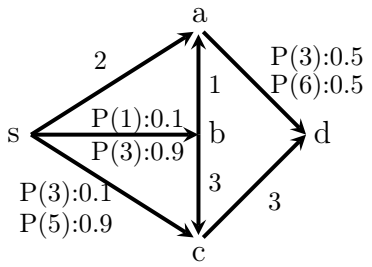


Figure 3.2: A network where the optimal policy cannot be decomposed

Proof by contradiction. In figure 3.2, the best path for going from s to a or s to c for any time budget is via the direct edges (s, a) and (s, c). Also, it is clear that it is necessary to travel through either a or c to reach d. However, in this example, the optimal policy for going from s to d with a time budget of 7 is to first go to b and then chose the next edge between a and c based on the remaining time budget.

3.2 Preprocessing techniques for SOTA

In this section we will focus on two preprocessing techniques that can be adapted to work with the SOTA problem, namely reach and arc-flags. We first define the stochastic variants of these techniques and explain how they can be used for fast query processing in the SOTA problem.

3.2.1 Reach.

The reach [60] of a node is a metric that quantifies the radius of a node's relevance. A node with a small reach value will only belong to shortest paths whose source or destination are close to the node, while a node with a large reach may belong to shortest paths involving sources and destinations that are far from it. We adapt the notion of reach to the stochastic setting of the SOTA problem and present a variant of the reach definition that allows for better pruning.

Definition 3.2 (Stochastic reach)**.** *Let m be a metric and $m(i,j)$ the minimal distance between i and j for this metric. For a node i and some time budget t, we define $\Psi_i(t)$ to be the set of source-destination pairs that contain i in their optimal policy for some time budget smaller than or equal to t.*

$$\Psi_i(t) = \{(s,d) \in V^2 : \exists t' \leq t, i \in V_{sd}(t')\}$$

We define the stochastic reach of a node i for a time budget t as:

$$r(i,t) := \max_{(s,d)\in\Psi_i(t)} \min(m(s,i), m(i,d))$$

The reach of a node can be used to speed up SOTA queries by pruning the graph as described below prior to processing a SOTA query.

Lemma 3.3 (Graph pruning with reach)**.** *Let (s,d) be a source-destination pair and T be the time budget for reaching the destination. Let i be a node and $t \geq T$ a budget for which the reach $r(i,t)$ has been precomputed. Node i can be pruned from the graph without changing the optimal solution if $r(i,t) < \min(m(s,i), m(i,d))$.*

Proof. If node i belongs to the optimal node set for source-destination pair (s,d) with a time budget T, i.e. $i \in V_{sd}(\mathrm{T})$, then $(s,d) \in \Psi_i(t), t > T$ and $r(i,t) \geq \min(m(s,i), m(i,d))$ by definition 3.2. Therefore, if $r(i,t) < \min(m(s,i), m(i,d))$, node i is not on any optimal path for the source-destination pair (s,d) with a time budget T and can be pruned. □

Partition-based reach. One drawback of reach pruning is that the pruned graph could contain a large number of false positives. This is to be expected because the reach metric is computed over the set of all source-destination (s,d) pairs in the network. However, we can improve the precision of the reach metric by computing multiple reach values for each node that are conditioned on some information about the (s,d) pair with respect to which the reach is being computed. One such method is to divide the graph into partitions and compute an individual reach value for each partition. We partition all possible source-destination (s,d) pairs into several clusters with respect to the node for which we are computing the reach, and compute the reach value corresponding to each of these clusters. In the pruning phase, we first find the cluster that the source-destination pair belongs to and look up the corresponding reach value. This leads to more precise reach values and improves the pruning

ability at the expense of an additional memory requirement. There is a trade off between the precision of the reach and the memory used; the two extrema been the regular reach and computing the reach for every possible source-destination pair. We first provide an abstract definition of partition-based reach and then present a specific partitioning scheme.

Definition 3.3 (Partition-based reach)**.** *Let i be a node. Let S be an arbitrary function such that $S(i)$ is a partition[2] of V^2. For notational simplicity let $S_i = S(i)$. For (s,d) in V^2, let $S_i^{(s,d)}$ be the unique set of the partition S_i such that $(s,d) \in S_i^{(s,d)}$. Then $\Psi_i(t) \cap S_i^{(s,d)}$ is the set of source-destination pairs that contain i in some optimal policy with a time budget smaller than or equal to t, and which are in the same cluster of S_i as (s,d).*

The partition-based reach $r_S(i,t)$ on S is defined as:

$$r_S^{(s,d)}(i,t) = \max_{(s',d') \in \Psi_i(t) \cap S_i^{(s,d)}} \min(m(s',i), m(i,d'))$$

It is easy to see that $r(i,t) = \max_{(s,d) \in V^2} r_S^{(s,d)}(i,t)$

Lemma 3.4 (Partition-based reach pruning)**.** *Let (s,d) be a source-destination pair and T be the time budget to reach the destination. Let i be a node and $t \geq T$ a budget for which the reach for the graph has been precomputed. If $r_S^{(s,d)}(i,t) < \min(m(s,i), m(i,d))$, i can be pruned from the graph without changing the optimal solution.*

Proof. If node i belongs to the optimal node set for source-destination pair (s,d) for a time budget T, i.e. $i \in V_{sd}(\mathrm{T})$, then $(s,d) \in \Psi_i(t), t \geq T$. As $(s,d) \in S_i^{(s,d)}$, $r_S^{(s,d)}(i,t) \geq \min(m(s,i), m(i,d))$ by definition 3.3. Therefore, if $r_S^{(s,d)}(i,t) < \min(m(s,i), m(i,d))$, node i is not on any optimal path for the source-destination pair (s,d) with a time budget t and can be pruned. □

Example: Directed reach. The original reach definition [60] does not take into account the position of the source and the destination relative to the candidate node to be pruned. If both source and destination are in the same direction from a node i, it is unlikely that this node i will be used in any optimal path as it requires moving away from the destination. This is the motivation for directed reach.

Definition 3.4 (Directed reach)**.** *Let n be an integer that specifies the number of node sets in the partition and π denote the mathematical constant pi.*

$$I_k := \begin{cases} [\frac{k-1}{n}\pi, \frac{k}{n}\pi[& \text{if } k \in [\![1, n-1]\!] \\ [\frac{n-1}{n}\pi, \pi] & \text{if } k = n \end{cases}$$

[2]Defined as: $\forall P \in S_i, P \neq \emptyset$; $\forall Q \neq P \in S_i, P \cap Q = \emptyset$; $\bigcup_{P \in S_i} P = V^2$.

We define S_i as the function:

$$i \to \{(s,d) : \widehat{sid} \in I_k\}_{k \in [\![1,n]\!]}$$

where $\widehat{sid}$ is the non-oriented angle between s, i and d.

In directed reach, the source-destination pairs that are likely to contain i in their optimal node set (those with $\widehat{sid}$ close to π) are assigned to the same clusters. The benefits of partition-based reach are validated experimentally in the results section.

Metric. The metric m used when computing the reach of a node does not impact the correctness of the solution, but can influence the quality of the resulting pruning. In our experiments, we compared the average travel time and the minimal travel time metrics on random queries. Experimentation showed that the average travel time generally provides better results, but there could be other metrics that we have not tested that perform better.

3.2.2 Arc-flags.

This method [67] is another well-known query speedup technique used in the deterministic SP problem. The idea is as follows: the graph G is divided into a set of regions R which is a partition of the nodes V. Each edge has an associated vector of booleans (with one value for each region) where each boolean is true if the edge is used by at least one SP ending in the corresponding region. During the query phase, prior to computing the shortest path, any edge that do not have the boolean corresponding to the region that the destination belongs is pruned from the graph. Arc-flags also has the nice property of being able to dynamically update the precomputed data [35]. We adapt arc-flags to the SOTA problem in the same way as we did for reach, by replacing the notion of *belongs to a shortest path* by *might be used by an optimal policy.*

Definition 3.5 (Optimal edge set)**.** *Let $E_{sd}(T)$ be the set of edges that span all realizable optimal paths for reaching a destination d from a source s within a time budget T.*

Definition 3.6 (Stochastic arc-flags)**.** *For a node d and some time t, we define $\Gamma_d(t)$ to be the set of edges that belong to some path of the optimal policy for traveling from any source s to the destination d with a time budget less than or equal to t.*

$$\Gamma_d(t) = \{e \in E : e \in E_{sd}(t'), t' \leq t, s \in V\}$$

We define the arc-flag of an edge e for a time budget t and a region $r \in R$ as:

$$AF(e,t,r) := \begin{cases} \textit{TRUE if } e \in \bigcup_{d \in r} \Gamma_d(t) \\ \textit{FALSE otherwise} \end{cases}$$

Arc-flags can be used to speed up SOTA queries by pruning the graph as described below prior to processing a SOTA query.

Lemma 3.5 (Query with the arc-flags)**.** *Let (s, d) be a source-destination pair, r the region d belongs to and T the time budget for reaching d. Let e be an edge and $t \geq T$ a budget for which the arc-flags has been precomputed. If $AF(e, t, r)$ is false, e can be pruned from the graph without changing the optimal solution.*

Proof. If e belongs to some path of the optimal policy for the source-destination pair (s, d), i.e. $e \in E_{sd}(\mathrm{T})$, then we have $e \in \Gamma_d(t)$ by definition 3.6. Furthermore, since r is the region that the destination d belongs to, it also follows that $AF(e, t, r)$ is true. □

3.2.3 Computing the reach and arc-flags.

One of the major limitations of this approach is the large computation time required to calculate the stochastic reach and arc-flags of the network. In the deterministic SP context, this limitation can be overcome by exploiting the property of sub-path optimality to come up with efficient algorithms for computing these metrics. For instance in arc-flags, it is easy to see that one only has to consider the boundary of a region as possible destinations nodes, since the optimal path to any node within the region from a node outside the region will include some optimal path to the boundary of the region. However, as we have seen in section 3.1, this does not work in the SOTA setting. Similarly, the reach metric can also be computed efficiently using a hierarchical method that takes advantage of sub-path optimality, as shown in [60]. Unfortunately, to this point, we have not been able to identify an efficient algorithm for computing the stochastic reach and arc-flags. We are exploring the possibility of upper bounds for the reach metric similar to what is done in the original article by Gutman [60].

Our current approach is to compute the reach and arc-flags in a brute force manner by running a SOTA search for all possible destinations in the graph. Table 3.4 in the results section shows that this approach is still tractable for some urban scale networks. It is important to note that the computation for each destination is an independent problem and can be done in parallel.

Computing reach. A high level overview of the process is given in Algorithm 3.1. For each destination d, we compute the optimal policy from all sources s to d for the time budget T. The SOTA policy for all sources can be computed simultaneously [118]. Then, for all sources s, we determine the nodes $i \in \bigcup_{t \leq T} V_{sd}(t)$ that belong to some optimal path from s to d and update their reach. Determining the set $\bigcup_{t \leq T} V_{sd}(t)$ is not trivial since it requires considering all possible realizations of the SOTA policy. We use a efficient priority queue based search with no re-computation of paths, but finding the set $\bigcup_{t \leq T} V_{sd}(t)$ and updating the reach from all the sources can take up to twenty times longer than finding the optimal policy. Furthermore, the value $m(s, i)$ might need to be computed multiple times when considering different destinations, but keeping these values in memory for all (s, i) requires too

Algorithm 3.1 Reach computation

Input: a graph G and a time budget T
Output: the reach $r(\cdot,\mathrm{T})$
Initialization: $\forall i \in V$, $r(i,\mathrm{T}) = 0$
for $d \in V$ **do** ▷ can be computed in parallel
  compute the optimal policy with budget T; $\forall s \neq d \in V$
  for $i \in V$ **do**
    compute $m(i,d)$
  end for
  for $s \in V$ **do**
    compute $\bigcup_{t \leq T} V_{sd}(t)$
    for $i \in \bigcup_{t \leq T} V_{sd}(t)$ **do**
      compute $m(s,i)$
      set $r(i,\mathrm{T}) = \max(r(i,\mathrm{T}), \min(m(s,i), m(i,d)))$
    end for
  end for
end for
return r

large of a memory footprint. Therefore, we recompute these values for each destination as needed. Fortunately, $m(s,i)$ can be computed efficiently in the sub-graph induced by the nodes $v \in \bigcup_{t \leq T} V_{sd}(t)$, which is much smaller than G. This approximation will lead to an upper bound for the reach, but this does not impact the optimality of the solution since it is an upper bound. This bound is usually quite tight since the shortest path is usually close to some path in the SOTA policy.

Computing arc-flags. Computing the arc-flags of the network is much a more straightforward process, since the arc-flags do not depend on the source. A high level overview of the process is given in Algorithm 3.2. Once again first the optimal policy for each destination is computed for the maximum time budget of interest. Then for each destination the optimal edge set $E_{sd}(t)$ is computed and all the edges in this set are marked as true for the region that the destination belongs to. To optimize the preprocessing, the arc-flags and reach can be computed simultaneously, since they both use the SOTA solution for each destination d and this computation can be shared.

3.3 Experimental results

Test instances. We use three different networks to test our algorithms: a San Francisco arterial network[3] (SF) with 2450 nodes and 6151 edges, a Luxembourg network with 30674

[3]We artificially modify the original network by adding two fast roads (highways) in the South/North and East/West directions, since the original network does not contain highways and thus has poor hierarchy.

Algorithm 3.2 Arc-flags computation

Input: a graph G, a partition of the edges R and a time budget T
Output: the arc-flags $AF(\cdot,\mathrm{T},\cdot)$
Initialization: $AF(e,\mathrm{T},r) = \mathrm{FALSE}\ , \forall (e,r) \in E \times R$
for $d \in V$ **do** ▷ can be computed in parallel
  compute the optimal policy with budget T; $\forall s \neq d \in V$
  compute $\bigcup_{t \leq T, s \in V} E_{sd}(t)$
  for $e \in \bigcup_{t \leq T, s \in V} E_{sd}(t)$ **do**
    set $AF(e,\mathrm{T},r(d)) = TRUE$
  end for
end for
return AF

nodes and 72492 edges and a synthetic network with 7921 nodes and 31328 edges. The synthetic network is a 89 × 89 Manhattan grid with 4 levels of roads, where the speed limits are 40, 60, 80 and 120 kmph, and the size of the network is 40 × 40 km. The travel time distributions for the SF network is derived from real world observations [68]. The travel time distributions for the Luxembourg network are created artificially using the speed limits as a baseline, as actual travel time information is not available. The distributions for the synthetic network are also generated using the same strategy. All distributions are a mixture of Gaussians corresponding to different traffic modes like slow or fast where all the weight over the speed limit has been moved to the minimal travel time.

Environment. The precomputation was done on 18 × 1.9GHz AMD Opteron(tm) 6168 processors with 30Gb of shared memory, and the queries where were performed on an Intel Core i7 Q740 with 8 × 1.73GHz cores and 4Gb of memory. All the code used for the experiments was written in Java 1.6. For each experiment, we randomly picked 200 source-destination pairs with a positive probability of arriving of time. The time discretization of the probability distributions for all experiments is one second.

Technique	Time budget (s)			
	300	500	800	1000
reach (RH)	1.2	1.5	1.4	1.3
directed-RH (DRH)	2.4	3.2	2.9	2.4
arc-flags (AF) 10x10	3.0	5.6	4.6	3.5
DRH & AF 10x10	4.8	8.8	7.2	5.4
baseline runtime (ms)	12	115	579	1554

Table 3.1: Relative speedups over no preprocessing (San Francisco)

Technique	Time budget (s)			
	500	1000	1500	2000
reach (RH)	1.5	1.7	1.9	2.2
directed-RH (DRH)	2.0	2.2	2.4	2.8
arc-flags (AF) 20x20	1.5	2.3	3.3	4.8
DRH & AF 20x20	2.5	3.8	5.2	7.2
baseline runtime (ms)	12	283	883	1369

Table 3.2: Relative speedups over no preprocessing (Luxembourg)

Technique	Time budget (s)			
	500	1000	1500	2000
reach (RH)	1.2	1.3	1.4	1.4
directed-RH (DRH)	1.7	2.0	2.1	2.4
arc-flags (AF) 10x10	2.0	3.0	3.2	3.6
arc-flags (AF) 20x20	3.3	5.1	5.8	5.5
DRH & AF 10x10	2.7	4.6	4.9	5.4
DRH & AF 20x20	3.9	6.8	7.5	8.1
baseline runtime (ms)	23	315	1293	4843

Table 3.3: Relative speedups over no preprocessing (synthetic network)

Speedup. Tables 3.1, 3.2 and 3.3 present the average speedups achieved using the preprocessing methods described above compared to computing the results with no preprocessing. The number associated with the arc-flags is the number of regions used. The general speedups achieved are fairly consistent across all the networks. As expected, the directed-reach (DRH) performs better than the regular reach (RH). Also, the performance of the arc-flag (AF) algorithms improves as we increase the number of regions used. Finally, we see that combining reach and arc-flags provides the best results. Figure 3.3 shows a visualization of how the four different preprocessing algorithms reduce the query-time search space of the problem. One important observation is that the speedups achieved using the preprocessing techniques increases with the time budget of the query. The decrease in the San Francisco network for large budgets is due to the boundary effects of the smaller graph. This is important because the total computation time increases with the budget and the efficiency of the SOTA algorithm must scale well with the time budget.

The stochastic reach and arc-flags algorithms are generally less efficient than their deterministic counterparts in term of proportional speedup. The first explanation we can

Figure 3.3: Pruning of the San Francisco network for some source-destination pair. From left to right the pruning achieved using reach, directed-reach, arc-flags and the combination of reach with arc-flags. The colored nodes are the nodes that belong in the optimal policy and the color denotes the probability of the node being used, where blue indicates a high probability and red indicates a low probability. The source is in the bottom of the graph.

Network	Speedup technique						
	reach	arc-flags		heuristic arc-flags			
	1000	1000	2000	10x10		20x20	
				1000	2000	1000	2000
SF	8 497	456	1 556	230	711	317	1 133
Synthetic	18 305	1 303	12 099	584	4 420	860	7 799
Luxembourg	45 643	7 907	75 898	316	5 325	579	8 317

Table 3.4: Precomputation time in seconds for reach, arc-flags and heuristic arc-flags. Results are presented for maximum time budgets of 1000 and 2000 seconds.

give is that the set $V_{sd}(\mathrm{T})$ contains more nodes than a specific shortest path, 50% more on average in our experiments. This means that the individual reach values of nodes are likely to be higher in the stochastic setting and that more edges are likely to be labeled as true in stochastic arc-flags. The stochastic reach and arc-flags are also functions of the time budget being considered, which makes it hard to give meaningful comparisons with the deterministic versions. The second explanation is that we have not performed queries for large time budgets due to computational resource limitations. We have limited the range of our preprocessing to trips of 2000 seconds. The best speedups achieved in deterministic road networks are obtained for longer queries. For instance the deterministic reach speedups reported in [60] show that the speedup increases with the length of the path from 4.5 to 19.2 on average for road lengths of 26 and 56 km. Similarly, the significant speedups for the basic arc-flags algorithm were obtained in the German countrywide network where the average trip is much longer [67]. We expect the speedups of the stochastic variants to also increase as we consider longer queries. One of the immediate next steps is to re-implement our algorithms in a programming language with better memory management features and to gain access to better hardware resources to run the experiments on larger networks for

larger time budgets.

Heuristic precomputation. As discussed in section 3.2.3, preprocessing the graph using stochastic reach and arc-flags is very inefficient due to the additional constraints of the SOTA problem. However, it is possible to efficiently compute heuristics of the reach and arc-flags that are close to optimal values in practice. One such approach is the compute the arc-flags by only considering destinations that are on the boundary of each region, as is done in the deterministic case. This cuts the total number of nodes that need to be considered by a considerable factor (specially when the regions are large and the number of nodes per region is large).

When computing the heuristic arc-flags for the synthetic network we noticed a 0.4% reduction in the total number of arc-flags, i.e. a 0.4% false negative rate. When computing the SOTA solution for 1000 random queries we found 8 non-optimal solutions, and the largest deviation in the probability of arriving on time was 10^{-5}. The results for the SF and Luxembourg networks were similar. Such approaches are especially promising for commercial applications where such minor deviations from the optimal solution are negligible and the measurement error in the probability distributions dominates the error.

The heuristic arc-flags can also be used to compute heuristic reach values as follows. First run Algorithm 3.1 and 3.2 using only destination nodes that are on the boundary of the regions. Then run Algorithm 3.1 from all the other destinations using the heuristic stochastic arc-flags to speed up the optimal policy computation.

Part II

System optimal dynamic traffic assignment

Chapter 4

Discrete-time system optimal dynamic traffic assignment (SO-DTA) with partial control for horizontal queuing networks

4.1 Introduction

In this chapter, we consider the *system optimal dynamic traffic assignment problem with partial control* (SO-DTA-PC) for general networks with horizontal queuing. The goal of which is to optimally control any subset of the network demand to minimize the total congestion across all the demand in the network. A brief introduction the dynamic traffic assignment is given in Section 1.3.

The single destination SO-DTA problem (with full control) can be formulated as a *Linear Program* (LP) under a LP relaxation that approximates the non-linear dynamics of the system [140]. However, the SO-DTA problem with partial control can not be formulated as a convex problem, even in the case of a single destination, without violating the *first-in-first-out* (FIFO) condition [22], due to the multiple commodities (selfish and cooperative demand) in this problem. Furthermore, solving the SO-DTA problem with an LP relaxation of the dynamics can lead to the holding of vehicles on links when the model allows for a larger flow. It has been argued that this holding can be achieved in practice via *variable speed limit* (VSL) signs [140] and makes sense when the goal of the problem is to also solve for the optimal VSL values [62], but is impractical to implement in most cases. Thus, there is a need for a more general solution that does not require VSL.

When solving an optimization problem subject to non-convex (potentially non-smooth) system dynamics, one can either relax the model to make the optimization problem linear (or convex) [20], or keep a more realistic model and use non-convex optimization methods [14], with the trade-off being computational complexity versus model accuracy. In this work, the goal is to use a model that is as close to the real world dynamics as possible, since we hope to use it in an operational setting for highway corridors in California. A further complication

of DTA in practical settings is the unavailability of *origin-destination* (OD) data for the entire demand. Most DTA solutions assume that this data is available, although it can be challenging to obtain in practice. Therefore, we formulate the partial control problem in a manner that requires full OD information only for the demand that can be controlled by the central authority, and junction split ration for the remaining demand (which are much easier to obtain via inductive loop detectors for example).

In this work, we formulate the *system optimal dynamic traffic assignment problem with partial control* (SO-DTA-PC), using a traffic dynamics model similar to the *Cell Transmission Model* (CTM) [31, 34], which is a Godunov discretization of the *Lighthill-Williams-Richards* (LWR) *partial differential equation* (PDE) [82, 111] with a triangular fundamental diagram[1]. The CTM is a horizontal queuing model and uses a latency function that gives a constant delay when the traffic density is below a certain threshold and progressively increases as the density increases beyond this threshold, and is well accepted in the transportation community as a good first order approximation of road traffic dynamics.

We propose solving the SO-DTA-PC problem with the non-convex traffic dynamics from [38] and limited OD data with complete split ratios as a non-linear optimal control problem. This formulation generalizes to multiple sources and multiple destinations. The next challenge is in finding efficient descent methods for this non-convex optimal control problem. There is a vast literature on optimization techniques for non-convex control problems (see [14] and the references therein) that can be utilized to solve this problem. While gradient based methods do not provide any guarantees of converging to the optimal solution in non-convex optimization problems, they can still be used to find local minima. One of the main computational challenges in this approach is the efficient computation of the gradient, since this computation must be repeated a large number of times. We show that the structure of our dynamical system allows for very efficient computation of the gradient via the discrete adjoint method [11, 56, 57, 71, 109]. If the state vector is n dimensional and the control vector is m dimensional, direct computation of the gradient takes $O(n^2m)$ time. The adjoint method generally reduces the complexity to $O(n^2 + nm)$, but the structure of our system allows for further reduction of the complexity to $O(n + m)$ by exploiting the sparse nature of the forward system.

It should be noted that this work currently only considers computing the optimal allocation of the cooperative agents and does not consider the corresponding response from the selfish agents. It is clear that a change in the network state will result in response by the selfish agents as in Stackelberg games. Finding the optimal control for a Stackelberg game is NP-Hard in the size of the network for the class of increasing latency functions even in the case of the static problem [113] and it is common to use approximate strategies [113, 130]. We wish to extend this work in the future to consider the Stackelberg response to the SO-DTA-PC problem.

The rest of the chapter is organized as follows. In Section 4.2, we present the traffic dynamics model with the corresponding junction solver and the notion of controllable and

[1]The flux as a function of the density takes a triangular shape.

non-controllable flows. In Section 4.3, we describe the forward system and provide explicit solutions to the junction problem. Section 4.4 formulates the optimization problem and derives the adjoint system, which is used to efficiently compute the gradient of the optimization problem. Section 4.5 concludes the chapter with some experimental results.

4.2 Traffic model

This section describes the multi-commodity traffic flow model that is used in this chapter. We begin by introducing some frequently used notation and then provide a detailed description of the flow model.

4.2.1 Notation

Constants

Δt, T	Time discretization and number of time steps[2]
v_i	Free flow speed on cell i
w_i	Congestion wave speed on cell i
L_i	Length of cell i
ρ_i^{jam}	Jam density on cell i
F_i	Max flow capacity of cell i
P_{ij}	Merge priority factor from cell i to cell j

Sets

$\mathcal{J}_z^{\text{in}}$	Incoming links to junction z
$\mathcal{J}_z^{\text{out}}$	Outgoing links to junction z
$\mathcal{A}$	The cells (including buffers and sinks)
$\mathcal{B}$	The buffer cells
$\mathcal{S}$	The sink cells
OD	The set of origin-destination (OD) pairs
$\Gamma^{-1}(i)$	The predecessors of cell i
$\Gamma(i)$	The successors of cell i
$\mathcal{C}$	The commodities
$\mathcal{CC}$	The controllable commodities

Inputs

$\rho_{i,c}(0)$	Initial density of commodity c on cell i
$\beta_{ij,c}(k)$	Split ratio for commodity c on cell i to cell j, time step k
$D_{(o,s)}(k)$	Demand rate of controllable agents going from $o \in \mathcal{B}$ to $s \in \mathcal{S}$, time step k

[2]Time from $t = 0$ and the final time step $T = T - 1$.

$D_{i,c}(k)$ Demand rate of non-controllable agents of commodity $c \in \mathcal{NC}$ on cell i, time step k

Variables

$f_i^{\text{in}}(k)$ Total flow into cell i, time step k
$f_i^{\text{out}}(k)$ Total flow out of cell i, time step k
$f_{i,c}^{\text{in}}(k)$ Flow of commodity c into cell i, time step k
$f_{i,c}^{\text{out}}(k)$ Flow of commodity c out of cell i, time step k
$\rho_i(k)$ Density on cell i, time step k
$\rho_{i,c}(k)$ Density contribution of commodity c on cell i, time step k
$\sigma_i(k)$ Supply on cell i, time step k
$\delta_i(k)$ Demand on cell i, time step k
$d_i(k)$ Boundary demand on cell i, time step k
$\gamma_c(k)$ Demand allocation for commodity c, time step k

4.2.2 Basic definitions

The aggregate traffic dynamics are modeled using a macroscopic traffic flow model based on the *Lighthill-Williams-Richards* (LWR) PDE [82, 111]. We use a multicommodity variant with buffers of an earlier PDE model developed in [53]. This model imposes strong boundary conditions at the entrances to the network, so that no vehicles are dropped due to congestion propagating outside the bounds of the network, an important consideration in the optimal control setting. We then use a Godunov discretization [58] of the network PDE model as explained in [110] to obtain an equivalent discrete model.

Network. The road network is divided into cells, indexed by $i \in \mathcal{A}$. We add a buffer cell at the entrances of the network, to be able to impose the boundary demands in a strong sense. Each junction, indexed by $z \in \mathcal{J}$, connects a set of incoming links $\mathcal{J}_z^{\text{in}}$ to a set of outgoing link $\mathcal{J}_z^{\text{out}}$. The total flow in the network is decomposed into a set of $|\mathcal{C}|$ commodities that correspond to different types of flow.

Definition 4.1 (Supply and demand)**.** *The supply of a cell i at time step k, denoted $\sigma_i(k)$, is maximal flow that can enter the cell, while the demand $\delta_i(k)$ is the maximal flow that can leave the cell. By assumption, buffers have no supply and the sinks have no demand.*

Definition 4.2 (Density)**.** *The density on a link i at time step k, denoted by $\rho_i(k)$, is the total number of vehicles on the link during that time step divided by the length of the link L_i. The vehicles in the link could be from any of the $|\mathcal{C}|$ commodities in the network.*

Definition 4.3 (Single commodity density)**.** *The density induced by a single commodity c on a link i at time step k, denoted by $\rho_{i,c}(k)$, is the total number of vehicles of commodity c on the link during that time step divided by the length of the link L_i, and satisfies*

$$\rho_i(k) = \sum_{c \in \mathcal{C}} \rho_{i,c}(k) \tag{4.1}$$

Definition 4.4 (Initial conditions)**.** *The initial conditions of the network are the densities of each commodity at each link at time step $k = 0$ and are denoted $\rho_{i,c}(0)$.*

Definition 4.5 (Inflow and outflow)**.** *The inflow (resp. outflow) from a cell i at time step k, denoted $f_i^{\text{in}}(k)$ (resp. $f_i^{\text{out}}(k)$), is the total flow leaving (resp. entering) the cell at time step k. By assumption, buffers have no inflow and sinks have no outflow.*

Definition 4.6 (Single commodity inflow and outflow)**.** *The inflow (resp. outflow) from a cell i at time step k corresponding to commodity c, denoted $f_{i,c}^{\text{in}}(k)$ (resp. $f_{i,c}^{\text{out}}(k)$), is the total flow of commodity c leaving (resp. entering) the cell at time step k.*

$$f_i^{\text{in}}(k) = \sum_{c \in \mathcal{C}} f_{i,c}^{\text{in}}(k) \tag{4.2}$$

$$f_i^{\text{out}}(k) = \sum_{c \in \mathcal{C}} f_{i,c}^{\text{out}}(k) \tag{4.3}$$

Definition 4.7 (State evolution)**.** *The state of the network at time step k is given by the density $\rho_{i,c}(k)$ of each commodity c at each cell i. The density evolution is governed by the following dynamics, which simply correspond to mass conservation.*

$$\rho_{i,c}(k) = \rho_{i,c}(k-1) + \frac{\Delta t}{L_i}\left(f_{i,c}^{\text{in}}(k-1) - f_{i,c}^{\text{out}}(k-1)\right) \quad \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![1, T_f]\!],\ \forall c \in \mathcal{C} \tag{4.4}$$

$$\rho_{i,c}(k) = \rho_{i,c}(k-1) + \frac{\Delta t}{L_i} \cdot f_{i,c}^{\text{in}}(k-1) \qquad \forall i \in \mathcal{S},\ \forall k \in [\![1, T_f]\!],\ \forall c \in \mathcal{C} \tag{4.5}$$

with initial condition

$$\rho_{i,c}(0) = \rho_{i,c}^0 \qquad \forall i \in \mathcal{A} \setminus \mathcal{S},\ \forall c \in \mathcal{C} \tag{4.6}$$

$$\rho_{i,c}(0) = 0 \qquad \forall i \in \mathcal{S},\ \forall c \in \mathcal{C} \tag{4.7}$$

Assumption 4.1. *The flux function defining the relationship between density and flow is given by the triangular fundamental diagram shown in figure 4.1. This is a first order approximation of the empirical relationship between flow and density [42].*

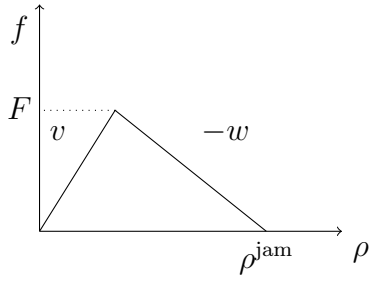


Figure 4.1: Triangular fundamental diagram.

Assumption 4.2 (First-in first-out (FIFO) property)**.** *We assume that no vehicles entering a link at time step t will overtake vehicles that have already entered the link at some time step $t' < t$.*

4.2.3 Discretization requirements

The network model is composed of links (each link representing a physical road) and nodes (each node representing a junction between some incoming and outgoing links). As with any numerical scheme, the accuracy of a macroscopic traffic model increases with the granularity of the network discretization. In addition, to ensure the convergence of the solution of the discretized model to the solution of the continuous LWR equation as the time step Δt and space discretization L goes to zero, the network must satisfy the *Courant-Friedrichs-Lewy* (CFL) conditions, which are standard requirements in numerical analysis [58, 81].

Requirement 4.1 (CFL condition 1)**.** $\forall i \in \mathcal{A} \setminus \mathcal{S},\ v_i \leq \frac{L_i}{\Delta t}$

For numerical stability, the vehicles in a given cell should only be able to travel forward at most one cell in a single time step. Requirement 4.1 ensures that this condition is satisfied by imposing an upper bound on the velocity.

Requirement 4.2 (CFL condition 2)**.** $\forall i \in \mathcal{A} \setminus \mathcal{B},\ w_i \leq \frac{L_i}{\Delta t}$

Each cell can not have a density greater than ρ^{jam}. Requirement 4.2 ensures that this condition is satisfied.

Proof. While the requirement that $v_i \leq \frac{L_i}{\Delta t}$ comes from the positive density, this one comes from the fact that the density has to be smaller than ρ_i^{jam}. Indeed, in the case of a cell with no outflow (because of an extreme congestion of the next cell), the inflow can be limited by the supply. If there is enough demand we have at the next time the density which has to be smaller than ρ_i^{jam}:

$$\rho_i(k) + \frac{\Delta t}{L_i} w_i(\rho_i^{\text{jam}} - \rho_i(k)) < \rho_i^{\text{jam}} \tag{4.8}$$

$$\rho_i(k)\left(1 - w_i\frac{\Delta t}{L_i}\right) < \rho_i^{\text{jam}}\left(1 - w_i\frac{\Delta t}{L_i}\right) \tag{4.9}$$

And because $0 \leq \rho_i(k) \leq \rho_i^{\text{jam}}$ we also have $1 - w_i\frac{\Delta t}{L_i} \geq 0$ which is the requirement. □

Requirement 4.3 (Finite time density discharge)**.** $\forall i \in \mathcal{A},\ v_i \geq \frac{L_i}{\Delta t}$

This condition guarantees that the density of a given cell should discharge in a finite amount of time when there is no incoming flow.

Proof. If $v_i < \frac{L_i}{\Delta t}$, we can have exponential decrease of the density in some cells while they should be emptied in only 1 steps. Indeed, taking the case of a cell without inflow, we have $\rho_i(k+1) = \rho_i(k) - \frac{\Delta t}{L_i}\rho_i(k)\, v_i$ which gives $\rho_i(k+t) = \rho_i(k)\,(1 - \frac{\Delta t v_i}{L_i})^t$. This is not an acceptable physical solution and thus should be excluded. □

Remark 4.1. *Satisfying requirements 4.1, 4.2 and 4.3 implies imposing* $v_i = \frac{L_i}{\Delta t}$. *Given that the velocity* v *is an exogenous parameter, this imposes a strict requirement on the space discretization of the road segments. However, any given road segment might not be divisible into cells of exact length* $v_i \cdot \Delta t$, *and in most cases a cell of length* $L \in (0, v_i \Delta t)$ *will remain. There are multiple solutions to this issue:*

- *Approximate the length of each road segment to be a multiple of* $v_i \cdot \Delta t$. *The relative rounding error decreases as the road gets longer and the descretization* Δt *gets smaller.*
- *Change the dynamics of the last cell in each road segment to have a special case that allows for vehicles to be fully discharged when the supply of the downstream link allows it. This makes the dynamics equations and optimization problem more complicated.*
- *Accept this model limitation and have a small amount of density stuck in the network. This is not so bad in practice, since the number of vehicles stuck in a link decreases exponentially with time.*

4.2.4 Controllable and non-controllable flow

There are two types of flows that are transported in the network. Controllable flows that have origin destination requirements, but can be routed along any path in the network, and non-controllable flows that have fixed paths. These flows are modeled by distributing the total flow of the network into multiple commodities, as explained below.

Assumption 4.3 (Path decomposition of controllable flow)**.** *We assume that the controllable flows from each origin destination pair are restricted to a small pre-determined subset of paths in the network.*

Definition 4.8 (Non-controllable commodity)**.** *There is a single non-controllable commodity* c_n *that represents all non-controllable flow in the network.*

The paths of the flow corresponding to the non-controllable commodity are defined via the junction split ratios.

Definition 4.9 (Split ratio)**.** *The split ratio of a commodity* c *at cell* i *and time step* k *among the outgoing cells* $j \in \Gamma(i)$, *denoted* $\beta_{ij,c}(k)$, *is the fraction of the commodity* c *flow out of cell* i *that is entering cell* j *at time step* k.

$$\sum_{j \in \Gamma(i)} \beta_{ij,c}(k) = 1 \tag{4.10}$$

Definition 4.10 (Controllable commodities)**.** *The controllable commodities* $c_c \in \mathcal{CC}$ *correspond to the controllable flow. There is a unique controllable commodity that corresponds to each path that the controllable flow can be routed along in the network. A controllable commodity is then equivalent to a tuple* $(origin, destination, path)$.

Definition 4.11 (Conservation of flow)**.** *The path of a controllable commodity is defined via the junction split ratios for this commodity.*

$$\beta_{ij,c}(k) = \begin{cases} 1 \text{ if the path of commodity } c \text{ includes cell } i \text{ and cell } j \\ 0 \text{ otherwise} \end{cases} \tag{4.11}$$

Definition 4.12 (Origin-destination demand)**.** *The number of controllable vehicles that seek to travel from origin $o \in \mathcal{B}$ to destination $s \in \mathcal{S}$ at time step k is given by $D_{(o,s)}(k) \cdot \Delta t$. It is an exogenous input.*

Assumption 4.4 (Data requirements)**.** *We assume that the origin destination information for all the controllable flow and the aggregate path information for all the non-controllable flow is known.*

While at first glance this might seem like a lot of information to gather, it is in fact reasonable to assume in road traffic networks. We assume that the controllable flows are vehicles that are cooperating with the traffic coordination system that is trying to route vehicles efficiently and therefore will share their origin destination information. The aggregate paths of the non-controllable flows can be obtained from historical traffic patterns. The caveat is that empirical split ratios also include the contribution of the controllable flows and therefore must be pre-processed to remove this contribution.

Remark 4.2. *The junction split ratios for the non-controllable commodities may be time-dependent, while the junction split ratios for the controllable commodities are not, since a controllable commodity corresponds to a single path.*

Definition 4.13 (Equivalence between controllable commodity and path)**.** *Any controllable commodity $c \in \mathcal{CC}$ is a tuple (o, s, p) where $o \in \mathcal{B}$, $s \in \mathcal{S}$ and p is a path (i.e. a sequence of cells). We define the function Ω as follows:*

$$\begin{aligned} \Omega\colon \ \mathcal{CC} &\to \mathcal{S} \times \mathcal{B} \\ c &\mapsto (o, s). \end{aligned} \tag{4.12}$$

$\Omega^{-1}(o, s)$ is then the set of commodities corresponding to the flows from source o to destination s.

Definition 4.14 (Controllable flow control)**.** *A control u is an allocation of the controllable demand over the set $\Omega^{-1}(o, s)$ for each time step. Formally u is defined as:*

$$\begin{aligned} u\colon \ \mathcal{CC} \times [\![0, T-1]\!] &\to [0, 1] \\ (c, k) &\mapsto \gamma_c(k) \end{aligned} \tag{4.13}$$

$\gamma_c(k)$ is called the demand allocation for commodity c at time step k. The number of vehicles with origin o and destination s that are allocated to commodity c at time step k is $D_{(o,s)}(k) \cdot \gamma_c(k) \cdot \Delta t$.

Definition 4.15 (Physically feasible control)**.** *A physically feasible control u verifies the mass conservation of the controllable demand allocation:*

$$\sum_{c \in \Omega^{-1}(o,s)} \gamma_c(k) = 1 \qquad \forall k \in [\![0, T]\!],\ \forall (o,s) \in \mathcal{B} \times \mathcal{S} \tag{4.14}$$

We will denote with $\mathcal{U}$ be the set of all physically feasible controls.

4.3 Forward system

4.3.1 Junction model

The junction model defines the dynamics of the flow between neighboring cells. We require that is satisfies the following properties.

Requirement 4.4 (Multicommodity *first-in first-out* (FIFO) condition)**.** *For any outgoing link i, the distribution of its flow across the different commodities must be proportional to the ratio of vehicles of each commodity on the link. If $\rho_i(k) \neq 0$ we must have:*

$$f_{i,c}^{\text{out}}(k) = f_i^{\text{out}}(k) \frac{\rho_{i,c}(k)}{\rho_i(k)} \tag{4.15}$$

Requirement 4.5 (Consistency with split ratios)**.** *Let $f_{ij,c}(k)$ be the flow of commodity c from cell i to j at time step k. The outflow must be consistent with the split ratios.*

$$f_{ij,c}(k) = f_{i,c}^{\text{out}}(k) \cdot \beta_{ij,c}(k) \tag{4.16}$$

Requirement 4.6 (Maximum flow constraint)**.** *The outflow cannot exceed the demand and the inflow cannot exceed the supply:*

$$0 \leq f_i^{\text{in}}(k) \leq \sigma_i(k) \quad \forall k \in [\![0, T]\!] \tag{4.17}$$

$$0 \leq f_i^{\text{out}}(k) \leq \delta_i(k) \quad \forall k \in [\![0, T]\!] \tag{4.18}$$

We wish to define a multi-commodity junction flow solver that assigns flows across the network in a manner that is consistent with the above requirements.

Definition 4.16 (Priority vector)**.** *In the case of a junction where there is more than one incoming cell and the aggregate demand of these cells is greater than the aggregate supply of the outgoing cells, the available supply needs to be distributed among the competing demands according to some priority vector as follows. The priority vector P_j for cell j defines the allocation of its supply over the incoming cells $i \in \Gamma^{-1}(j)$. The priority for a given incoming cell i is given by P_{ij}.*

$$\sum_{i \in \Gamma^{-1}(j)} P_{ij} = 1 \tag{4.19}$$

There are many junction models that can satisfy our requirements, but vary in other ways. The multi-commodity junction solver we consider is similar to the *source destination model* (SDM) in [53]. We first give solutions for diverge only $(1 \times m)$ and merge only $(n \times 1)$ junctions, and then present the general $(n \times m)$ solution.

Definition 4.17 (Aggregate split ratio)**.** *The aggregate split ratio $\beta_{ij}(k)$ over all commodities for a given path through a junction is defined as follows:*

$$\begin{aligned} \beta_{ij}(k) &= \sum_{c \in \mathcal{C}} \frac{\rho_{i,c}(k)}{\rho_i(k)} \beta_{ij,c}(k) \\ &= \frac{1}{\rho_i(k)} \sum_{c \in \mathcal{C}} \rho_{i,c}(k) \beta_{ij,c}(k) \end{aligned} \tag{4.20}$$

Remark 4.3. *The aggregate split ratio is only defined for positive aggregate densities, i.e. $\rho_i(k) > 0$.*

Diverge solver $(1 \times m)$

We consider a diverging junction z with one incoming link i and m outgoing links. There are $|\mathcal{C}|$ commodities that flow through the junction each with their own time-varying split ratio $\beta_{ij,c}(k)$.

Remark 4.4. *If $\rho_i(k) = 0$, then $\delta_i(k) = 0$ and $f_i^{\text{out}}(k)$ is zero. We consider the case of $\rho_i(k) \neq 0$.*

Given the split ratios and densities of the cells at a junction, we wish to maximize the flow across the junction subject to the maximum flow constraints.

$$\begin{aligned} &\textbf{max } f_i^{\text{out}}(k) \\ &\textbf{subject to} \\ &\quad 0 \leq f_j^{\text{in}}(k) \leq \sigma_j(k) \qquad \forall j \in \mathcal{J}_z^{\text{out}} \\ &\quad 0 \leq f_i^{\text{out}}(k) \leq \delta_i(k) \end{aligned} \tag{4.21}$$

We replace $f_i^{\text{in}}(k)$ using the following relation:

$$\begin{aligned} f_j^{\text{in}}(k) &= \sum_{c \in \mathcal{C}} \beta_{ij,c}(k) \cdot f_{j,c}^{\text{out}}(k) && \text{[mass conservation]} \\ &= f_i^{\text{out}}(k) \sum_{c \in \mathcal{C}} \frac{\rho_{i,c}(k)}{\rho_i(k)} \cdot \beta_{ij,c}(k) && \text{[by the FIFO constraint]} \\ &= f_i^{\text{out}}(k) \cdot \beta_{ij}(k) && \text{[by definition of the aggregate split ratios]} \end{aligned} \tag{4.22}$$

Which gives us a trivial maximization problem that implies the following equality.

$$f_i^{\text{out}}(k) = \min\left(\left\{\frac{\sigma_j(k)}{\beta_{ij}(k)},\ \forall j \in \mathcal{J}_z^{\text{out}} \mid \beta_{ij}(k) > 0\right\}, \delta_i(k)\right) \tag{4.23}$$

The total outflow $f_i^{\text{out}}(k)$ for each incoming link i is then divided among the commodities according to the FIFO law:

$$f_{i,c}^{\text{out}}(k) = \frac{\rho_{i,c}(k)}{\rho_i(k)} f_i^{\text{out}}(k) \tag{4.24}$$

The commodity flows are split among the outgoing links according to the split ratios constraints:

$$f_{j,c}^{\text{in}}(k) = \beta_{ij,c}(k) f_{i,c}^{\text{out}}(k) \tag{4.25}$$

Existence and uniqueness of the solution. A non-zero solution exists if none of the constraints of the optimization/feasibility problem imposes a zero flow. In other words, as long as the demand is non-zero and none of the outgoing links with positive demand ($\beta_{ij}(k) > 0$) have non-zero supply, a non-zero solution exists. Since the solution to the maximum junction flow is given by equation (4.23) and the outflows are uniquely determined by the split ratios, the solution is unique.

Merge solver ($n \times 1$)

We consider a merging junction z with n incoming links and one outgoing link j. A priority vector P_j (s.t. $\sum P_{ij} = 1$) prescribes the priorities at which the outgoing link accepts flows from the n incoming links when the junction is supply constrained.

If the problem is demand constrained (i.e. $\sum_{i \in \mathcal{J}_z^{\text{in}}} \delta_i(k) < \sigma_j(k)$), then the solution is given by:

$$f_i^{\text{out}}(k) = \delta_i(k) \quad \forall i \in \mathcal{J}_z^{\text{in}} \tag{4.26}$$

Otherwise the problem is supply constrained and the solution to the junction problem is given by solving the following quadratic optimization problem that finds the flow-maximizing solution with the smallest violation of the priority vector, where the violation is measured using the L_2 distance:

$$\mathbf{min}_{t, \left\{f_i^{\text{out}}(k)\right\}_{i \in \mathcal{J}_z^{\text{in}}}} \sum_{i \in \mathcal{J}_z^{\text{in}}} \left(f_i^{\text{out}}(k) - t \cdot p_{ij}\right)^2 \tag{4.27}$$

subject to

$$\sum_{i \in \mathcal{J}_z^{\text{in}}} f_i^{\text{out}}(k) = \sigma_j(k)$$

$$0 \le f_i^{\text{out}}(k) \le \delta_i(k) \qquad \forall i \in \mathcal{J}_z^{\text{in}}$$

The total outflow $f_i^{\text{out}}(k)$ for each incoming link i is then divided among the commodities according to the FIFO law:

$$f_{i,c}^{\text{out}}(k) = \frac{\rho_{i,c}(k)}{\rho_i(k)} f_i^{\text{out}}(k) \tag{4.28}$$

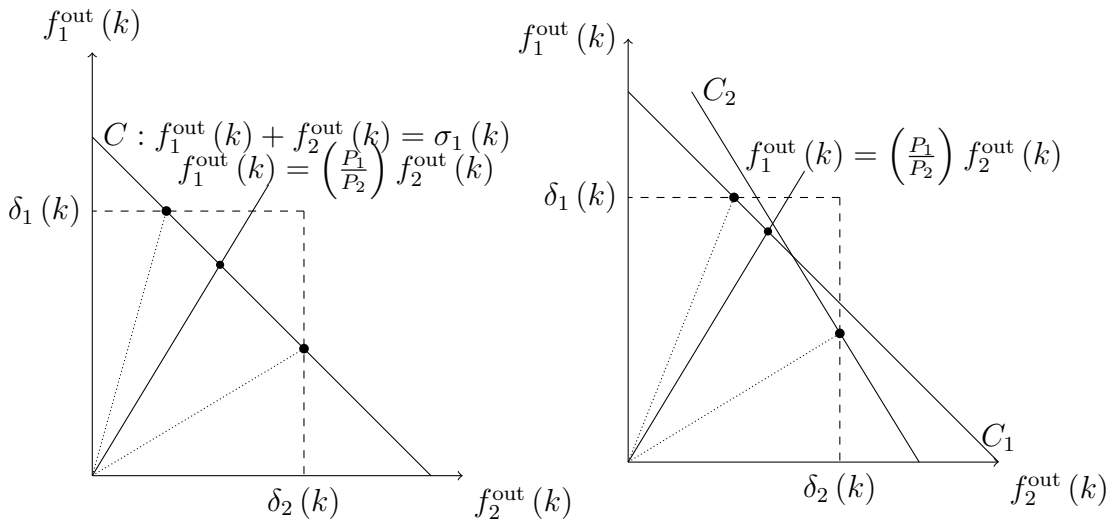
See figure 4.2a for a graphical illustration of the solution to a 2×1 junction.

Remark 4.5. *The priorities are satisfied exactly when the intersection of the maximum flow isoline and the priority constraint is feasible. When this point is outside the feasible set, the flow-maximizing feasible point that is closest to the priority constraint (in euclidean distance) is chosen.*

Remark 4.6. *The solution violates the priority rule only in the case where the demand for one or more of the incoming links is less than what its flow-maximizing allocation is, based on the priority vector. In other words, the priority rule is only violated when an incoming link doesn't have enough flow to satisfy its priority-based allocation. It is reasonable in the physical sense to maximize flow and only violate the priority when it is a lack of demand that causes the violation. The model is not denying any vehicles with priority the ability to pass through the junction. This is an important property to note, because it avoids having to solve a multi-objective optimization problem to come up with a physically meaningful set of flows through the junction.*

Existence and uniqueness of solution

- *Demand-constrained case*: In the demand constrained case, existence and uniqueness are trivial.
- *General case*: In the general case, the solution is the feasible point that lies on the boundary of the feasible supply set (a segment) and minimizes the euclidean distance to the priority vector (a line), where the feasible set is given by the supply constraint (an n-dimensional hyperplane: $\sum_{i \in \mathcal{J}_z^{\text{in}}} f_i^{\text{out}}(k) \leq \sigma_j(k)$) and demand constraints (an n-dimensional hyperrectangle: $f_i^{\text{out}}(k) \leq \delta_i(k),\ \forall i \in \mathcal{J}_z^{\text{in}}$).
  - A solution exists when the feasible set is non-empty, which is the case if the supply/demand constraints are greater than zero. This proves the existence of a solution in all non-degenerate (zero supply or demand) cases.
  - The boundary of the supply constraint hyperplane intersects each coordinate axis at $x_i(k) = \sigma_j(k)$ and can not be parallel to the priority constraint P, which is a line that goes through the origin. Therefore, since the solution must lie on a segment that is not parallel to the priority constraint line P, the solution that minimizes the distance to the P must be unique. This concludes the proof. See figure 4.2a for an illustration of the (2×1) case.



(a) 2×1 junction

(b) 2×2 junction

Figure 4.2: An illustration of the solutions to merging junctions. The dashed lines denote the demand constraints imposed by the density on the incoming links. The solid lines (C) denote the supply constraints imposed by the density in the outgoing links. The solid lines going through the origin denote the merge priority vector. If the priority vector intersects any supply constraint inside the feasible demand set, the solution will be the feasible intersection point. If the intersection is outside the feasible demand set, then the solution will be the nearest feasible point where the supply and the demand constraints intersect (marked with a dot).

Merge-diverge solver ($2 \times m$)

We consider a junction with 2 incoming links and m outgoing links[3].

Assumption 4.5. *The priority vectors P_j for each outgoing link j are identical. This implies that the inflow priorities are allocated with respect to the total flow that enters the junction and that the priority does not depend on which outgoing link the vehicles will enter.*

The priority vector P_j prescribes the ratios at which the m outgoing links allocate their available supply to the 2 incoming links. It satisfies $P_i = P_{i1} = P_{i2}$ and $\sum_{i \in \mathcal{J}_z^{\text{in}}} P_i = 1$.

Let $\mathcal{J}_z^{\text{in}}$ and $\mathcal{J}_z^{\text{out}}$ be the sets of incoming and outgoing links at the junction.

If the problem is demand-constrained (i.e. $\sum_{i \in \mathcal{J}_z^{\text{in}}} \beta_{ij}(k)\, \delta_i(k) \leq \sigma_j(k),\ \forall j \in \mathcal{J}_z^{\text{out}}$), then

[3]We limit our analysis to merge-diverge junctions of no more than two incoming links because our model does not prescribe a unique solution when the number of incoming links is greater then two. Thus, our model can only be used with $1 \times m, n \times 1$ and $2 \times m$ junctions.

the solution is given by:

$$f_i^{\text{out}}(k) = \delta_i(k) \quad \forall i \in \mathcal{J}_z^{\text{in}} \tag{4.29}$$

Otherwise, the flows through the junction are given by the following optimization problem.

$$\mathbf{min}_{t,\left\{f_i^{\text{out}}(k)\right\}_{i\in\mathcal{J}_z^{\text{in}}}} \sum_{i\in\mathcal{J}_z^{\text{in}}} \left(f_i^{\text{out}}(k) - t\cdot P_i\right)^2 \tag{4.30}$$

subject to

$$\sum_{i\in\mathcal{J}_z^{\text{in}}} \beta_{ij}(k) f_i^{\text{out}}(k) \le \sigma_j(k) \qquad \forall j \in \mathcal{J}_z^{\text{out}}$$

$$\mathbf{max}_j \left(\sum_{i\in\mathcal{J}_z^{\text{in}}} \beta_{ij}(k) f_i^{\text{out}}(k) - \sigma_j(k)\right) = 0 \qquad \forall j \in \mathcal{J}_z^{\text{out}}$$

$$f_i^{\text{out}}(k) \le \delta_i(k) \qquad \forall i \in \mathcal{J}_z^{\text{in}}$$

The total outflow $f_i^{\text{out}}(k)$ for each incoming link i is then divided among the commodities according to the FIFO law:

$$f_{i,c}^{\text{out}}(k) = \frac{\rho_{i,c}(k)}{\rho_i(k)} f_i^{\text{out}}(k) \tag{4.31}$$

The commodity flows are split among the outgoing links according to the split ratio constraints:

$$f_{j,c}^{\text{in}}(k) = \sum_{i:(i,j)\in A} \beta_{ij,c}(k) f_{i,c}^{\text{out}}(k) \tag{4.32}$$

See figure 4.2b for a graphical illustration of the solution to a 2×2 junction.

Existence and uniqueness of solution

- *Demand constrained case*: In the demand constrained case, existence and uniqueness are trivial.
- *General case*: In the general case, the solution is the feasible point (with respect to the supply and demand constraints) that lies on the boundary of the feasible supply set (a union of segments) and minimizes the euclidean distance to the priority vector (a line).
  - A solution exists when the feasible set is non-empty, which is the case if the supply/demand constraints are greater than zero. This proves the existence of a solution in all non-degenerate (zero supply or demand) cases.

– The feasible supply set is the intersection of m two dimensional hyperplanes, which is a convex set. Therefore, the boundary of the feasible supply set (a union of segments) is also convex. Furthermore, the boundary of the feasible supply set intersects each coordinate axis at $x_i(k) = \mathbf{min}_j \frac{\sigma_j(k)}{\beta_{ij}(k)}$ and therefore can not be parallel to the priority constraint P, which is a line that goes through the origin. Therefore, since the solution must lie on a convex union of segments and none of these segments is parallel to the priority constraint line P, the solution that minimizes the distance to P must be unique. This concludes the proof.

4.3.2 Boundary conditions

The boundary conditions at each source link of the network dictate the flows that enter the network. Each boundary condition is given as a flow rate at the boundary.

Definition 4.18 (Boundary demand)**.** *The number of vehicles of commodity c leaving cell $i \in \mathcal{B}$ at time step k is the boundary demand of commodity $d_{i,c}(k)$. Let c_{n} be the commodity corresponding to non-controllable flow. The non-zero terms are defined as:*

$$d_{i,c}(k) = \Delta t \cdot D_{i,c}(k) \qquad \forall i \in \mathcal{B},\ c = c_{\mathrm{n}} \qquad \text{(Non-controllable demand)}$$
$$d_{i,c}(k) = \Delta t \cdot D_{\Omega(c)}(k) \cdot \gamma_c(k) \qquad \forall i \in \mathcal{B},\ \forall c \in \mathcal{CC} \qquad \text{(Controllable demand)}$$

Since the inflow to the network is limited by the maximum flow capacity and density of the immediate downstream link, all of the demand at a given time step might not make it into the network. A source buffer is used to accumulate the flow that cannot enter the network to guarantee the conservation of boundary flows. In the single commodity case, this model is sufficient. However, in the multi commodity case, we also need to make sure that the flow through the boundary respects the multicommodity FIFO condition given in requirement 4.4.

As stated in Definition 4.18, $d_{i,c}(k)$ is the boundary demand per commodity on cell i at time step k. The FIFO condition dictates that the vehicles entering the boundary buffer at time k should enter link i at the ratio $\frac{d_{i,c}(k)}{\sum_{c \in C} d_{i,c}(k)}$ for each commodity c.

Single buffer model

The simplest solution is to have a single buffer l at the boundary, as in the single commodity case, and keep track of how many vehicles of each commodity are at the buffer. The flow into the boundary cell will be as follows:

$$f_{i,c}^{\mathrm{in}}(k) = \frac{l_{i,c}(k)}{l_i(k)} f_i^{\mathrm{in}}(k) \tag{4.33}$$

This equality satisfies the FIFO condition assuming that the vehicles in the buffer are uniformly distributed. However, in reality the buffer can accumulate vehicles arriving at the

boundary at different time steps with different commodity ratios $\frac{d_{i,c}(k)}{d_i(k)}$. Thus, this model can violate the FIFO property across multiple time steps if some vehicles cannot leave the buffer in one time step.

As the length of a buffer can be seen as the density of a cell of length 1, we use the same notation $\rho_{i,c}(k)$ for a buffer i. The speed of this cell is then $v_i = \frac{1}{\Delta t}$ because of requirements 4.1 and 4.3.

Remark 4.7 (The multicommodity FIFO condition is only satisfied approximately on the interior of the network.)**.** *It is important to note that the FIFO condition is violated in its strict sense even within the network. The flow propagation model assumes that all the flow within a cell is uniformly distributed according to the individual commodity ratios regardless of when the vehicles arrived at the cell.*

Example 4.1. *Consider the following simple example. There are two commodities a, b in cell i with 10 vehicles of each commodity at time k. At time k + 1, 10 vehicles exit the cell (5 of a and 5 of b by the FIFO rule) and 10 new vehicles (3 of a and 7 of b) enter the cell. The new ratio of vehicles at i is 8 a to 12 b. At time k + 2, once again 10 vehicles exit the network. According to the cell level FIFO rule, the 10 vehicles will consist of 4 a's and 6 b's. However, the first 10 cars of those currently in cell i came at the ratio of 1:1 and truly satisfying the FIFO rule would require the 10 exiting vehicles to consist of 5 a's and 5 b's.*

Remark 4.8 (Cell-level multicommodity FIFO condition)**.** *The strict multicommodity FIFO condition is not satisfied in most traffic flow models, but this is considered to be acceptable by the traffic modeling community. The multicommodity FIFO condition is therefore in practice at best limited to the cell level.*

While this argument is satisfactory at the interior of the network, due to the bounded number of vehicles in a given cell (due to the jam density), the FIFO violation can be significant at the boundaries of the network. The number of vehicles in a source buffer at the boundary of the network can be arbitrarily large, and thus, there is no bound on how badly the multicommodity FIFO constraint can be violated.

Multi-buffer model

A simple extension that limits the mixing of vehicles entering at different time steps is to have a series of source buffers at the boundary, where the multicommodity FIFO constraint is enforced when vehicles move between the buffers. In this model, the commodity ratios are maintained separately for each buffer. Any vehicles that enter the network at a given time step are added to the last active (non-empty buffer) buffer. This restricts the violation of the FIFO condition across multiple steps to the capacity of a single buffer.

The capacity of each buffer is chosen such that the buffer can satisfy the maximum supply of the first cell of the network boundary that the buffer serves. This prevents artificial delays at the source. The buffer capacity is set to $\Delta t \cdot F_b$ where b is the boundary cell.

The limitation of this model is that we need to maintain $\frac{l^{\max}}{\Delta t \cdot F_b}$ buffers per source, where $l^{\max}$ is the maximum queue length at the boundary and $\Delta t \cdot F_b$ is the capacity of each buffer. Let $l_{i,c}^b(k)$ be the number of vehicles of commodity c in the b^{th} buffer for source node i at time step k and $l_i^b(k)$ its sum over all the commodities.

The buffers are updated as follows:

- First, given $f_i^{\text{in}}(k)$, move flow out of the initial buffer

$$f_{i,c}^{\text{in}}(k) = \frac{l_{i,c}^1(k)}{l_i^1(k)} f_i^{\text{in}}(k) \qquad \forall c \in \mathcal{C} \tag{4.34}$$

$$l_{i,c}^1(k+1) = l_{i,c}^1(k) - f_{i,c}^{\text{in}}(k) \qquad \forall c \in \mathcal{C} \tag{4.35}$$

- Let B be the number of buffers in use. Iterate through the buffers and push flow upstream using the following algorithm.

Algorithm 4.1 Update buffers

for $b = 1$ to $B - 1$ **do**
  $\Delta l^b = \min\left(L - l_i^b(k+1),\ l_i^{b+1}(k+1)\right)$ ▷ Maximum flow that can enter buffer b
  $l_{i,c}^b(k+1) = l_{i,c}^b(k+1) + \frac{l_{i,c}^{b+1}(k+1)}{l_i^{b+1}(k+1)} \Delta l^b \quad \forall c \in C$ ▷ Per-commodity flow entering buffer b
  $l_{i,c}^{b+1}(k+1) = l_{i,c}^{b+1}(k+1) - \frac{l_{i,c}^{b+1}(k+1)}{l_i^{b+1}(k+1)} \Delta l^b \quad \forall c \in C$ ▷ Per-commodity flow leaving buffer $b+1$
end for
$\Delta d = d_i(k)$ ▷ Total demand at time step $k+1$
$b = B$
▷ Allocate the demand to last buffer and create new buffers if needed
while $\Delta d > 0$ **do**
  $\Delta l^b = \min\left(L - l_i^b(k+1),\ \Delta d\right)$ ▷ Maximum flow that can enter buffer b
  $l_{i,c}^b(k+1) = l_{i,c}^b(k+1) + \frac{d_{i,c}(k+1)}{d_i(k+1)} \Delta l^b \quad \forall c \in C$ ▷ Per-commodity flow entering buffer b
  $b = b + 1$
end while

The only demand that is exposed to the system dynamics and the optimization problem is the demand that is captured in the first buffer.

4.3.3 System dynamics

For a given control u, we can determine the evolution of the network using the following equations that prescribe the system dynamics. Let $x(u)$ give the state of the network under

these dynamics subject to the control u.

The system of equations governing the evolution of the network (implicit definition of x) are written formally in the form $H(x,u)=0$, thus x is an implicit function of u. The discretized system dynamics can be described using six types of constraints, given by $H^h=0, h\in\{1,\dots,6\}$, listed below.

H^1: Mass conservation and boundary conditions
H^2: Demand constraints
H^3: Supply constraints
H^4: Aggregate split ratios
H^5: Flow out of junctions
H^6: Flow in to junctions

These six constraints have different individual instantiations depending on the specific setting such as link type or junction type. The explicit formulation is given below.

Mass conservation

$$H^1_{k,i,c}: \quad \rho_{i,c}(k)=\rho_{i,c}(k-1)+\frac{\Delta t}{L_i}\left(f^{\text{in}}_{i,c}(k-1)-f^{\text{out}}_{i,c}(k-1)\right) \quad \forall i\in\mathcal{A}\setminus(\mathcal{B}\cup\mathcal{S}),\ \forall k\in[\![1,T_f]\!],\ \forall c\in\mathcal{C} \tag{H1a}$$

$$H^1_{k,i,c}: \quad \rho_{i,c}(k)=\rho_{i,c}(k-1)+\frac{\Delta t}{L_i}\cdot f^{\text{in}}_{i,c}(k-1) \quad \forall i\in\mathcal{S},\ \forall k\in[\![1,T_f]\!],\ \forall c\in\mathcal{C} \tag{H1b}$$

with initial conditions

$$H^1_{0,i,c}: \quad \rho_{i,c}(0)=\rho^0_{i,c} \quad \forall i\in\mathcal{A}\setminus\mathcal{S},\ \forall c\in\mathcal{C} \tag{I1a}$$

$$H^1_{0,i,c}: \quad \rho_{i,c}(0)=0 \quad \forall i\in\mathcal{S},\ \forall c\in\mathcal{C} \tag{I1b}$$

Boundary conditions[4]

$$H^1_{k,i,c}: \quad \rho_{i,c}(k)=\rho_{i,c}(k-1)+\frac{\Delta t}{L_i}\left(D_{\Omega(c)}(k)\cdot\gamma_c(k)-f^{\text{out}}_{i,c}(k-1)\right) \quad \forall i\in\mathcal{B},\ \forall k\in[\![1,T_f]\!],\ \forall c\in\mathcal{CC} \tag{H1c}$$

$$H^1_{k,i,c}: \quad \rho_{i,c}(k)=\rho_{i,c}(k-1)+\frac{\Delta t}{L_i}\left(D_{i,c}(k)-f^{\text{out}}_{i,c}(k-1)\right) \quad \forall i\in\mathcal{B},\ \forall k\in[\![1,T_f]\!],\ c=c_n \tag{H1d}$$

with initial conditions

$$H^1_{0,i,c}: \quad \rho_{i,c}(0)=\rho^0_{i,c}+\frac{\Delta t}{L_i}\cdot D_{\Omega(c)}(0)\cdot\gamma_c(k) \quad \forall i\in\mathcal{B},\ \forall c\in\mathcal{CC} \tag{I2a}$$

$$H^1_{0,i,c}: \quad \rho_{i,c}(0)=\rho^0_{i,c}+\frac{\Delta t}{L_i}\cdot D_{i,c}(0) \quad \forall i\in\mathcal{B},\ c=c_n \tag{I2b}$$

[4]For notational simplicity, the equations given here are for the single buffer model.

Flow propagation

Recall that $\rho_i(k) = \sum_{c=1}^{C} \rho_{i,c}(k)$ is the total density of cell i at time step k.

$$H^2_{k,i}: \qquad \delta_i(k) = \min(F_i, v_i \rho_i(k)) \qquad \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \tag{H2a}$$

$$H^2_{k,i}: \qquad \delta_i(k) = \min\left(F_i, \frac{\rho_i(k) L_i}{\Delta t}\right) \qquad \forall i \in \mathcal{B},\ \forall k \in [\![0, T_f]\!] \tag{H2b}$$

$$H^3_{k,i}: \qquad \sigma_i(k) = \min\left(F_i, w_i\left(\rho_i^{\text{jam}} - \rho_i(k)\right)\right) \qquad \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \tag{H3a}$$

$$H^3_{k,i}: \qquad \sigma_i(k) = F_i \qquad \forall i \in \mathcal{S},\ \forall k \in [\![0, T_f]\!] \tag{H3b}$$

Remark 4.9. *The sinks having no outgoing cells and the buffers having no incoming cells, their demand and supply respectively are not of any use. We can arbitrarily choose to set them to zero.*

Junction solution

The derivation of the explicit solutions to the $1 \times 2, 2 \times 1$ and 2×2 junctions listed below are given in the next section. The general solutions for the junction model are given in section 4.3.1. We restrict our attention to these junctions since most road networks can be modeled using only these junction types.

To simplify the notation, we use the following shorthand:

- We drop the time index k
- We abbreviate $i_1 = 1$ and $i_2 = 2$
- We use the following notation $\underline{i_1} = i_2$ and $\underline{i_2} = i_1$

When $\rho_i = 0$, there are no vehicles in the incoming link i and the outflow of this link is zero for all commodities. The following equations only apply for $\rho_i \neq 0$.

Aggregate split ratios:

$$H^4_{k,i,j,z}: \qquad \beta_{ij} = \frac{1}{\rho_i} \sum_{c \in \mathcal{C}} \rho_{i,c} \beta_{ij,c} \qquad \forall z \in \mathcal{J},\ \forall i \in \mathcal{J}_z^{\text{in}},\ \forall j \in \mathcal{J}_z^{\text{out}},\ \forall k \in [\![0, T_f]\!] \tag{H4}$$

Flow out of incoming links by commodity:

$$H^5_{k,i,c}: \quad f^{\text{out}}_{i,c} = \frac{\rho_{i,c}}{\rho_i} \min\left(\left\{ \frac{\sigma_j}{\beta_{ij}},\ \forall j \in \mathcal{J}^{\text{out}}_z \mid \beta_{ij} > 0 \right\}, \delta_i \right) \quad \forall z \in \mathcal{J}_{1\times n},\ \forall i \in \mathcal{J}^{\text{in}}_z \tag{H5a}$$

$$H^5_{k,i,c}: \quad f^{\text{out}}_{i,c} = \frac{\rho_{i,c}}{\rho_i} \begin{cases} \delta_i & \text{if } P_i \left(\min\left(\delta_1+\delta_2, \sigma_1\right) - \delta_i\right) > \delta_i P_{\underline{i}} \\ \min\left(\delta_1+\delta_2, \sigma_1\right) - \delta_{\underline{i}} & \text{if } P_{\underline{i}} \left(\min\left(\delta_1+\delta_2, \sigma_1\right) - \delta_{\underline{i}}\right) > \delta_{\underline{i}} P_i \\ P_i \min\left(\delta_1+\delta_2, \sigma_1\right) & \text{otherwise} \end{cases} \quad \forall z \in \mathcal{J}_{2\times 1},\ \forall i \in \mathcal{J}^{\text{in}}_z \tag{H5b}$$

$$H^5_{k,i,c}: \quad f^{\text{out}}_1 = \begin{cases} \delta_1 & \text{if } \frac{P_1}{P_2} > \frac{\delta_1}{\min\left(\delta_2, \frac{\sigma_1-\beta_{11}\delta_1}{\beta_{21}}, \frac{\sigma_2-\beta_{12}\delta_1}{\beta_{22}}\right)} \\ \min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right) & \text{if } \frac{P_1}{P_2} < \frac{\min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right)}{\delta_2} \\ \min\left(\frac{P_1\sigma_1}{P_1\beta_{11}+P_2\beta_{21}}, \frac{P_1\sigma_2}{P_1\beta_{12}+P_2\beta_{22}}\right) & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{J}^{in}_{2\times 2} \tag{4.39a}$$

$$f^{\text{out}}_2 \text{ is obtained by symmtery.} \tag{4.39b}$$

$$f^{\text{out}}_{i,c} = \frac{\rho_{i,k}}{\rho_i} f^{\text{out}}_i \quad \forall i \in \{1,2\},\ \forall c \in C \tag{H5c}$$

Flow into outgoing links by commodity:

$$H^6_{k,i,c}: \qquad f^{\text{in}}_{i,c} = \sum_{x \in \mathcal{J}^{\text{in}}_z} \beta_{xi,c} f^{\text{out}}_{x,c} \quad \forall z \in \mathcal{J},\ \forall i \in \mathcal{J}^{\text{out}}_z,\ \forall c \in \mathcal{C} \tag{H6}$$

4.3.4 Explicit solutions for the junction flows

We now present the explicit solutions used for the dynamics of the optimization problem and in the software implementation.

(1×2) diverge junction

The explicit solution was already derived in section 4.3.1 for the general $(1 \times n)$ case.

(2×1) merge

Let i_1 be one incoming link, i_2 be the other incoming link and j be the outgoing link. As shown in Figure 4.2a, we distinguish 3 cases based on where the priority vector intersects the demand constraints. The cases are $\frac{P_1}{P_2} > \left(\frac{P_1}{P_2}\right)_{\max}$, $\left(\frac{P_1}{P_2}\right)_{\min} \le \frac{P_1}{P_2} \le \left(\frac{P_1}{P_2}\right)_{\max}$ and $\frac{P_1}{P_2} < \left(\frac{P_1}{P_2}\right)_{\min}$, where $\left(\frac{P_1}{P_2}\right)_{\max} = \frac{\delta_{i_1}}{f^{\text{in}}_j - \delta_{i_1}}$ and $\left(\frac{P_1}{P_2}\right)_{\min} = \frac{f^{\text{in}}_j - \delta_{i_2}}{\delta_{i_2}}$.

In call cases, the flow into the outgoing link is the minimum of the total supply and demand values.

$$f^{\text{in}}_j(k) = \min\left(\delta_{i_1}(k) + \delta_{i_2}(k), \sigma_j(k)\right) \tag{4.40}$$

The flow out of incoming links depends on the priority vector and is given by the solution to the optimization problem in equation (4.27).

$$f_{i_1}^{\text{out}}(k) = \begin{cases} \delta_{i_1}(k) & \text{if } P_{i_1}\left(f_j^{\text{in}}(k) - \delta_{i_1}(k)\right) > \delta_{i_1}(k) P_{i_2} \\ f_j^{\text{in}}(k) - \delta_{i_2}(k) & \text{if } P_{i_2}\left(f_j^{\text{in}}(k) - \delta_{i_2}(k)\right) > \delta_{i_2}(k) P_{i_1} \\ P_{i_1} f_j^{\text{in}}(k) & \text{otherwise} \end{cases}$$

$$f_{i_2}^{\text{out}}(k) = f_j^{\text{in}}(k) - f_{i_1}^{\text{out}}(k) \tag{4.41}$$

The flow out by commodity can then be computed as follows.

$$f_{i,c}^{\text{out}}(k) = \frac{\rho_{i,c}(k)}{\rho_i(k)} f_i^{\text{out}}(k) \qquad \forall i \in \{i_1, i_2\} \tag{4.42}$$

(2×2) merge and diverge

Let i_1, i_2 be the incoming links and j_1, j_2 be the outgoing links. To simplify the notation, we use the following shorthand:

- drop the time index k
- $\delta_1 = \delta_{i_1}$, $\delta_2 = \delta_{i_2}$
- $\sigma_1 = \sigma_{j_1}$, $\sigma_1 = \sigma_{j_2}$
- $P_1 = P_{i_1}$, $P_2 = P_{i_2}$

The aggregate split ratios are computed as follows.

$$\beta_{ij}(k) = \frac{1}{\rho_i(k)} \sum_{c=1}^{C} \rho_{i,c}(k) \beta_{ij,c}(k) \qquad \forall (i,j) \in \{1,2\} \times \{1,2\} \tag{4.43}$$

As shown in Figure 4.2b, we once again distinguish 3 cases based on where the priority vector intersects the demand constraints. The cases are $\frac{P_1}{P_2} > \left(\frac{P_1}{P_2}\right)_{\max}$, $\left(\frac{P_1}{P_2}\right)_{\min} \leq \frac{P_1}{P_2} \leq \left(\frac{P_1}{P_2}\right)_{\max}$ and $\frac{P_1}{P_2} < \left(\frac{P_1}{P_2}\right)_{\min}$.

The values of $\left(\frac{P_1}{P_2}\right)_{\max}$ (and $\left(\frac{P_1}{P_2}\right)_{\min}$) can be obtained by plugging substituting δ_1 (and resp. δ_2) for f_1^{out} (and resp. f_1^{out}) in the equations C_j for the supply constraints.

$$C_j : \beta_{1j} f_1^{\text{out}} + \beta_{2j} f_2^{\text{out}} = \sigma_j \tag{4.44}$$

Thus, we obtain:

$$\left(\frac{P_1}{P_2}\right)_{\max} = \frac{\delta_1}{\min\left(\delta_2, \frac{\sigma_1-\beta_{11}\delta_1}{\beta_{21}}, \frac{\sigma_2-\beta_{12}\delta_1}{\beta_{22}}\right)} \tag{4.45}$$

$$\left(\frac{P_1}{P_2}\right)_{\min} = \frac{\min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right)}{\delta_2} \tag{4.46}$$

We can then compute the flow out of incoming links using the optimization problem in equation (4.30). Since the problem is symmetric in i_1 and i_2, we just solve it for $i_1 = 1$.

- If $\frac{P_1}{P_2} > \left(\frac{P_1}{P_2}\right)_{\max}$, the solution is the intersection of $f_1^{\text{out}} = \delta_1$ and the most constraining supply constraint (C_1 in figure 4.2b), and results in the trivial solution of $f_1^{\text{out}} = \delta_1$.
- If $\frac{P_1}{P_2} < \left(\frac{P_1}{P_2}\right)_{\min}$, the solution is the intersection of $f_2^{\text{out}} = \delta_2$ and the most constraining supply constraint (C_2 in figure 4.2b), and therefore we obtain:
  $f_1^{\text{out}} = \min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right)$
- If $\left(\frac{P_1}{P_2}\right)_{\min} \leq \frac{P_1}{P_2} \leq \left(\frac{P_1}{P_2}\right)_{\max}$, the solution lies at the point where the priority vector intersects the most constraining supply constraint ((C_2 in figure 4.2b)), and therefore we obtain:
  $f_1^{\text{out}} = \min\left(\frac{P_1\sigma_1}{P_1\beta_{11}+P_2\beta_{21}}, \frac{P_1\sigma_2}{P_1\beta_{12}+P_2\beta_{22}}\right)$

This gives us the explicit following explicit solution for f_1^{out} at a 2×2 junction.

$$f_1^{\text{out}} = \begin{cases} \delta_1 & \text{if } \frac{P_1}{P_2} > \frac{\delta_1}{\min\left(\delta_2, \frac{\sigma_1-\beta_{11}\delta_1}{\beta_{21}}, \frac{\sigma_2-\beta_{12}\delta_1}{\beta_{22}}\right)} \\ \min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right) & \text{if } \frac{P_1}{P_2} < \frac{\min\left(\delta_1, \frac{\sigma_1-\beta_{21}\delta_2}{\beta_{11}}, \frac{\sigma_2-\beta_{22}\delta_2}{\beta_{12}}\right)}{\delta_2} \\ \min\left(\frac{P_1\sigma_1}{P_1\beta_{11}+P_2\beta_{21}}, \frac{P_1\sigma_2}{P_1\beta_{12}+P_2\beta_{22}}\right) & \text{otherwise} \end{cases} \tag{4.47}$$

f_2^{out} is obtained by symmetry.

Finally, we can now compute the flow out of incoming links by commodity.

$$f_{i,c}^{\text{out}} = \frac{\rho_{i,k}}{\rho_i} f_i^{\text{out}} \quad \forall i \in \{1,2\},\ \forall c \in C \tag{4.48}$$

As shown in figure 4.3, the dynamics equations have a topological ordering that allows for the efficient forward simulation given in algorithm 4.2:

Remark 4.10. *For any given time step k, each of the three internal loops in algorithm 4.2 are trivially parallelizable problems.*

Algorithm 4.2 Computes the state vector $x(u)$

for $i \in \mathcal{A}$ **do**
  $\rho_{i,c}(0) = \rho^0_{i,c}$
end for
for $k = 0 \to T-1$ **do**
  for all $i \in \mathcal{A}$ **do**
    Compute $\delta_i(k)$ ▷ by equations H^2
    Compute $\sigma_i(k)$ ▷ by equations H^3
    Compute $\beta_{ij}(k)$ ▷ by equations H^4
  end for
  for all $z \in \mathcal{J}$ **do**
    Compute $\{f^{\text{out}}_{i,c}(k)\}_{c\in\mathcal{C},\ i\in\mathcal{J}^{\text{in}}_z}$ ▷ by equations H^5
    Compute $\{f^{\text{in}}_{i,c}(k)\}_{c\in\mathcal{C},\ i\in\mathcal{J}^{\text{out}}_z}$ ▷ by equations H^6
  end for
  for all $i \in \mathcal{A}$ **do**
    Compute $\{\rho_{i,c}(k+1)\}_{c\in\mathcal{C}}$ ▷ by equations H^1
  end for
end for

4.4 Adjoint based optimization

This section describes the adjoint based optimization framework for efficiently computing the gradient of the forward system. We formulate the optimization problem first and then present a detailed description of how the discret adjoint method can be applied to this problem.

4.4.1 Problem formulation

For a given control u, we can determine the evolution of the network using the equations for the system dynamics. Let $x(u)$ be the state of the network under these dynamics subject to the control u.

The total travel-time $J(x(u))$ is defined as:

$$J = \sum_{k=0}^{T-1} \sum_{i\in\mathcal{A}\setminus\mathcal{S}} \rho_i(k) \cdot L_i \tag{4.49}$$

The system optimal dynamic traffic assignment with partial compliance (SO-DTA-PC) is a physically acceptable (see Definition 4.14) division of the compliant agents among the different commodities that minimizes the total travel-time (including the travel-time of the non-compliant commodities). The solution is obtained by solving the following optimization problem.

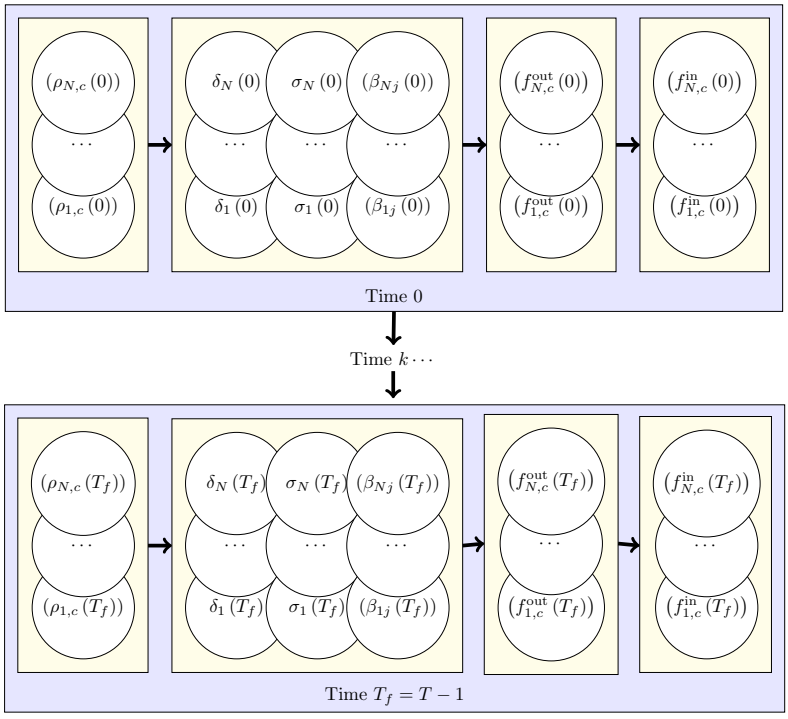


Figure 4.3: Dependency diagram of the variables in the system.

$$
\begin{aligned}
&\mathbf{min}_{u\in\mathcal{U}}\, J(x(u))\\
&\textbf{subject to}\\
&\quad \textit{system dynamics}\\
&\quad \textit{control constraints}
\end{aligned}
$$

where the system dynamics are given in Section 4.3.3 and the control constraints are the following.

$$
\begin{aligned}
&\gamma_c(k) \geq 0 && \forall c \in \mathcal{CC}, k \in [\![0, T_f]\!]\\
&\sum_{c\in\Omega^{-1}\{(o,s)\}} \gamma_c(k) = 1 && \forall k \in [\![0, T_f]\!]
\end{aligned}
$$

Note that this is a non-convex optimization problem that might contain multiple local minima. Therefore, gradient methods will not guarantee global optimality. However, descent algorithms can still be used to obtain locally optimal solutions and can be improved by using multiple starting points [18, 89]. Furthermore, non-convex optimization techniques such as

subgradient and interior point methods [132] require the gradient of the system. We use the discrete adjoint method, which will be explained in the next section, to efficiently solve for the gradient of the system. The control constraints can be satisfied either using a projected gradient descent or a barrier function. In our implementation, we use the projected gradient descent approach.

4.4.2 Overview of the adjoint method

We consider the following general optimization problem:

$$\begin{aligned} &\mathbf{min}_{u\in\mathcal{U}} && J(x,u) \\ &\mathbf{subject\ to} && H(x,u)=0 \end{aligned} \tag{4.50}$$

where $x \in \mathcal{X}$ denotes the state variables and $u \in \mathcal{U}$ denotes the control variables.

The adjoint method [46] is a technique to compute the gradient $\nabla_u J(x,u) = \frac{dJ}{du}$ of the objective function without fully computing $\nabla_u x = \frac{dx}{du}$. The gradient is then used to perform a gradient descent. We suppose that for any control u, $\frac{\partial H}{\partial x}(x,u)$ is not singular.

Under equality constraints $H(x,u)=0$, the Lagrangian

$$L(x,u,\lambda) = J(x,u) + \lambda^T H(x,u) \tag{4.51}$$

coincides with the objective function for any feasible point $(x(u),u)$. The problem is then equivalent to computing the gradient of the Lagrangian:

$$\begin{aligned} \nabla_u L(x,u,\lambda) &= \frac{\partial J}{\partial u} + \frac{\partial J}{\partial x}\frac{dx}{du} + \lambda^T\left(\frac{\partial H}{\partial u} + \frac{\partial H}{\partial x}\frac{dx}{du}\right) \\ &= \frac{\partial J}{\partial u} + \lambda^T\frac{\partial H}{\partial u} + \left(\frac{\partial J}{\partial x} + \lambda^T\frac{\partial H}{\partial x}\right)\frac{dx}{du} \end{aligned} \tag{4.52}$$

In particular, if λ satisfies the adjoint equation:

$$\frac{\partial J}{\partial x} + \lambda^T\frac{\partial H}{\partial x} = 0 \tag{4.53}$$

then the gradient is,

$$\nabla_u L(x,u) = \frac{\partial J}{\partial u} + \lambda^T\frac{\partial H}{\partial u} \tag{4.54}$$

Remark 4.11. *The solution for λ exists and is unique if $\frac{\partial H}{\partial x}$ is not singular, which is the case in our forward system, as explained in the following section.*

4.4.3 Applying the adjoint method

To be able to use the adjoint method to compute the gradient, the partial derivatives of the forward system with respect to the state variables $\frac{\partial H}{\partial x}$ must not be singular. We can rewrite our system of equations in the form $H(x,u)=0$ and verify this condition trivially.

All the diagonal terms of $\frac{\partial H}{\partial x}$ are non zero (since equal to 1 or -1 depending on the way we rewrite H_v). As seen in the dependency chain shown in Figure 4.3, the non zero derivative terms of H_v depend only on variables that have a smaller index in x. This means that $\frac{\partial H}{\partial x}$ is lower triangular with no zero terms on the diagonal and is thus non singular. Therefore, we can apply the adjoint method to compute the gradient of this system.

Reduced state space. The forward system dynamics that were described in section 4.3.3 had a large number of state variables. However, the only required state variables of the system are the partial densities $\rho_{i,c}(k)$. All the others variables were introduced to make the forward system easier to understand. We will now drop most of these auxiliary variables to simplify the computation of the adjoint system. We only use $\rho_{i,c}(k)$, $f^{\text{out}}_{i,c}(k)$ and $f^{\text{in}}_{i,c}(k)$ to describe the system and replace the other variables by their expressions as a function of the three state variables that we retain.

We define:

$$x = \begin{pmatrix} \left(\left(\rho_{i,c}(k)\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \\ \left(\left(f^{\text{out}}_{i,c}(k)\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \\ \left(\left(f^{\text{in}}_{i,c}(k)\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \end{pmatrix}_{k\in[\![0,T]\!]} \qquad H = \begin{pmatrix} \left(\left(H^1_{k,i,c}\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \\ \left(\left(H^5_{k,i,c}\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \\ \left(\left(H^6_{k,i,c}\right)_{c\in\mathcal{C}}\right)_{i\in\mathcal{A}} \end{pmatrix}_{k\in[\![0,T]\!]}$$

Computational complexity. Let n be the dimension of the state vector $x\in R^n$, m be the dimension of the control vector is $u\in R^m$ and $N_c=|\mathcal{C}|$ be the total number of commodities. From the above definition of the state vector, we can see that $n=|\mathcal{A}|\cdot T\cdot N_c$. The dimension of $\mathcal{H}$ is also n as defined above.

Direct computation of the gradient $\nabla_u J(x,u)$ takes $O(n^2m)$ time.

$$\nabla_u J(x,u) = \frac{\partial J}{\partial x}\cdot\nabla_u x + \frac{\partial J}{\partial u} \tag{4.55}$$

Computing $\nabla_u J$ requires solving the system $H(x,u)=0 \Rightarrow \frac{\partial H}{\partial x}\frac{dx}{du}+\frac{\partial H}{\partial u}=0$, which is equivalent to solving m different $n\times n$ linear systems and takes $O(n^2m)$ time. The final step of multiplying $\frac{\partial J}{\partial x}\frac{dx}{du}$ and adding $\frac{\partial J}{\partial u}$ takes $O(nm)$ time, but is dominated by the time to compute $\frac{dx}{du}$.

The discrete adjoint method reduces this complexity to $O(n^2+nm)$ by solving for λ in the adjoint system. Computing the adjoint variables $\lambda^T\in R^n$ using equation (4.53) only takes $O(n^2)$ time since it only requires solving one $n\times n$ linear system. Multiplying $\lambda^T\frac{\partial H}{\partial u}$

and adding $\frac{\partial J}{\partial u}$ to complete the computation in equation (4.54) takes $O(nm)$ time, so the total computation time is $O(n^2 + nm)$.

The structure of our system allows for further reduction of the complexity to $O(n+m|\mathcal{C}|)$. As shown in section 4.4.3, $\frac{\partial H}{\partial x}$ is a lower triangular matrix and therefore we can compute the solution to equation (4.53) using backwards substitution. We will exploit the fact that the matrix $\frac{\partial H}{\partial x}$ is extremely sparse. The maximum row cardinality is four because the forward system does not contain any constraints with more than four variables. Therefore, equation (4.53) can be solved in $O(n)$ time. If the maximum in-degree of the network is d_{in}, the maximum column cardinality is $2 + |\mathcal{C}|(1 + d_{\text{in}})$, as will be clear in the next section from equation (4.59). Assuming that d_{in} is a small constant, the multiplication step in equation (4.54) takes $O(m|\mathcal{C}|)$ time. This leads to a total computation time of $O(n + m|\mathcal{C}|)$.

4.4.4 Adjoint equations

The adjoint equations are given by the system:

$$\frac{\partial J}{\partial x} + \lambda^T \frac{\partial H}{\partial x} = 0 \tag{4.56}$$

$$\Rightarrow \frac{\partial J}{\partial x} + \sum_{x' \in x} \lambda'_x \frac{\partial H_{x'}}{\partial x} = 0 \tag{4.57}$$

where x is the state vector and H'_x is the forward system equation corresponding to the variable $x' \in x$. To write the adjoint system equation corresponding to x' , we have to look at all the forward system equations where x' appears and consider all the non-null $\frac{\partial H_{x'}}{\partial x}$ terms. In particular we write the equations such that $\frac{\partial H_{x'}}{\partial x'} = -1$. Note that this can be done because the Godunov scheme provides an explicit expression for the forward system constraints.

Computing $\frac{\partial J}{\partial x}$

$$\frac{\partial J}{\partial \rho_{i,c}(k)} = \begin{cases} L_i & \forall c \in \mathcal{C},\ \forall i \in \mathcal{A} \setminus \mathcal{S},\ \forall k \in [\![0, T]\!] \\ 0 & \text{otherwise} \end{cases} \tag{4.58}$$

Computing $\boldsymbol{\lambda^T \frac{\partial H}{\partial x}}$

$$\frac{\partial H}{\partial \rho_{i,c}(k)}: \quad \sum_{x' \in x} \lambda_{x'} \frac{\partial H_{x'}}{\partial \rho_{i,c}(k)} = \lambda_{\rho_{i,c}(k)} \frac{\partial H_{\rho_{i,c}(k)}}{\partial \rho_{i,c}(k)} + \lambda_{\rho_{i,c}(k+1)} \frac{\partial H_{\rho_{i,c}(k+1)}}{\partial \rho_{i,c}(k)} + \sum_{c' \in \mathcal{CC}} \left(\lambda_{f^{\text{out}}_{i,c'}(k)} \frac{\partial H_{f^{\text{out}}_{i,c'}(k)}}{\partial \rho_{i,c}(k)} + \lambda_{f^{\text{out}}_{x,c'}(k)} \sum_{x:(x,i)\in\mathcal{A}} \frac{\partial H_{f^{\text{out}}_{x,c'}(k)}}{\partial \rho_{i,c}(k)} \right) \tag{4.59}$$

$$\frac{\partial H}{\partial f^{\text{out}}_{i,c}(k)}: \quad \sum_{x' \in x} \lambda_{x'} \frac{\partial H_{x'}}{\partial f^{\text{out}}_{i,c}(k)} = \lambda_{\rho_{i,c}(k+1)} \frac{\partial H_{\rho_{i,c}(k+1)}}{\partial f^{\text{out}}_{i,c}(k)} + \lambda_{f^{\text{out}}_{i,c}(k)} \frac{\partial H_{f^{\text{out}}_{i,c}(k)}}{\partial f^{\text{out}}_{i,c}(k)} + \sum_{j:j\in(i,j)} \lambda_{f^{\text{in}}_{j,c}(k)} \frac{\partial H_{f^{\text{in}}_{j,c}(k)}}{\partial f^{\text{out}}_{i,c}(k)} \tag{4.60}$$

$$\frac{\partial H}{\partial f^{\text{in}}_{i,c}(k)}: \quad \sum_{x' \in x} \lambda_{x'} \frac{\partial H_{x'}}{\partial f^{\text{in}}_{i,c}(k)} = \lambda_{\rho_{i,c}(k+1)} \frac{\partial H_{\rho_{i,c}(k+1)}}{\partial f^{\text{in}}_{i,c}(k)} + \lambda_{f^{\text{in}}_{i,c}(k)} \frac{\partial H_{f^{\text{in}}_{i,c}(k)}}{\partial f^{\text{in}}_{i,c}(k)} \tag{4.61}$$

The next section shows how to compute all the individual partial derivatives that are required in the above equations. Once they are computed, we can simply plug them into the above equations and solve the system via backwards substitution since $\frac{\partial H}{\partial x}$ is lower triangular.

4.4.5 Partial derivatives

Computing the gradient of the system via the adjoint method requires computing the partial derivatives $\frac{\partial J}{\partial u}$, $\frac{\partial J}{\partial x}$, $\frac{\partial H}{\partial u}$ and $\frac{\partial H}{\partial x}$. The first three of these can be computed trivially.

- Partial derivatives of the cost function with respect to the control ($\boldsymbol{\frac{\partial J}{\partial u}}$) from equation (4.49)

$$\frac{\partial J}{\partial \gamma_c(k)} = 0 \tag{4.62}$$

- Partial derivatives of the cost function with respect to the state variables ($\boldsymbol{\frac{\partial J}{\partial x}}$) from equation (4.49)

$$\frac{\partial J}{\partial \rho_{i,c}(k)} = \begin{cases} L_i & \text{if } c \in \mathcal{C},\ i \in \mathcal{A} \setminus \mathcal{S},\ k \in [\![0, T]\!] \\ 0 & \text{otherwise} \end{cases} \tag{4.63}$$

- Partial derivatives of the constraints with respect to the state variables ($\boldsymbol{\frac{\partial H}{\partial u}}$) from equation (H1c)

$$\frac{\partial \rho_{i,c}(k)}{\partial \gamma_c(k)} = \begin{cases} \frac{\Delta t}{L_i} \cdot D_{\Omega(c)}(k) & \text{if } c \in \mathcal{CC},\ i \in \mathcal{B},\ k \in [\![0, T]\!] \\ 0 & \text{otherwise} \end{cases} \tag{4.64}$$

Computing the partial derivatives of the constraints with respect to the state variables ($\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{x}}$)

We first iterate through the three classes of variables. All unlisted derivatives evaluate to zero.

Commodity density $\rho_{i,c}(k)$ from equations (H1a, H1b, H1c, H1d))

$$\frac{\partial \rho_{i,c}(k)}{\partial \rho_{i,c}(k-1)} = 1, \qquad \forall c \in \mathcal{C},\ \forall i \in \mathcal{A} \setminus \mathcal{B}, \qquad \forall k \in [\![1, T]\!] \tag{4.65}$$

$$\frac{\partial \rho_{i,c}(k)}{\partial f^{\text{in}}_{i,c}(k-1)} = \frac{\Delta t}{L_i}, \qquad \forall c \in \mathcal{C},\ \forall i \in \mathcal{A} \setminus \mathcal{B}, \qquad \forall k \in [\![1, T]\!] \tag{4.66}$$

$$\frac{\partial \rho_{i,c}(k)}{\partial f^{\text{out}}_{i,c}(k-1)} = -\frac{\Delta t}{L_i}, \qquad \forall c \in \mathcal{C},\ \forall i \in \mathcal{A} \setminus \mathcal{S}, \qquad \forall k \in [\![1, T]\!] \tag{4.67}$$

Flow in $f^{\text{in}}_{i,c}$ from equation (H6)

$$\frac{\partial f^{\text{in}}_{j,c}}{\partial f^{\text{out}}_{i,c}} = \beta_{ij,c} \qquad \forall i \in \mathcal{J}^{\text{in}}_z,\ \forall j \in \mathcal{J}^{\text{out}}_z,\ \forall z \in \mathcal{J} \qquad \forall k \in [\![1, T-1]\!] \tag{4.68}$$

Flow out $f^{\text{out}}_{i,c}$

Computing the partial derivatives of the flow out $f^{\text{out}}_{i,c}$ is requires a much more involved process. We begin by computing the following intermediate partial derivatives:

- Computing $\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)} \right)$:

  From equations (H2a, H2b),

$$\delta_i(k) = \begin{cases} \min\left(F_i, v_i \rho_i(k)\right) & \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \\ \min\left(F_i, \frac{\rho_i(k) L_i}{\Delta t}\right) & \forall i \in \mathcal{B},\ \forall k \in [\![0, T_f]\!] \end{cases} \tag{4.69}$$

  which gives the following equations:

$$\frac{\rho_{i,c}(k)}{\rho_i(k)} \delta_i(k) = \begin{cases} \min\left(\frac{\rho_{i,c}(k)}{\rho_i(k)} F_i, \rho_{i,c}(k)\, v_i\right) & \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \\ \min\left(\frac{\rho_{i,c}(k)}{\rho_i(k)} F_i, \frac{\rho_{i,c}(k) L_i}{\Delta t}\right) & \forall i \in \mathcal{B},\ \forall k \in [\![0, T_f]\!] \end{cases} \tag{4.70}$$

Using equation (4.1),

$\forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \delta_i(k) \right) = \begin{cases} \frac{(\rho_i(k) - \rho_{i,c}(k))}{\rho_i(k)^2} F_i & \text{if } F_i < v_i \rho_i(k) \\ v_i & \text{otherwise} \end{cases} \tag{4.71}$$

$\forall i \in \mathcal{B},\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \delta_i(k) \right) = \begin{cases} \frac{(\rho_i(k) - \rho_{i,c}(k))}{\rho_i(k)^2} F_i & \text{if } F_i < \frac{L_i}{\Delta t} \rho_i(k) \\ \frac{L_i}{\Delta t} & \text{otherwise} \end{cases} \tag{4.72}$$

Remark 4.12. *If $\rho_i(k) = 0$, then $F_i > \frac{L_i}{\Delta t} \rho_i(k)$ or $F_i > v_i \rho_i(k)$, so the derivatives are well defined.*

- Computing $\frac{\partial}{\partial \rho_{i,c'}(k)} \left(\frac{\rho_{i,c}(k) \delta_i(k)}{\rho_i(k)} \right)$: (Note the two different commodities c and c'.)

  $\forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{i,c'}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \delta_i(k) \right) = \begin{cases} \frac{-\rho_{i,c}(k)}{\rho_i(k)^2} F_i & \text{if } F_i < v_i \rho_i(k) \\ 0 & \text{otherwise} \end{cases} \tag{4.73}$$

  $\forall i \in \mathcal{B},\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{i,c'}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \delta_i(k) \right) = \begin{cases} \frac{-\rho_{i,c}(k)}{\rho_i(k)^2} F_i & \text{if } F_i < \frac{L_i}{\Delta t} \rho_i(k) \\ 0 & \text{otherwise} \end{cases} \tag{4.74}$$

Remark 4.13. *If $\rho_i(k) = 0$, then $F_i > \frac{L_i}{\Delta t} \rho_i(k)$ or $F_i > v_i \rho_i(k)$, so the derivatives are well defined.*

- Computing $\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k) \sigma_j(k)}{\rho_i(k)} \right)$ and $\frac{\partial}{\partial \rho_{i,c'}(k)} \left(\frac{\rho_{i,c}(k) \sigma_j(k)}{\rho_i(k)} \right)$ when $i \neq j$:

  Note that $\sigma_j(k)$ does not contain $\rho_i(k)$ terms.

$$\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \sigma_j(k) \right) = \frac{(\rho_i(k) - \rho_{i,c}(k))}{\rho_i(k)^2} \sigma_j(k) \tag{4.75}$$

$$\frac{\partial}{\partial \rho_{i,c'}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \sigma_j(k) \right) = \frac{-\rho_{i,c}(k)}{\rho_i(k)^2} \sigma_j(k) \tag{4.76}$$

Remark 4.14. *This partial derivative is only needed in cases where the junction is strictly supply constrained and $\rho_i(k) > 0$, so we can ignore the fact that the derivative is undefined at $\rho_i(k) = 0$.*

- Computing $\frac{\partial}{\partial \rho_{j,c'}(k)} \left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)} \right)$:

  From equations (H3a, H3b),

$$\sigma_j(k) = \begin{cases} \min\left(F_i, w_j\left(\rho_j^{\text{jam}} - \rho_j(k)\right)\right) & \forall j \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \\ F_j & \forall j \in \mathcal{S},\ \forall k \in [\![0, T_f]\!] \end{cases} \tag{4.77}$$

$$\frac{\rho_{i,c}(k)}{\rho_i(k)}\sigma_j(k) = \begin{cases} \min\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}F_j, \frac{\rho_{i,c}(k)}{\rho_i(k)}w_j\left(\rho_j^{\text{jam}} - \rho_j(k)\right)\right) & \forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!] \\ \frac{\rho_{i,c}(k)}{\rho_i(k)}F_j & \forall i \in \mathcal{S},\ \forall k \in [\![0, T_f]\!] \end{cases} \tag{4.78}$$

  $\forall i \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{S}),\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}\sigma_j(k)\right) = \begin{cases} 0 & \text{if } F_j < w_j\left(\rho_j^{\text{jam}} - \rho_j(k)\right) \\ -\frac{\rho_{i,c}(k)}{\rho_i(k)}w_j & \text{otherwise} \end{cases} \tag{4.79}$$

  $\forall i \in \mathcal{S},\ \forall k \in [\![0, T_f]\!]$

$$\frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}\sigma_j(k)\right) = 0 \tag{4.80}$$

  Remark 4.15. *This partial derivative is only needed in cases where the junction is strictly supply constrained and $\rho_i(k) > 0$, so we can ignore the fact that the derivative is undefined at $\rho_i(k) = 0$.*

- Computing $\frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)\beta_{ij}(k)} \right)$:

  From equation (H4),

$$\beta_{ij}(k) = \frac{1}{\rho_i(k)} \sum_{c' \in \mathcal{C}} \rho_{i,c'}(k)\,\beta_{ij,c'}(k) \qquad \forall k \in [\![0, T_f]\!] \tag{4.81}$$

  Let $\kappa_{ij}(k) = \sum_{c' \in \mathcal{C}} \rho_{i,c'}(k)\,\beta_{ij,c'}(k)$

$$\frac{\rho_{i,c}(k)}{\rho_i(k)}\frac{\sigma_j(k)}{\beta_{ij}(k)} = \frac{\rho_{i,c}(k)\,\sigma_j(k)}{\kappa_{ij}(k)} \tag{4.82}$$

$$\frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}\frac{\sigma_j(k)}{\beta_{ij}(k)}\right) = \frac{\kappa_{ij}(k)\sigma_j(k) - \rho_{i,c}(k)\,\sigma_j(k)\,\beta_{ij,c}(k)}{\kappa_{ij}(k)^2} \tag{4.83}$$

Remark 4.16. *This partial derivative is only needed in cases where the junction is strictly supply constrained and $\rho_i(k) > 0$, so we can ignore the fact that the derivative is undefined at $\rho_i(k) = 0$.*

- Computing $\frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)\beta_{ij}(k)}\right)$:

$$\frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}\frac{\sigma_j(k)}{\beta_{ij}(k)}\right) = \frac{-\rho_{i,c}(k)\,\sigma_j(k)\,\beta_{ij,c'}(k)}{\kappa_{ij}(k)^2} \tag{4.84}$$

Remark 4.17. *The $\rho_i(k) = 0$ condition is just as in the previous case.*

- Computing $\frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)\beta_{ij}(k)}\right)$:

$$\frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)}{\rho_i(k)}\frac{\sigma_j(k)}{\beta_{ij}(k)}\right) = \frac{1}{\beta_{ij}(k)}\frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\,\sigma_j(k)}{\rho_i(k)}\right) \tag{4.85}$$

which can then be simplified using equations (4.79, 4.80).

Now we can proceed to computing the partial derivatives of $f_{i,c}^{\text{out}}$.

Definition 4.19 (Demand-constrained junction)**.** *A junction is demand-constrained if the flow through the junction is limited by the incoming flow of cell i. We denote this condition by $DC(i)$.*

Definition 4.20 (Supply-constrained junction)**.** *A junction is supply-constrained if the flow through the junction is limited by the outgoing flow into some outgoing cell j. We denote this condition by $SC(j)$.*

Solution for 1×2 junctions

The solutions to all the partial derivatives that appear in the expressions below have already been solved explicitly.

From equation (H5a),

$$f_{i,c}^{\text{out}} = \frac{\rho_{i,c}}{\rho_i}\min\left(\left\{\frac{\sigma_j(k)}{\beta_{ij}(k)},\ \forall j \in \mathcal{J}_z^{\text{out}} \mid \beta_{ij}(k) > 0\right\}, \delta_i(k)\right) \quad \forall z \in \mathcal{J}_{1\times n},\ \forall i \in \mathcal{J}_z^{\text{in}} \tag{4.86}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \begin{cases} \frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)} \right) & \text{if } DC(i) \\ \frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \frac{\sigma_i(k)}{\beta_{ij}(k)} \right) & \text{if } SC(j) \end{cases} \tag{4.87}$$

$$\frac{\partial f_{i,c'}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \begin{cases} \frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c'}(k)\delta_i(k)}{\rho_i(k)} \right) & \text{if } DC(i) \\ \frac{\partial}{\partial \rho_{i,c}(k)} \left(\frac{\rho_{i,c'}(k)}{\rho_i(k)} \frac{\sigma_i(k)}{\beta_{ij}(k)} \right) & \text{if } SC(j) \end{cases} \tag{4.88}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{j,c'}\left(k\right)} = \begin{cases} 0 & \text{if } DC(i) \\ \frac{\partial}{\partial \rho_{j,c'}(k)} \left(\frac{\rho_{i,c}(k)}{\rho_i(k)} \frac{\sigma_i(k)}{\beta_{ij}(k)} \right) & \text{if } SC(j) \end{cases} \tag{4.89}$$

Remark 4.18. *It is important to note that these derivatives are undefined if the junction is both supply and demand-constrained. However, this can only occur if the density of the cell is exactly equal to the value at which the demand and supply constraints meet. This is extremely unlikely in practice with floating point numerical operations. In the rare event that it does occur, we assume that the junction is supply-constrained.*

Solution for 2×1 junctions

The solutions to all the partial derivatives that appear in the expressions below have already been solved explicitly.

From equation (H5b)

$$f_{i,c}^{\text{out}} = \frac{\rho_{i,c}}{\rho_i} \begin{cases} \delta_i & \text{if } P_i\left(\min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) - \delta_i\right) > \delta_i P_{\underline{i}} \\ \min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) - \delta_{\underline{i}} & \text{if } P_{\underline{i}}\left(\min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) - \delta_{\underline{i}}\right) > \delta_{\underline{i}} P_i \\ P_i \min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) & \text{otherwise} \end{cases} \quad \forall z \in \mathcal{J}_{2\times 1},\ \forall i \in \mathcal{J}_z^{\text{in}} \tag{4.90}$$

case 1: $P_i\left(\min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) - \delta_i\right) > \delta_i P_{\underline{i}}$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \frac{\partial}{\partial \rho_{i,c}\left(k\right)} \left(\frac{\rho_{i,c}\left(k\right)\delta_i\left(k\right)}{\rho_i\left(k\right)} \right) \tag{4.91}$$

$$\frac{\partial f_{i,c'}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \frac{\partial}{\partial \rho_{i,c}\left(k\right)} \left(\frac{\rho_{i,c'}\left(k\right)\delta_i\left(k\right)}{\rho_i\left(k\right)} \right) \tag{4.92}$$

$$\frac{\partial f_{i,c'}^{\text{out}}}{\partial \rho_{j,c}\left(k\right)} = 0 \tag{4.93}$$

case 2: $P_{\underline{i}}\left(\min\left(\delta_i + \delta_{\underline{i}}, \sigma_j\right) - \delta_{\underline{i}}\right) > \delta_{\underline{i}} P_i$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \begin{cases} \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)}\right) & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right) - \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_{\underline{i}}(k)}{\rho_i(k)}\right) & \text{otherwise} \end{cases} \tag{4.94}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c'}\left(k\right)} = \begin{cases} \frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)}\right) & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ \frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right) - \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_{\underline{i}}(k)}{\rho_i(k)}\right) & \text{otherwise} \end{cases} \tag{4.95}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{j,c'}\left(k\right)} = \begin{cases} 0 & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ \frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right) & \text{otherwise} \end{cases} \tag{4.96}$$

case 3: otherwise

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c}\left(k\right)} = \begin{cases} P_i\left(\frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)}\right) + \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_{\underline{i}}(k)}{\rho_i(k)}\right)\right) & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ P_i\left(\frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right)\right) & \text{otherwise} \end{cases} \tag{4.97}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{i,c'}\left(k\right)} = \begin{cases} P_i\left(\frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\delta_i(k)}{\rho_i(k)}\right) + \frac{\partial}{\partial \rho_{i,c}(k)}\left(\frac{\rho_{i,c}(k)\delta_{\underline{i}}(k)}{\rho_i(k)}\right)\right) & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ P_i\left(\frac{\partial}{\partial \rho_{i,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right)\right) & \text{otherwise} \end{cases} \tag{4.98}$$

$$\frac{\partial f_{i,c}^{\text{out}}}{\partial \rho_{j,c'}\left(k\right)} = \begin{cases} 0 & \text{if } \delta_i + \delta_{\underline{i}} < \sigma_j \\ P_i\left(\frac{\partial}{\partial \rho_{j,c'}(k)}\left(\frac{\rho_{i,c}(k)\sigma_j(k)}{\rho_i(k)}\right)\right) & \text{otherwise} \end{cases} \tag{4.99}$$

Solution for 2×2 junctions The solution for the 2×2 junctions can be obtained using a similar set of computations, but is omitted here for readability and due to length constraints.

This concludes the computation of all the partial derivatives required for computing the gradient of the system using the discrete adjoint method.

Remark 4.19. *If we do not have closed form solutions for the junctions, it may not be possible to compute the explicit partial derivatives of the outgoing flow with respect to the partial densities of the incoming and outgoing links of the junction. However, for any junction that cannot be solved explicitly, it is still possible to compute* $\frac{\partial f_{i,c}^{\text{out}}(k)}{\partial \rho_{i',c'}(k)}$ *for all* $i, i' \in \mathcal{J}_z^{\text{in}} \cup \mathcal{J}_z^{\text{out}}$ *and* $c, c' \in \mathcal{C}$ *with a finite differences method using* $|\mathcal{J}_z^{\text{in}}| \cdot |\mathcal{J}_z^{\text{out}}| \cdot |\mathcal{C}|$ *local simulations of just the junction dynamics (not the entire system). We can then continue to use the adjoint method while numerically differentiating these junctions that do not admit an explicit solution. This local finite differences method can still allow a very efficient computation of the gradient for heterogeneous networks with some junctions or sub-networks that contain complex dynamics.*

4.5 Numerical Results

To illustrate the effectiveness of our framework for computing the system optimal dynamic flow allocation with partial control, we have implemented the algorithm and tested it on both synthetic and practical traffic rerouting scenarios using experimental field data. Our implementation uses the discrete adjoint method to compute the gradient and uses a log barrier function with a projection step to keep the solution in the physically feasible control set as explained in section 4.4.1. We use the Rprop [112] algorithm as our gradient descent technique. All the experiments were run on a 1.8 GHz Intel Core i5 dual-core processor with 8GB of RAM. The performance cost C of each scenario is measured using the total travel time of all the vehicles passing through the network.

$$C = \sum_{k=0}^{T-1} \sum_{i \in \mathcal{A} \setminus \mathcal{S}} \rho_i(k) \cdot L_i \cdot \Delta t$$

We present numerical results for two network scenarios.

1. A network adapted from the synthetic network used in [140]. The network is illustrated in figure 4.4.
2. A subsection of Interstate 210 with a parallel arterial route, as depicted in figure 4.6.

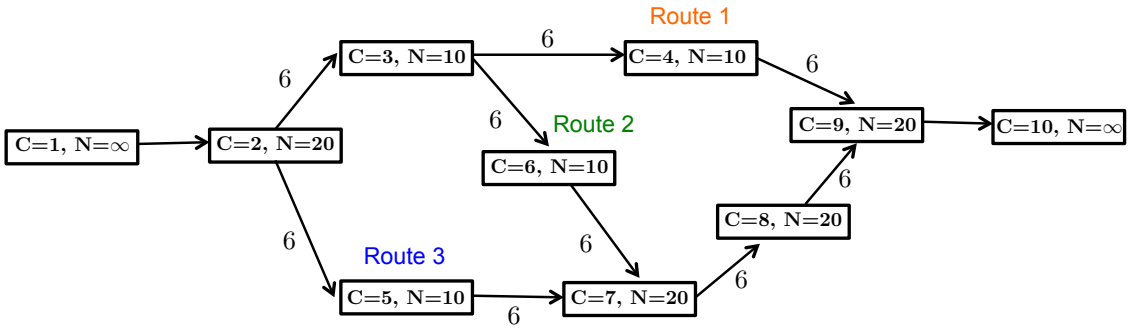


Figure 4.4: The synthetic network adapted from [140]. There are 10 cells marked $C = 1, \cdots, 10$ and the jam density ρ^{jam} of each cell is denoted by the maximum number of vehicles (N), since the length of each cell is normalized to 1. The edge weights represent the max flow F between the neighboring cells.

4.5.1 Synthetic network

The synthetic network is a simple 10 cell network adapted from the example used in [140][5]. It contains three paths over which vehicles can be routed. The demand at the

[5]We have modified the network to increase the capacity of links 8 and 9 such that they can accommodate flow-from both route 2 and route 3, and changed some of the other parameters to satisfy the CFL conditions in section 4.2.3.

origin is given in table 4.1. The time discretization is set to one time unit and the length of each cell is also normalized to one unit. Therefore, the total capacity of each cell in terms of the number of vehicles N is equal to the jam density ρ^{jam}. Each cell in figure 4.4 is annotated with its cell capacity N, while the edge weights in the network prescribe the max flow F between the cells. The free flow speed v of each cell is also normalized to one and the congestion speed w is equal to the free flow speed. The network is simulated for 10 time steps, which gives enough time for all the entering flow to exit the network.

Time step	1	2	3	4	5	6	7	8	9	10
Demand (vehicles)	8	16	8	0	0	0	0	0	0	0

Table 4.1: Demands at origin

First we use the discrete adjoint optimization framework to compute the system optimal flow allocation for the network assuming that all of the flow is compliant. Table 4.2(a) shows the optimal route allocation for the origin demands at each time step with non-zero demand. The total travel time cost (C) with the optimal flow allocation is 178 time units. The solution converges to within 0.5% of the optimal solution in three iterations.

Time step	1	2	3
Route 1	0.75	0.5	0.5
Route 2	0	0	0
Route 3	0.25	0.5	0.5

(a) normal operation

Time step	1	2	3
Route 1	0.25	0.417	.417
Route 2	0	0	0
Route 3	0.75	0.583	0.583

(b) incident local minimum

Time step	1	2	3
Route 1	0	0.25	0.25
Route 2	0.25	0	0
Route 3	0.75	0.75	0.75

(c) incident global minimum

Table 4.2: Optimal allocation of demand across routes

Capacity reduction due to incident. We now consider the case where the capacity between cell 3 and cell 4 is temporarily reduced due to some incident. The corresponding capacities for link $(3, 4)$ are given in table 4.3.

Time step	1	2	3	4	5	6	7	8	9	10
$F_{(3,4)}$	6	6	0	0	3	3	6	6	6	6

Table 4.3: Capacity reduction due to incident

If the vehicles continue to be routed using the previous path allocation, the total travel cost (C) will now be 244 time units. The total cost increases by 37% because a large percentage of vehicles are routed along the path that is temporarily closed and then subjected to a reduced capacity. If we recompute the system optimal flow allocation, the total cost decreases to 211 time units and the corresponding flow allocation is given in table 4.2(b). This solution is actually a local minimum in the system due to the FIFO condition for vehicles departing cell 3. Whenever there is some non-zero flow for route 1 when the capacity is zero,

the flow of vehicles that take route 2 is also restricted to zero. This causes a non-convexity that results in a discontinuity of the gradient at the point where the flow of vehicles on route 1 is zero. Gradient-based methods are not well suited to deal with such conditions because the information obtained from the gradient only provides local information. The global optimal solution occurs with the flow allocation given in table 4.2(c) and results in a total travel time cost of 207. As mentioned in Section 4.4.1, the effect of local minima can in general be mitigated using multi-start strategies [18, 89] and the efficient gradient computation obtained via the adjoint method can be combined with non-convex optimization techniques such as interior point methods [132].

Partial control. In many situations, it might not be possible to reroute all the vehicles in the system. Therefore, we also analyze the behavior of the system when only some fraction of the vehicles are rerouted. Figure 4.5 shows how the total travel time changes with the percentage of vehicles that can be rerouted. In this example, we see that the system optimal (local minimum) can be achieved by controlling only 60% of the vehicles.

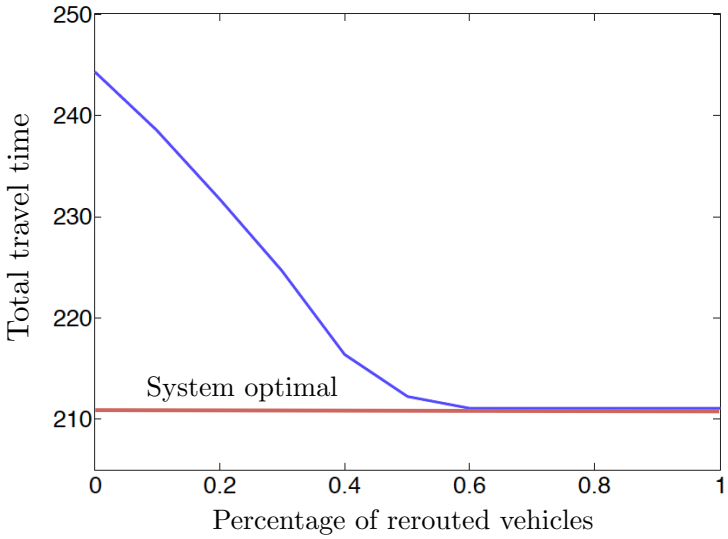


Figure 4.5: The change in total travel time vs percentage of vehicles that can be rerouted. All performance measures are with respect to the local minimum found by the optimizer.

4.5.2 Interstate 210 network

The experimental analysis was conducted on a 8 mile corridor of Interstate 210 in Arcadia, California with a parallel arterial route, as illustrated in figure 4.6. The network has 24 cells corresponding to satisfying the CFL condition for a time step of 30 seconds. The physical properties of the network such as the capacity were obtained using the Scenario Editor software developed as part of the Connected Corridors project, a collaboration between

the University of California Berkeley and California Partners for Advanced Transportation Technology (PATH). Calibrated fundamental diagram parameters, split ratios, and boundary data were also obtained from other parallel research efforts at Connected Corridors. The data used for calibrating these parameters was obtained from the Freeway Performance Measurement System (PeMS) [26]. We consider a prototypical one hour time horizon during the morning commute. The density profile of the freeway under the calibrated parameters and estimated boundary flows is shown in figure 4.7(a).

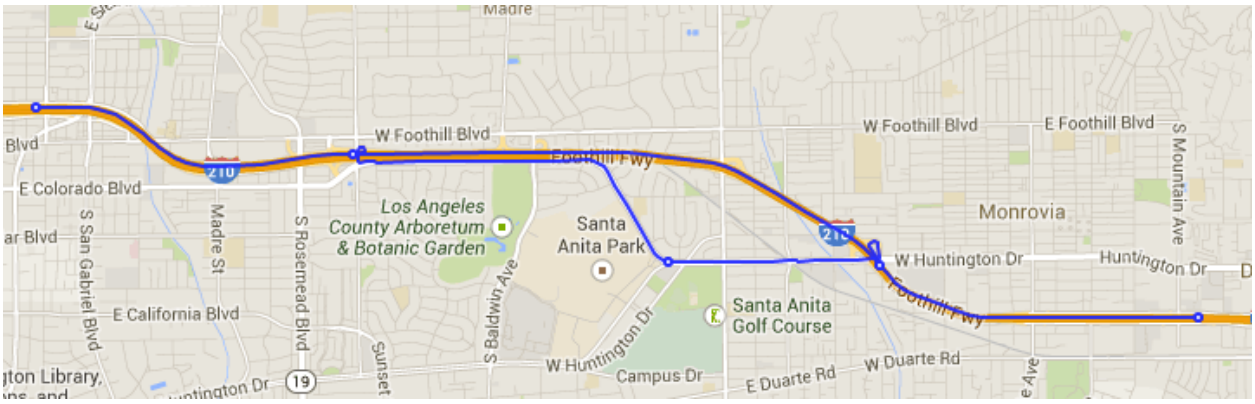
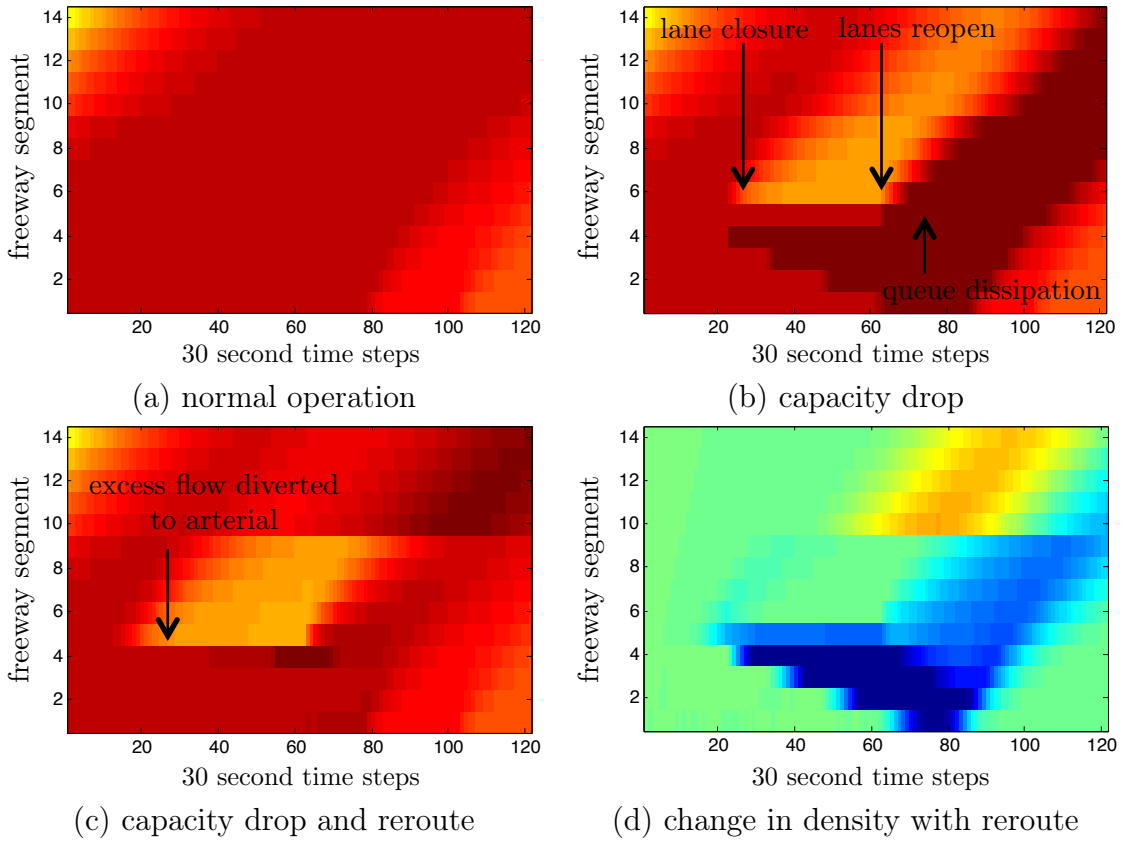


Figure 4.6: The Interstate 210 sub-network

Capacity drop during morning commute. We analyze the behavior of the freeway corridor in the event of a capacity drop caused by some incident. We assume that the capacity drop occurs at the fifth freeway road segment 10 minutes into the simulation and that it lasts for 20 minutes, as illustrated in figure 4.7 (b). The freeway capacity at segment five will be assumed to be reduced by half during this period, corresponding to a closure of two lanes (out of four) at the location of the incident. Figure 4.7 shows the density profile corresponding to; (a) normal operation with capacity drop, (b) a capacity drop due to a two lane closure during the incident with no traffic diversion, (c) the same capacity drop with traffic being diverted to the parallel arterial, and (d) the change in the density profile due to the traffic diversion. As the figure shows, rerouting the excess flow to the parallel arterial eliminates the bottleneck during the incident and improves the throughput of the freeway corridor. In this example, the parallel arterial is assumed to prioritize vehicles being routed from the freeway and the full arterial capacity is used for this purpose. However, in certain situations, municipalities may want to allocate some capacity of the parallel arterial for local traffic. In this case, the optimizer can be limited to only use a certain fraction of the capacity of the parallel arterial. Figure 4.8 shows the density evolution when the arterial capacity allocated for rerouting freeway traffic is limited to 40% and 50% in comparison to full arterial utilization. The arterial capacity allocation can be controlled via the traffic signal controls along the arterial.



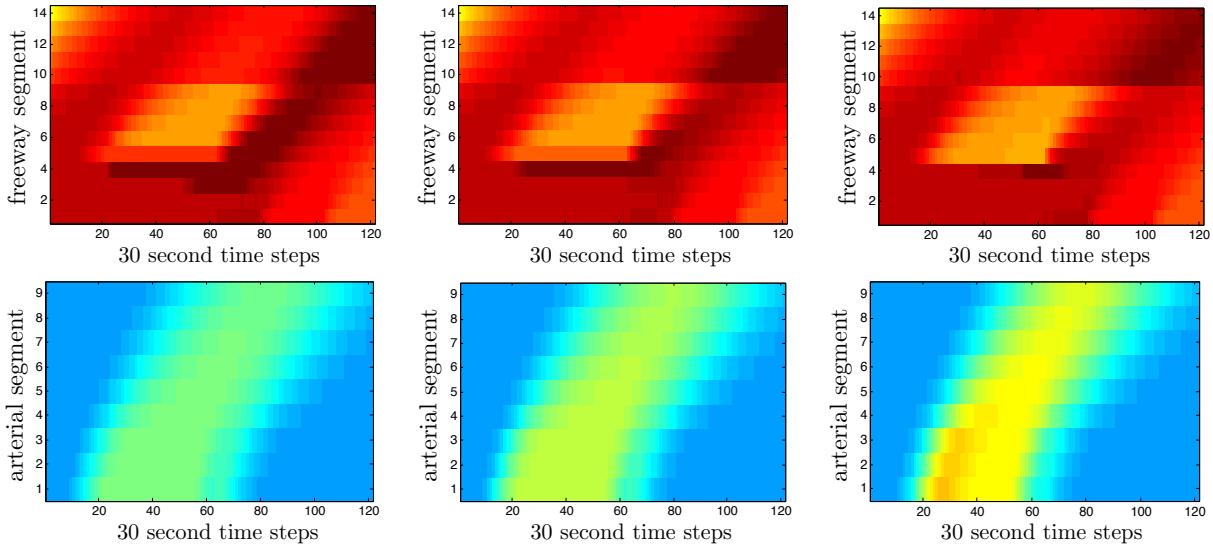
(a) normal operation (b) capacity drop

(c) capacity drop and reroute (d) change in density with reroute

Figure 4.7: The density evolution along the 14 freeway road links with; (a) no incident, (b) a two lane capacity drop from minutes 10-30 at link 5, and (c) flow being rerouted to the parallel arterial due to the capacity drop, and (d) the density difference between the incident profiles with and without rerouting.

Adjoint method vs finite differences. To demonstrate the increased efficiency of computing the solution via the discrete adjoint method, we also implemented the gradient computation using a simple finite differences method [6] by perturbing each variable and measuring the response of the system. This approach, which approximates the gradient at a given point, has a runtime complexity of $O(nm)$ where n is the dimension of the state vector and m is the dimension of the control vector. Recall that the size of the state vector is $n = |\mathcal{A}| \cdot T \cdot |\mathcal{C}|$ and that the size of the control vector is $m = T \cdot |\mathcal{C}|$, where $|\mathcal{C}|$ is the total number of commodities (feasible paths) in the problem. Therefore, $O(nm) = O(|\mathcal{A}| \cdot T^2 \cdot |\mathcal{C}|^2)$ and the finite difference method has a computation time that is quadratic in the number of time steps. In comparison, the adjoint method has a time complexity of $O(n + m|\mathcal{C}|) = O(|\mathcal{A}| \cdot T \cdot |\mathcal{C}| + T \cdot |\mathcal{C}|^2)$, which is linear in the number of time steps. The complexity of both methods is quadratic in the total number of feasible paths, but this is assumed to be a small number in practical routing

[6] See [96] for a detailed analysis of finite difference methods.



(a) 40% of arterial capacity (b) 50% of arterial capacity (c) full arterial utilization

Figure 4.8: A comparison of the density evolution for different rerouting capacities on the parallel arterial route; (a) only 40% of the arterial capacity can be utilized for rerouting freeway vehicles, (b) only 50% of the arterial capacity can be utilized for rerouting freeway vehicles, and (c) The entire arterial capacity is utilized for rerouting freeway vehicles (i.e. the parallel arterial temporarily closed for other traffic).

problems, since vehicles that travel between a fix origin-destination pair will only typically have a small number of reasonable paths. Figure 4.9 shows the time taken for one gradient computation as a function of the network number of time steps in the problem for the I-210 network. The simulations are run by changing the time discretization of the problem to control the total number of time steps. Reducing the time discretization also increases the number of cells due to the maximum cell length imposed by the CFL condition. The results show that the finite differences approach quickly becomes computationally intractable as the number of time steps in the problem increases and highlights the value of the discrete adjoint method for solving large problems in a tractable manner.

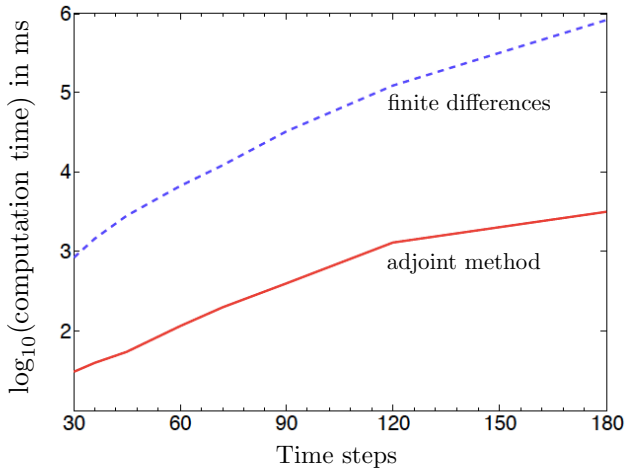


Figure 4.9: The base 10 logarithm ($\log_{10}$) of the total computation time for solving the I-210 network vs the number of time steps in the problem. The total time horizon is fixed, so a larger number of time steps implies a smaller time discretization. This also results in a larger number of cells (smaller in length) due to the CFL condition.

Part III

Control of user equilibrium

Chapter 5

A mathematical framework for delay analysis in single source networks

5.1 Introduction

Modeling and analysing the dynamics of network flows is an important problem that has applications in many different areas such as transportation planning [33, 78, 108], air traffic control [94, 126], communication networks [3, 24, 51, 70], processor scheduling [128] and supply chain optimization [98]. Flow models are crucial for understanding the response of networked systems under different boundary conditions, estimating the state of the system, measuring system performance under different tunable parameters and devising the appropriate control strategies for efficient operation of the system. For example, in transportation networks, flow models are used for traffic estimation [136], dynamic traffic assignment or demand response assessment [90], traffic signal control [84], ramp-metering control [110] and incident rerouting [122].

This chapter focuses on modeling heterogeneous (multi-path) physical flows through a network with a single source and multiple sinks with the specific objective of expressing the delays at each node of the network as a function of the boundary flows at the source. This can be a critical requirement when solving control and optimization problems over a network where the flow entering the network is one of the direct or indirect control parameters of the system. For example, when trying to eliminate congestion at a critical node of the network by manipulating the boundary flows, as explained in Chapter 6. We present our model in the context of physical flow networks and particularly freeway transportation networks, which have the following physical requirements, but the results can be applied to any network that satisfies the following properties: 1) link flows are capacity restricted, 2) the flow through each junction satisfies the first-in-first-out (FIFO) condition, and 3) there is no holding of flow, i.e. the flow through a junction is maximized subject to the FIFO condition.

While there is a vast literature on network flow propagation, particularly for various packet networks, a large majority of these dynamics models violate the FIFO and no hold-

ing requirements listed above, which are essential requirements in phyisical flow networks. Many models proposed for transportation network flows do infact satisfy these physical requirements [33, 78], but none of these models analytically describe the internal delays of the network as a function of the boundary flows, thus requiring a new framework .

Our approach can be summarized as follows. We assume that the traffic flow is differentiated by the destination of the flow (i.e. Lagrangian flow) and that the different flow groups satisfy the FIFO condition at each junction. The queuing in the network is assumed to be contained at each junction node and spill-back to the previous junction if occurs is ignored[1]. We show that our model leads to a well-posed ordinary differential equation for computing the dynamics of the network as a function of the boundary flows and prove that the solution is unique through a mathematical derivation of the model properties. The main benefit of this framework is the ability to analytically describe the delays at any junction in the network and across any sub-path as a function of the the boundary flows, which can be a important requirement when solving certain control and optimization problems. This is achieved via the creation of a time mapping operator that maps the traffic flow at a given node at a given time to the corresponding flow at the origin of the network when that flow entered the network. We also show that this model can be solved numerically using a simple and efficient forward simulation approach. Finally, we demonstrate the application of the model by applying it to two example networks, a single path of multiple bottlenecks and a diverge junction with complex junction dynamics.

The rest of this chapter is organized as follows. Section 5.2 introduces the network properties and junction dynamics. Section 5.3 formalizes the time mapping operator, shows the well-posedness of the problem and proves the uniqueness of the solution to this model. Finally, Section 5.4 demonstrates the applicability of this framework using two examples.

5.2 Point queue model for network flow

The traffic network with a single source is modeled as an arborescence[2]. The congestion at each bottleneck is modeled as a vertical queue that is located at the start of the bottleneck. Thus, the physical propagation of the queue forming at the bottleneck is not modeled. This modeling choice is only restrictive when the queue propagates upstream to the preceding junction, as the change in dynamics at the junction due to the queue is not taken into account, but the model is equivalent to a horizontal queuing model otherwise.

[1]Spill back to the previous junction can be observed and flagged when it occurs. The primary goal of this model is for being used in optimization problems where (in most cases) a good solution will eliminate long spill backs.

[2]An arborescence is a directed rooted tree where all edges point away from the root.

5.2.1 Network definitions

A node v denotes a junction in the network and V is the set of all nodes. A link $l = (v_l^{\text{in}}, v_l^{\text{out}})$ is a couple consisting of an origin node v_l^{in} and a destination node v_l^{out}, and L is the set of all links.

The congestion-free travel time on link l is denoted by T_l, an agent that enters link l at time t will exit link l at time $t + T_l$. The congestion-free travel time between nodes v_1 and v_2 is denoted by $T_{(v_1,v_2)}$, an agent that enters node v_1 at time t will reach node v_2 at time $t + T_{(v_1,v_2)}$

The set of incoming links to node v is denoted by L_v^{in}, the set of outgoing links from node v is denoted by L_v^{out} and the set of all links l connected to node v is denoted by L_v.

$$L_v^{\text{in}} = \{l \in L | v_l^{\text{out}} = v\},\ L_v^{\text{out}} = \{l \in L | v_l^{\text{in}} = v\} \tag{5.1}$$

A node v is a source if it admits no incoming link ($L_v^{\text{in}} = \emptyset$). A node v is a sink if it admits no exiting link ($L_v^{\text{out}} = \emptyset$). The set of sinks is denoted by S.

The set of nodes V and the set of links L compose a network. Due to the network being an arborescence, it contains a unique source indexed by v_0. For all nodes $v \in V \backslash \{v_0\}$, L_v^{in} is a singleton. The element of this singleton is called the *parent* node and is denoted by π_v: $L_v^{\text{in}} = \{(\pi_v, v)\}$.

We define a path $p_{(v_{\text{orig}}, v_{\text{dest}})}$ as a finite sequence of distinct nodes from an origin node v_{orig} to a destination node v_{dest} such that there is a link connecting each pair of subsequent nodes.

$p_{(v_{\text{orig}}, v_{\text{dest}})} = (v_{\text{orig}}, \cdots, v_{\text{dest}})$ s.t. $(\pi_{v_i}, v_i) \in L \quad \forall i \in p \backslash v_{\text{orig}}$

There is a unique path from any source to any destination since the network is tree structured. For each sink s, let p_s be the path starting at the origin v_{orig} and ending at node $v_s = s$, and V_{p_s} be the sequence of nodes on path p_s. The set of paths P_v is the set of all paths p for which $v \in p$. The set of paths P_l is the set of all paths p for which $l \in p$.

$$P_v = \{p | v \in V_p\}\ ;\ P_l = \{p | v_l^{\text{in}} \in V_p \text{ and } v_l^{\text{out}} \in V_p\} \tag{5.2}$$

Remark 5.1. *The path sets P_l where l is a link in L_v^{out} form a partition of P_v*

$$P_v = \cup_{l \in L_v^{out}} P_l \tag{5.3}$$

□

5.2.2 Modeling the flow of agents

The traffic flow at a node is measured by counting the number of agents that pass through the node between an arbitrary initial time t_{initial} and any given time t.

For a node $v \in V \backslash v_0$ (that is not the source) and path $p \in P_v$, the arrival curve $A_v^p(t)$ gives the total number of agents on path p that arrive at node v during the time interval

$(t_{\text{initial}}, t]$. Similarly, for a node $v \in V \backslash S$ (that is not a sink) and $p \in P_v$, the departure curve $D_v^p(t)$ gives the total number of agents on path p that leave node v during the time interval $(t_{\text{initial}}, t]$.

Remark 5.2. *The arrival curve $A_v^p(t)$ (resp. departure curve $D_v^p(t)$) also gives the agent number of the last agent on path p to arrive at (resp. leave) node v by t. Arrival and departure curves are monotonically increasing: if $t_1 < t_2$, $A(t_2) - A(t_1)$ (resp. $D_p(t_2) - D_p(t_1)$) is the total number of agents who arrive at (resp. pass) node v in the interval $(t_1, t_2]$, and is therefore non-negative.*

Definition 5.1. *Acceptable cumulative arrival and departure curves $\mathbf{A}(t_{initial}, t_{final}]$, $\mathbf{D}(t_{initial}, t_{final}]$*
Given times $t_{initial}$ and t_{final}, a function on $(t_{initial}, t_{final}]$ is an acceptable cumulative curve on $(t_{initial}, t_{final}]$ if it is continuous, piecewise C^1, and strictly increasing functions on $(t_{initial}, t_{final}]$.

The assumption that the cumulative curves are strictly increasing is made for mathematical convenience, but can be relaxed[3]. Cumulative curves are required to be C^1 in order to be able to define flows.

The outgoing flow λ_p^v at a node v is the piecewise continuous derivative of the departure curve D_p^v

$$\lambda_p^v = \frac{dD_p^v}{dt} \tag{5.4}$$

Remark 5.3. *Zero congestion-free travel time*
Let π_v, v be two consecutive nodes on path p. agents on path p leaving node v at time t arrive at node v at $t + T_{(\pi_v,v)}$. For all links (π_v, v) and paths $p \in P_v$, without loss of generality we set the congestion-free travel time $T_{(\pi_v,v)}$ to be zero: $T_{(\pi_v,v)} = 0$. This implies that:

$$D_p^{\pi_v} = A_p^v \quad \forall l = (\pi_v, v) \in L, p \in P_v \tag{5.5}$$

This modeling choice is made purely for mathematical convenience, since the goal of this framework is to analyze delays in the network. The total travel time for each agent can be easily reconstructed a posteriori by adding the actual congestion-free travel time for each link of the path traveled by the agent.

Thus, for all links $(\pi_v, v) \in L$ and paths $p \in P$ we have:

$$\frac{dA_p^v}{dt} = \frac{dD_p^{\pi_v}}{dt} = \lambda_p^{\pi_v} \tag{5.6}$$

$$\frac{dD_p^v}{dt} = \lambda_p^v. \tag{5.7}$$

[3]We could relax the assumption that the cumulative curves are strictly increasing and allow for monotonically increasing curves. However, this results in the time mapping function $T^{(\cdot\pi_v)}v$ introduced in section 5.3.2 being a correspondence instead of a function and makes the analysis significantly more complicated. Therefore, for mathematical convenience, we make the assumption that the cumulative curves are strictly increasing.

5.2.3 Queuing and diverge model

This section defines the model dynamics for queuing and the flow propagation through a junction, which will then lead to a definition of the feasible departure curves that the model admits.

The capacity $\mu_l(t)$ of a link l is the maximum flow that can enter the link from its input node v_l^{in} at time t. Road capacity may vary with time due to weather conditions, accidents, or other factors. Thus, capacity is a time varying quantity.

Requirement 5.1. *Capacity constrained flows*
The inflow entering a link is always no greater than the links capacity.

$$\sum_{p \in P_l} \lambda_p^{v_l^{in}}(t) \leq \mu_l(t) \qquad \forall t, l \in L \tag{5.8}$$

If the flows arriving at a node v are larger than available outflow capacity, a queue will form at node v.

Definition 5.2. *Queue length $n_{v,p}(t)$*
We define the path queue length $n_{v,p}(t)$ at node v as the number of agents on path p that arrive at node v by time t and are yet to depart node v

$$n_{v,p}(t) = D_p^v(t) - A_p^v(t) \tag{5.9}$$

The total queue length $n_v(t)$ at node v is the sum of the path queue lengths.

$$n_v(t) = \sum_{p \in P_v} n_{v,p}(t) \tag{5.10}$$

Remark 5.4. *Let $[D^v]^{-1}$ be the inverse of the departure curve D^v. Since D^v is strictly increasing, $t_k = [D^v]^{-1}(k)$ gives the time at which agent number k leaves node v.*

Definition 5.3. *Delay in queue v*
We define $\delta_{v,p}(t)$ as the delay encountered in queue v by the agent which entered the queue at time t.

$$\begin{aligned} \delta_{v,p}(t) &= [D_p^v]^{-1}\left(A_p^v(t)\right) - t \\ &= [D_p^v]^{-1}\left(D_p^{\pi_v}(t)\right) - t \end{aligned} \tag{5.11}$$

As D_p^v is continuous, piecewise C^1, and strictly increasing, its inverse is continuous, piecewise C^1 and strictly increasing. Thus, as $D_p^{\pi_v}$ is also continuous, piecewise C^1 and strictly increasing, the function $[D_p^v]^{-1} \circ D_p^{\pi_v}$ is continuous and piecewise C^1, and delay $\delta_{v,g}$ is continuous and piecewise C^1.

Remark 5.5.
If $n_{v,p}(t) = 0$, *then* $D_p^v(t) = A_p^v(t) \implies \delta_{v,p}(t) = 0$.
If $n_{v,p}(t) > 0$, *then* $D_p^v(t) < A_p^v(t) \implies [D_p^v]^{-1}\left(A_p^v(t)\right) > t$ *and* $\delta_{v,p}(t) > 0$.
Therefore,

$$\forall t, \delta_{v,p}(t) > 0 \Leftrightarrow n_{v,p}(t) > 0 \tag{5.12}$$

Requirement 5.2. *First-in-first-out (FIFO) property*
The model satisfies the FIFO property. The delay encountered in queue v *at time* t *is identical for all paths* p *in* P_v.

$$\delta_v(t) = \delta_{v,p}(t) = [D_p^v]^{-1}\left(D_p^{\pi_v}(t)\right) - t \quad \forall t, \forall p \in P_v \tag{5.13}$$

FIFO property implies that agents exit the queue in the same order that they enter the queue regardless of which path they belong to.

$$t_1 < t_2 \Leftrightarrow [D_{p_1}^v]^{-1}\left(D_{p_1}^{\pi_v}(t_1)\right) < [D_{p_2}^v]^{-1}\left(D_{p_2}^{\pi_v}(t_2)\right) \tag{5.14}$$

Interpreting A_p^v (resp D_p^v) as the identifier of the agent which arrives in (resp. leaves) queue v at time t, we can see that the queues respect the FIFO rule for each path p. Let x_1 and x_2 be two agents: agent x_1 enters queue v at time t_1^{in} such that $A_p^v(t_1^{in}) = x_1$ and leaves queue v at time t_1^{out} such that $D_p^v(t_1^{out}) = x_1$, agent x_2 entered in queue v at time t_2^{in} such that $A_p^v(t_2^{in}) = x_2$ and leaves queue v at time t_2^{out} such that $D_p^v(t_2^{out}) = x_2$. As A_p^v and D_p^v are both strictly increasing functions, $t_1^{in} \leq t_2^{in} \Rightarrow x_1 \leq x_2 \Rightarrow t_1^{out} \leq t_2^{out}$, which means that if x_1 is enters queue v before x_2, it will leave v before x_2.

Proposition 5.1. *FIFO implies conservation of the ratio of flows*
If p_1 *and* p_2 *are two paths in* P_v *such that* $\lambda_{p_1}^{\pi_v}, \lambda_{p_2}^{\pi_v} > 0$, *then the ratio of their flows is conserved when exiting node* v

$$\frac{\lambda_{p_1}^v(t + \delta_v(t))}{\lambda_{p_2}^v(t + \delta_v(t))} = \frac{\lambda_{p_1}^{\pi_v}(t)}{\lambda_{p_2}^{\pi_v}(t)}, \quad \forall t \in (t_{init}, t_{final}] \tag{5.15}$$

Proof: Let t be an arbitrary time. The FIFO assumption gives $\delta_{v,p}(t) = \delta_v(t)$. By definition of delay $\delta_{v,p}(t)$,

$$D_p^{\pi_v}(t) = D_p^v(t + \delta_{v,p}(t)) \quad \forall p \in P_v$$

Taking the derivative with respect to t and using $\delta_{v,p}(t) = \delta_v(t)$,

$$\frac{dD_p^{\pi_v}(t)}{dt} = \left(1 + \frac{d\delta_v(t)}{dt}\right) \cdot \left.\frac{dD_p^v}{dt}\right|_{t+\delta_v(t)}$$

Using equation (5.7) we obtain,

$$\lambda_p^{\pi_v}(t) = \left(1 + \frac{d\delta_v(t)}{dt}\right) \cdot \lambda_p^v(t + \delta_v(t)) \quad \forall p \in P_v$$

Therefore, it follows that

$$\frac{\lambda_{p_1}^{v}\left(t+\delta_v\left(t\right)\right)}{\lambda_{p_2}^{v}\left(t+\delta_v\left(t\right)\right)} = \frac{\lambda_{p_1}^{\pi_v}\left(t\right)}{\lambda_{p_2}^{\pi_v}\left(t\right)}$$

□

Definition 5.4. *Queue state η_v - state transitions*
We define queue state as the boolean valued function $\eta_v\left(t\right)$:

$$\eta_v\left(t\right) = \begin{cases} 1 & \text{if } \delta_v\left(t\right) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

If $\eta_v = 1$, queue v is said to be active, or in active state
If $\eta_v = 0$, queue v is said to be inactive, or in inactive state

A queue state transition happens at time t if

$$\exists \epsilon > 0 \text{ s.t. } \forall \theta \in [-\epsilon, \epsilon], \quad \eta_v\left(t-\theta\right) = 1 - \eta_v\left(t+\theta\right) \tag{5.17}$$

When queue v is inactive, $D^v = D^{\pi_v}$.

Definition 5.5. *Link constraint $c_{v,l}\left(t\right)$*
Let $v \in V \backslash \{v_0 \cup S\}$ be a node which is not a source or a sink. For all links $l \in L_v^{out}$, we define the link constraint $c_{v,l}\left(t\right)$ as the ratio of arriving flows at time t on capacity at queue v when this flow leaves queue v[4]*.*

$$c_{v,l}\left(t\right) = \frac{\sum_{p \in P_l} \lambda_p^{\pi_v}\left(t\right)}{\mu_l\left(t+\delta_v\left(t\right)\right)} \tag{5.18}$$

Definition 5.6. *Active link $\gamma_v\left(t\right)$ and set of active paths $\Gamma_v\left(t\right)$ of a node*
We define the active link $\gamma_v\left(t\right)$ of a node v at time t as the most constrained link [5] *in L_v^{out}:*

$$\gamma_v\left(t\right) \in \underset{l \in L_v^{out}}{\arg\max}\, c_{v,l}\left(t\right) \tag{5.19}$$

We define the set of active paths $\Gamma_v\left(t\right)$ in queue v as the set of paths in the most constrained link $\gamma_v\left(t\right)$

$$\Gamma_v\left(t\right) = P_{\gamma_v(t)} \tag{5.20}$$

Remark 5.1 gives $\Gamma_v \subset P_v$.

[4]The dissipation rate of the point queue at the node is only governed by the capacities of the outgoing links. This model can be extended to also impose a discharge rate constraint based on the capacity of the incoming link, but increases the complexity of the notation and the proofs.

[5]When there is a tie, one of them is chosen arbitrarily.

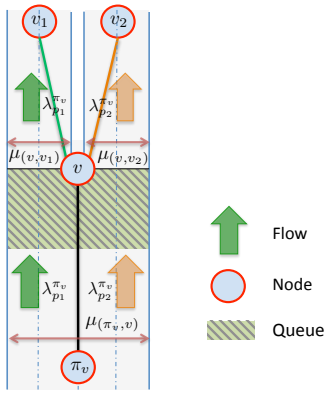


Figure 5.1: Diverge model.

Requirement 5.3. *Full capacity discharge property*
The model satisfies the full capacity discharge property. For each node v and time t, if queue v is active at t, then the active link $\gamma_v(t)$ discharges at full capacity.

$$\delta_v(t) > 0 \Rightarrow \sum_{p \in \Gamma_v(t)} \lambda_p^v(t + \delta_v(t)) = \mu_{\gamma_v(t)}(t + \delta_v(t)) \tag{5.21}$$

With this last property, we complete the definition of the dynamics model.

Definition 5.7. ***Feasible flows***
A feasible flow λ_p^v at a node v is a flow that satisfies the FIFO, capacity constraint and full capacity discharge properties from requirements 5.1, 5.2 and 5.3.

The definition of the initial conditions on the network completes the definition of the model.

Definition 5.8. *Initial times for each non-source node*
Given a set of initial delays at each node $\delta_v(t_{initial}) \geq 0, \forall v \in V \backslash (S \cup \{v_0\})$ and an initial time $t_{initial}$, we define the set of initial times over which the departure curves are defined for each non-source node recursively as follows:

$$\begin{cases} t_{0,initial} &= t_{initial} & \text{for node } v_0 \\ t_{v,initial} &= t_{\pi_v,initial} + \delta_v(t_{\pi_v,initial}) \end{cases} \tag{5.22}$$

5.2.4 Existence and uniqueness of the solution to the model

Now that we have fully defined the model dynamics, we consider the well-posedness of the model. In other words, given a network, link capacities and the departure functions at the source, we want to know whether the dynamics of the model admits a unique solution.

Problem 1: General network problem
Input. An arborescence (V, L) with source v_0 and sink set S, capacities $\mu_l(t), \forall l \in L,\ t \in$

$[t_{\text{initial}}, t_{\text{final}}]$, acceptable departure functions from the source $D_p^{v_0} \in \mathbf{D}(t_{\text{initial}}, t_{\text{final}})\ \forall p \in P_{v_0}$ and initial delays $\delta_v(t_{\text{initial}}) \geq 0,\ \forall v \in V \backslash (S \cup \{v_0\})$
Question. Does a corresponding set of feasible flows exist for all internal nodes $v \in V \backslash v_0$ and are they unique?

Theorem 5.1 stats that the solution to problem 1 both exists and that the solution is unique, under certain conditions on the departure curves at the origin and the link capacities of the network.

Theorem 5.1. ***Existence and uniqueness of the solution to problem 1***
Problem 1 admits a unique solution under the following conditions.
1) the path flows at the origin $\lambda_p^0(t)$ are piecewise polynomial,
2) link capacities μ_l are piecewise constant over time.

Note that neither of the assumptions of the theorem are restrictive in a practical sense[6].

The next section is devoted to a constructive proof of Theorem 5.1. The general flow of the proof is as follows. Sections 5.3.1-5.3.3 first develop a set of differential equations for delays in the network. In section 5.3.4, we then prove that a unique solution to differential equation on delays also implies a unique solution to problem 1. Section 5.3.5 proves that the differential equations on the delay at each node always admit an unique solution, which finally leads to the proof of Theorem 5.1.

5.3 A solution based on time mapping

This section builds a constructive proof of Theorem 5.1. Throughout sections 5.3.1-5.3.3, we require that the flows at the origin are acceptable departure curves as defined in definition 5.1 and that the outflows at each node satisfy the model requirements (i.e. result in feasible flows as defined in definition 5.7).

5.3.1 Local study of point queues

We begin by proving proposition 5.2, which gives an analytical expression for the derivative of the delay at node as a function of its downstream capacities and outgoing flow at its parent nodes.

Proposition 5.2. *Evolution law of a single queue*
If queue v is active at time t,

$$\left.\frac{d\delta_v}{dt}\right|_t = \frac{\sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}(t)}{\mu_{\gamma_v(t)}(t + \delta_v(t))} - 1 \tag{5.23}$$

[6]Neither of these assumptions are restrictive in a practical sense, because any piecewise continuous function on a closed interval can be approximated to an arbitrary accuracy by a polynomial of appropriate degree (Stone-Weierstrass theorem [129]) and link capacities do not evolve in a continuous manner. Link capacities are typically subject to discrete changes due to incidents such as accidents and changes in weather.

The proof of this proposition requires the following lemma.

Lemma 5.1. *Derivative of queue's length n_v with respect to time*
If node v is active at time t (i.e. t: $\gamma_v(t) = 1$),

$$\sum_{p\in\Gamma_v(t)} \left.\frac{dn_{v,p}}{dt}\right|_{t+\delta_v(t)} = \left[\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}\left(t+\delta_v(t)\right)\right] - \mu_{\gamma_v(t)}\left(t+\delta_v(t)\right) \tag{5.24}$$

Proof: By definition 5.2, $n_{v,p}(t) = D_p^{\pi_v}(t) - D_p^v(t)$. Thus,

$$\begin{aligned}\sum_{p\in\Gamma_v} \left.\frac{dn_{v,p}}{dt}\right|_t &= \sum_{p\in\Gamma_v} \left.\left(\frac{dD_p^{\pi_v}}{dt} - \frac{dD_p^v}{dt}\right)\right|_t \\ &= \sum_{p\in\Gamma_v} \left.\left(\lambda_p^{\pi_v} - \lambda_p^v\right)\right|_t \end{aligned} \tag{5.25}$$

As queue v is active at time t, requirement 5.3 gives $\sum_{p\in\Gamma_v}\lambda_p^v(t+\delta_{v,t}) = \mu_{\gamma_v(t)}(t+\delta_v(t))$, thus we have

$$\sum_{p\in\Gamma_v(t)} \left.\frac{dn_{v,p}}{dt}\right|_{t+\delta_v(t)} = \left(\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}(t+\delta_{v,t})\right) - \mu_{\gamma_v(t)}(t+\delta_v(t)) \tag{5.26}$$

□

Lemma 5.2. *Discharge relationship between queue length and delay*

$$n_{v,p}(t+\delta_v(t)) = D_p^{\pi_v}(t+\delta_v(t)) - D_p^{\pi_v}(t), \quad \forall v\in V, p\in P_v \tag{5.27}$$

Proof: By definition 5.2 on queue length, we have $n_{v,p}(t) = D_p^{\pi_v}(t) - D_p^v(t)$, which evaluated at time $t+\delta_{v,p}(t)$ gives $n_{v,p}(t+\delta_{v,p}(t)) = D_p^{\pi_v}(t+\delta_{v,p}(t)) - D_p^v(t+\delta_{v,p}(t))$. From definition 5.3 on queue delay, we have $D_p^v(t+\delta_{v,p}(t)) = D_p^{\pi_v}(t)$. Combining these two results we obtain,

$$n_{v,p}(t+\delta_v(t)) = D_p^{\pi_v}(t+\delta(t)) - D_p^{\pi_v}(t) \tag{5.28}$$

□

We can now prove Proposition 5.2.
Proof of Proposition 5.2: Let t be a time such that $\eta_v(t) = 1$. Equation (5.24) multiplied by $\left(1 + \left.\frac{d\delta_v}{dt}\right|_t\right)$ gives

$$\left(1+\left.\frac{d\delta_v}{dt}\right|_t\right)\cdot \sum_{p\in\Gamma_v(t)} \left.\frac{dn_{v,p}}{dt}\right|_{t+\delta_v(t)} = \tag{5.29}$$

$$\left[\left(1+\left.\frac{d\delta_v}{dt}\right|_t\right)\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}(t+\delta_v(t))\right] - \left(1+\left.\frac{d\delta_v}{dt}\right|_t\right)\mu_{\gamma_v(t)}(t+\delta_v(t)) \tag{5.30}$$

Taking the derivative of equation (5.27) with respect to time and summing over $p \in \Gamma_v$, gives the following equality

$$\left(1 + \left.\frac{d\delta_v}{dt}\right|_t\right) \cdot \sum_{p \in \Gamma_v(t)} \left.\frac{dn_{v,p}}{dt}\right|_{t+\delta_v(t)} =$$

$$\left[\left(1 + \left.\frac{d\delta_v}{dt}\right|_t\right) \cdot \sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}\left(t + \delta_v\left(t\right)\right)\right] - \sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}\left(t\right) \tag{5.31}$$

Given equations (5.29) and (5.31) have the same left hand side, equalizing their respective right hand sides and simplifying $\left[\left(1 + \left.\frac{d\delta_v}{dt}\right|_t\right) \cdot \sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}\left(t + \delta_{v,p}\left(t\right)\right)\right]$ gives the following equation:

$$\left(1 + \left.\frac{d\delta_v}{dt}\right|_t\right) \cdot \mu_{\gamma_v(t)}\left(t + \delta_v\left(t\right)\right) = \sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}\left(t\right) \tag{5.32}$$

Which gives the result,

$$\left.\frac{d\delta_v}{dt}\right|_t = \frac{\sum_{p \in \Gamma_v(t)} \lambda_p^{\pi_v}\left(t\right)}{\mu_{\gamma_v(t)}\left(t + \delta_v\left(t\right)\right)} - 1 \tag{5.33}$$

□

5.3.2 Time mapping

The evolution law stated above for any given node v depends on the outgoing flows $\lambda_p^{\pi_v}$ at the parent node. However, this it not an input of Problem 1. In this section, we introduce the notion of time mapping to obtain a modified law for the delay evolution that replaces the outgoing flows at the parent node with the outgoing flow at the origin.

Definition of time mapping functions

The evolution law from Proposition 5.2 gives a non-linear *ordinary differential equation* (ODE) that governs the evolution of $\delta_v\left(t\right)$. The evolution of delay encountered by an agent x entering queue v at time t depends on the flows entering the queue at t and the capacity of the active link(s) γ_v at time $t + \delta_v\left(t\right)$ when agent x leaves the queue. The non-linearity of the ODE makes directly computing the dynamics along a path algebraically complex. Therefore, we introduce a time mapping function.

Let v be an internal node of the network and its parent node be π_v. an agent leaving node π_v at time t will leave node v at time $t + \delta_v\left(t\right)$. We now introduce the following time mapping function:

Definition 5.9. *Node time mapping function T^{v,π_v}*
We define the time mapping function T^{v,π_v} by

$$T^{v,\pi_v} : t \mapsto t + \delta_v\left(t\right) \tag{5.34}$$

an agent leaving node π_v at time t will leave node v at time $T^{v,\pi_v}(t)$

The notation T^{v,π_v} (variable ordering) is chosen for mathematical convenience with respect to the derivatives of the function, as will be apparent in the rest of the discussion. In equation (5.34), T^{v,π_v} takes a time with a physical meaning at the exit of node π_v on its right hand side, and gives back a time with a physical meaning at the exit of node v on its left hand side.

Proposition 5.3. *T^{v,π_v} is strictly increasing and bijective*
The function T^{v,π_v} is strictly increasing and thus bijective from its domain to its image. Its derivative is

$$\left.\frac{dT^{v,\pi_v}}{dt}\right|_t = \frac{\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}(t)}{\mu_l\left(t+\delta_v(t)\right)} > 0 \tag{5.35}$$

Physically, this means that the FIFO assumption is respected: i.e. an agent x_2 entering queue v after another agent x_1 will also leave the queue after x_1

Proof: Taking the derivative of equation (5.34) and applying equation (5.23) in Proposition 5.2 gives,

$$\left.\frac{dT^{v,\pi_v}}{dt}\right|_t = \frac{\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}(t)}{\mu_l\left(t+\delta_v(t)\right)}. \tag{5.36}$$

The departure curves at the origin are strictly increasing since they must be acceptable departure curves. The full capacity discharge property from requirement 5.3 requires that one outgoing link at each node discharges at full capacity. Finally, these properties combined with Proposition 5.1, which states that the out flows at a node are proportional to the inflows, give us the result that $\left.\frac{dT^{v,\pi_v}}{dt}\right|_t > 0$. □

Thus T^{v,π_v} is invertible and its inverse is an increasing function[7].

Definition 5.10. *Node time mapping function $T^{\pi_v,v}$ Given an internal node v, we define the function $T^{\pi_v,v}$ as the inverse of T^{v,π_v}*

$$T^{\pi_v,v} \circ T^{v,\pi_v} = \mathbb{1} \text{ and } T^{v,\pi_v} \circ T^{\pi_v,v} = \mathbb{1} \tag{5.37}$$

We now consider the unique path $(v_0, v_1, \cdots, v_{n-1}, v_n)$ which leads from the source v_0 to some node v_n. As each node has a unique parent, we can recursively trace the path from node v back to the source node v_0. Let t^{v_n} be a fixed time. If an agent x leaves node v_n at the time t^{v_n}, we can recursively define the following:
1) $t^{v_{n-1}}=T^{v_{n-1},v_n}(t^{v_n})$ is the time that agent x left v_{n-1}, $t^{v_n}=t^{v_{n-1}}+\delta_v(t^{v_{n-1}})$
2) $t^{v_{n-2}}=T^{v_{n-2},v_{n-1}}(t^{v_{n-1}})$ is the time that agent x left v_{n-2}, $t^{v_n}=t^{v_{n-2}}+\delta_{v_{n-1}}(t^{v_{n-2}})+\delta_v(t^{v_{n-2}}+\delta_{v_{n-1}}(t^{v_{n-2}}))$ 3) $t^{v_{n-3}}=T^{v_{n-3},v_{n-2}}(t^{v_{n-2}})$ is the time that agent x left v_{n-3}, $\cdots$
As T^{v,π_v} and $T^{\pi_v,v}$ are bijective for all internal nodes v, we can give the following definition

[7]If the acceptable set of departure curves $\mathbf{D}$ is relaxed to allow monotonically increasing instead of strictly increasing functions, T^{v,π_v} becomes a correspondence, and the mathematical treatment would be more involved.

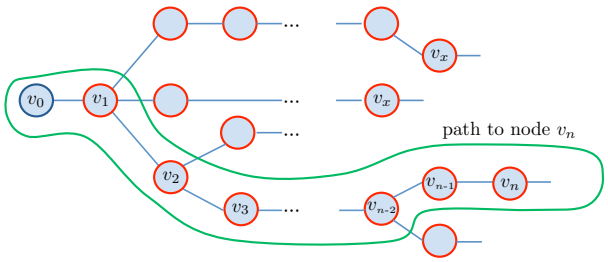


Figure 5.2: Time mapping nodes

Definition 5.11. *Time mapping function from and to the origin T^{v,v_0} and $T^{v_0,v}$*
Let v_n be a node, and $(v_0, v_1, v_2, \cdots, v_{\pi_n}, v_n)$ be a path from the origin v_0 to node v. We define the time mapping function to the origin as the composition of the node time mapping function on the path between the source and v_n

$$T^{v_0,v_n} = T^{v_0,v_1} \circ T^{v_1,v_2} \circ \cdots \circ T^{v_{\pi_n},v_n} \tag{5.38}$$

an agent that leaves node v_n at time t left the origin v_0 at time $T^{v_0,v_n}(t)$.

$$T^{v_n,v_0} = T^{v_n,v_{\pi_n}} \circ \cdots \circ T^{v_2,v_1} \circ T^{v_1,v_0} \tag{5.39}$$

an agent that leaves the origin at time t will leave node v_n at time $T^{v_n,v_0}(t)$

A sample path from the origin v_0 to a node v_n is illustrated in figure 5.2. We can now define the time mapping function between any arbitrary pair of nodes.

Definition 5.12. *Time mapping function between two arbitrary nodes*
We define the time mapping function $T^{i,j}$ between node i and node j as follows.
1) There exists a path between nodes i and j (for example nodes v_2 and v_n in figure 5.2),

$$T^{i,j} = \begin{cases} T^{i,i+1} \circ T^{i+1,i+2} \circ \cdots \circ T^{j-2,j-1} \circ T^{j-1,j} & \text{if } i \prec j \\ T^{i,i-1} \circ T^{i-1,i-2} \circ \cdots \circ T^{j+2,j+1} \circ T^{j+1,j} & \text{if } i \succ j \end{cases} \tag{5.40}$$

Let x be an agent that leaves node j at time t. $T^{i,j}(t)$ is the time that agent x leaves node j.

2) There does not exist a path between nodes i and j (for example nodes v_2 and v_x in figure 5.2),

$$T^{i,j} = T^{i,v_0} \circ T^{v_0,j} \tag{5.41}$$

Let x_j be an agent that leaves node j at t. From definition 5.11 we know that x_j leaves the origin at time $T^{0,j}(t)$. Let x_i be an agent that also leaves the origin at time $T^{0,j}(t)$. Then $T^{i,j}(t)$ is the time that agent x_i leaves node i.

Definition 5.13. *Time mapping operator $\mathbf{T}^{i,j}$*
We define the time mapping operator $\mathbf{T}^{i,j}$ on the set F of time dependent functions as follows:

$$\begin{array}{rccc} \mathbf{T}^{i,j}: & F & \to & F \\ & f & \mapsto & f \circ T^{j,i} \end{array} \tag{5.42}$$

We now consider the physical interpretation of $T^{i,j}$.

Time mapping of model quantities

This section first studies the relationship between departure curves at different nodes and the time mapping function. We then define the time mapped versions of the other quantities in the model. The time mapping operators allow for mapping any quantity from one node to the other. This definition of a time mapped quantities thus allows any quantity to be defined with respect to the source node of the network.

Proposition 5.4. *Physical interpretation of the time mapping function*
Let p be a path, and $(v_0, v_1, v_2, \cdots, v_n)$ be a sequence of consecutive nodes on the path.

$$D_p^{v_i} = D_p^{v_0} \circ T^{v_0,v_i} \qquad \forall v_i \in p \tag{5.43}$$

Let $x = D_p^{v_0}(t^{v_0})$ be an agent on path p that leaves the origin at time t^{v_0} and $t^{v_i} = T^{v_i,0}(t^{v_0}) \, \forall v_i \in p$.

$$D_p^{v_0}(t^{v_0}) = D_p^{v_1}(t^{v_1}) = \cdots = D_p^{v_i}(t^{v_i}) = \cdots = D_p^{v_n}(t^{v_n}) \tag{5.44}$$

Proof: Proof by induction on the length of the sequence k. If $k = 0$, the result is trivial. Let $k \in [1, i]$ be an integer. By the induction hypothesis, we assume that the result is true for to $k = i - 1$, i.e. $D_p^{v_{i-1}} = D_p^{v_0} \circ T^{0,v_{i-1}}$. By the definition of path delay $\delta_{v,p}$, $D_p^{v_i}(t + \delta_{v_i,p}(t)) = D_p^{v_{i-1}}(t), \forall t$, which means $D_p^{v_{i-1}} = D_p^{v_i} \circ T^{v_i,v_{i-1}}$. Composing both sides of the equality with T^{v_{i-1},v_i} we get $D_p^{v_i} = D_p^{v_{i-1}} \circ T^{v_{i-1},v_i}$. Substituting the induction hypothesis and simplifying the results completes the proof.

$$\begin{aligned} D_p^{v_i} &= D_p^{v_{i-1}} \circ T^{v_{i-1},v_i} \\ &= D_p^{v_0} \circ T^{0,v_{i-1}} \circ T^{v_{i-1},v_i} \\ &= D_p^{v_0} \circ T^{v_0,v_i} \end{aligned}$$

Equation (5.44) follows directly from equation (5.43). □

Remark 5.6. *As function $T^{i,j}$ is the inverse of $T^{j,i}$, the operator $\mathbf{T}^{j,i}$ is the inverse of $\mathbf{T}^{i,j}$.*

We can now reformulate the first equation of Proposition 5.4 as follows:

Proposition 5.5. *Time mapping of departure curve D_p^v*
Let i and j be two nodes on path p.

$$D_p^i = \mathbf{T}^{i,j}(D_p^j) \tag{5.45}$$

Proof: Using definition 5.13 we have,

$$\begin{aligned} \mathbf{T}^{i,j}\left(D_p^j\right) &= D_p^j \circ T^{j,i} = D_p^j \circ T^{j,0} \circ T^{0,i} \\ &= D_p^0 \circ T^{0,i} = D_p^i \end{aligned}$$

□

Proposition 5.6. *Time mapping and flows*
Let v be a node on path p.

$$\lambda_p^i = \mathbf{T}^{i,j}\left(\lambda_p^j\right) \cdot \frac{dT^{j,i}}{dt} \tag{5.46}$$

Proof: From the definition of flow, $\lambda_p^v = \dfrac{dD_p^v}{dt}$. The result is obtained by simply taking the derivative of the equation $D_p^i = D_p^j \circ T^{j,i}$ (from Proposition 5.5) with respect to time. □

Remark 5.7. *The time mapping and derivative operators do not commute.*

Definition 5.14. *Time mapping of delay δ_j^i*
Let v be an internal node[8]*. We define the time mapped delay in queue v at node π_v, $\delta_v^{\pi_v}$ as the delay encountered in queue v by an agent leaving node π_v:*

$$\delta_v^{\pi_v} \doteq \delta_v \tag{5.47}$$

Let i be an arbitrary node and j be an internal node. We define the time mapped delay in queue j at node i, δ_j^i as

$$\delta_j^i \doteq \mathbf{T}^{i,\pi_j}\left(\delta_j^{\pi_j}\right) = \delta_j^{\pi_j} \circ T^{\pi_j,i} \tag{5.48}$$

Physically, if nodes i and j are on the same branch with $i \prec j$ (resp. $i \succ j$), then $\delta_j^i(t)$ is the time that an agent which leaves queue i at time t will be (resp. has been) delayed at in queue j.

Definition 5.15. *Time mapping for capacity*
We define the time mapped capacity of a link l, $\mu_l^{v_l^{in}}$ as the capacity encountered by an agent at queue v_l^{in} in link l

$$\mu_l^{v_l^{in}} \doteq \mu_l \tag{5.49}$$

Let l be an arbitrary link and v an internal node. We define the time mapped capacity of link l at node v as

$$\mu_l^v \doteq \mathbf{T}^{v,v_l^{in}}\left(\mu_l^{v_l^{in}}\right) = \mu_l^{v_l^{in}} \circ T^{v_l^{in},v} \tag{5.50}$$

Physically, if link l and node v are on the same branch with $v_l^{in} \prec v$ (resp. $v_l^{in} \succ v$), then $\mu_v^l(t)$ is the capacity an agent that leaves queue v at time t encountered (resp. encounters) at link l.

Proposition 5.7. *Physical interpretation of mapped delay and mapped capacity*
Let v_j be an arbitrary node, p be a path, and $(v_0, v_1, v_2, \cdots, v_n)$ be a sequence of consecutive nodes on the path p. Also, let $t^{v_i} = T^{v_i,0}\left(t^{v_0}\right), \forall v_i \in p$.

$$\delta_{v_j}^{v_0}\left(t^{v_0}\right) = \delta_{v_j}^{v_1}\left(t^{v_1}\right) = \cdots = \delta_{v_j}^{v_i}\left(t^{v_i}\right) = \cdots = \delta_{v_j}^{v_n}\left(t^{v_n}\right) \tag{5.51}$$

Let l be an arbitrary link.

$$\mu_l^{v_0}\left(t^{v_0}\right) = \mu_l^{v_1}\left(t^{v_1}\right) = \cdots = \mu_l^{v_i}\left(t^{v_i}\right) = \cdots = \mu_l^{v_n}\left(t^{v_n}\right) \tag{5.52}$$

[8] An internal node is a node v which is neither a sink nor the source $k \in K \backslash (\{0\} \cup S)$

Proof: Let i be an arbitrary node and j be an internal node. From definition (5.14) for time mapped delay we have.

$$\begin{aligned}\delta_j^i\left(t^i\right) &\doteq \delta_j^{j-1}\left(T^{j-1,i}\left(t^i\right)\right)\\ &= \delta_j^{j-1}\left(t^{j-1}\right)\end{aligned}$$

Therefore, $\delta_{v_j}^{v_i}\left(t^{v_i}\right) = \delta_{v_j}^{v_{j-1}}\left(t^{v_{j-1}}\right), \forall v_i \in p$, which proves equation (5.51). The proof for equation (5.52) is identical. □

Definition 5.16. *Time mapping of active link and active paths*
Let v be an internal node. We define mapped active link $\gamma_v^{\pi_v}$ as the active link for flow exiting node π_v at queue v, and mapped active paths $\Gamma_v^{\pi_v}$ as the active paths for flow exiting node π_v at queue v.

$$\gamma_v^{\pi_v} \doteq \gamma_v \quad ; \quad \Gamma_v^{\pi_v} \doteq \Gamma_v \tag{5.53}$$

Let j be an arbitrary node, we define the mapped active link and mapped paths for flow exiting queue v at node j as

$$\gamma_v^j = \mathbf{T}^{j,\pi_v}\left(\gamma_v^{\pi_v}\right) \quad ; \quad \Gamma_j^v = \mathbf{T}^{j,\pi_v}\left(\Gamma_v^{\pi_v}\right) \tag{5.54}$$

Physically, if node j and node v are on the same branch with $j \prec v$ (resp. $j \succ v$), then $\gamma_v^j(t)$ is the active link that an agent leaving node j at time t will encounter (resp. encountered) at queue v, and $\Gamma_j^v(t)$ are the corresponding active paths.

Definition 5.17. *Time mapped link constraint*
Let v be a internal node and $l \in L_v^{out}$. We define the mapped link constraint $c_{v,l}^{\pi_v}$ as the link constraint at link l for an agent leaving node π_v.

$$c_{v,l}^{\pi_v}(t) \doteq \frac{\sum_{p\in P_l}\lambda_p^{\pi_v}(t)}{\mu_l\left(t+\delta_v(t)\right)} \tag{5.55}$$

$$= \frac{\sum_{p\in P_l}\lambda_p^{\pi_v}(t)}{\mu_l^v\left(t+\delta_v(t)\right)}$$

$$= \frac{\sum_{p\in P_l}\lambda_p^{\pi_v}(t)}{\mu_l^{\pi_v}(t)} \tag{5.56}$$

Let j be an arbitrary node, we define the mapped link constraint for link l at node j as

$$c_{v,l}^j \doteq \mathbf{T}^{j,\pi_v}\left(c_{v,l}^{\pi_v}\right) = c_{v,l}^{\pi_v} \circ T^{\pi_v,j} \tag{5.57}$$

$$c_{v,l}^j(t) = \frac{\sum_{p\in P_l}\lambda_p^j(t)}{\mu_l^j(t)} \cdot \frac{dT^{\pi_v,j}}{dt} \tag{5.58}$$

Physically, if node j and node v are on the same branch with $j \prec v$ (resp. $j \succ v$), then $c_{v,l}^j(t)$ is the link constraint that an agent leaving node j at time t will encounter (resp. encountered) at link l.

Remark 5.8. *The notation of the link constraint can be simplified for convenience as follows when time mapped.*

$$c_{v,l}^{j} = c_{l}^{j} \tag{5.59}$$

We use the simplified notation in the rest of the discussion.

Proposition 5.8. *The mapping of link constraints and active links is coherent*
For all non-sink nodes $j \in V \backslash S$, internal nodes $v \in V \backslash (S \cup \{0\}$ and time $t \in (t_{initial}, t_{final}]$, we have

$$\gamma_{v}^{j}(t) \in \underset{l \in L_{v}^{out}}{\arg\max}\, c_{l}^{j}(t) \tag{5.60}$$

Proof: Let v be an internal node and let t^j be a time. Let $t^{\pi_v} = T^{\pi_v,j}(t^j)$. Proving the proposition is equivalent to proving the following set equality

$$\underset{l \in L_{v}^{\text{out}}}{\arg\max}\, c_{v,l}(t^{\pi_v}) = \underset{l \in L_{v}^{\text{out}}}{\arg\max}\, c_{l}^{j}\left(t^{j}\right) \tag{5.61}$$

From the definition of the link constraint in equation (5.18) we have

$$c_{v,l}(t^{\pi_v}) \doteq \frac{\sum_{p \in P_l} \lambda_{p}^{\pi_v}(t^{\pi_v})}{\mu_{l}(t^{\pi_v} + \delta_{v}(t^{\pi_v}))} \tag{5.62}$$

By definition of μ_l^v in equation (5.49), we have $\mu_l(t^{\pi_v} + \delta_v(t^{\pi_v})) = \mu_l^v(t^{\pi_v} + \delta_v(t^{\pi_v}))$ and defining $t^v \doteq T^{v,\pi_v}(t^{\pi_v}) = t^{\pi_v} + \delta_v(t^{\pi_v})$, we obtain $\mu_l(t^{\pi_v} + \delta_v(t^{\pi_v})) = \mu_l^v(T^{v,\pi_v}(t^{\pi_v})) = \mu_l^v(t^v)$. Equation (5.52) finally gives

$$\mu_{l}(t^{\pi_v} + \delta_{v}(t^{\pi_v})) = \mu_{l}^{j}\left(t^{j}\right) \tag{5.63}$$

Moreover, using equation (5.46) gives $\lambda_p^{\pi_v}(t^{\pi_v}) \cdot \left.\frac{dT^{\pi_v,j}}{dt}\right|_{t^j} = \lambda_p^j(t^j)$. Summing on all paths p in P_l, we obtain

$$\sum_{p \in P_l} \lambda_{p}^{\pi_v}(t^{\pi_v}) = \frac{1}{\left.\frac{dT^{\pi_v,j}}{dt}\right|_{t^j}} \cdot \sum_{p \in P_l} \lambda_{p}^{j}\left(t^{j}\right) \tag{5.64}$$

Substituting equations (5.63) and (5.64) in the right hand side of equation (5.62) and using the time mapped link constraint from equation (5.58), we obtain

$$c_{v,l}(t^{\pi_v}) = \frac{1}{\left.\frac{dT^{\pi_v,j}}{dt}\right|_{t^j}} \cdot \left[\frac{\sum_{p \in P_l} \lambda_{p}^{j}(t^{j})}{\mu_{l}^{j}(t^{j})}\right] \tag{5.65}$$

$$= \frac{1}{\left.\frac{dT^{\pi_v,j}}{dt}\right|_{t^j}} \cdot c_{l}^{j}\left(t^{j}\right) \tag{5.66}$$

For all $l \in L_v^{\text{out}}$, $c_{v,l}(t^{\pi_v})$ and $c_l^j(t^j)$ are proportional (and the proportionality ratio is independent from l). Therefore, the arg max in equation (5.61) are the same. which concludes the proof. □

Definition 5.18. *Capacity of the active link*
For notational simplicity we denote the capacity of the active link of an agent that enters queue v at time t as follows:

$$\begin{aligned} Q_v(t) &\doteq \mu^{\pi_v}_{\gamma_v(t)}(t) && (5.67)\\ &= \mu^{v}_{\gamma_v(t)}(t+\delta_v(t)) \\ &= \mu_{\gamma_v(t)}(t+\delta_v(t)) && (5.68)\end{aligned}$$

Definition 5.19. *Time mapped capacity of the active link*
Let v be a internal node. We define the time mapped active link capacity $Q^{\pi_v}_v$ as the capacity of link γ_v as seen by an agent at node π_v.

$$Q^{\pi_v}_v \doteq Q_v \tag{5.69}$$

Let j be an arbitrary node, we define the mapped active link capacity for link $\gamma_v(t)$ as seen by an agent at node j as

$$Q^j_v \doteq \mathbf{T}^{j,\pi_v}(Q^{\pi_v}_v) = Q^{\pi_v}_v \circ T^{\pi_v,j} = Q_v \circ T^{\pi_v,j} \tag{5.70}$$

Physically, if node j and node v are on the same branch with $j \prec v$ (resp. $j \succ v$), then $Q^j_v(t)$ is the active link capacity that an agent leaving node j at time t will encounter (resp. encountered) at link $\gamma^j_v(t)$.

Definition 5.20. *Time mapping of queue state*
Let i be an arbitrary node and j be an internal node. We define the time mapped queue state of queue j at node i, η^i_j as the queue state at queue j as seen by an agent at queue i

$$\eta^i_j \doteq \mathbf{T}^{i,\pi_v}\left(\eta^{\pi_v}_j\right) = \eta^{\pi_v}_j \circ T^{\pi_v,i} = \eta_j \circ T^{\pi_v,i} \tag{5.71}$$

Physically, if queue i and node j are on the same branch with $i \prec j$ (resp. $i \succ j$), then $\eta^i_j(t)$ is the queue state an agent that leaves node i at time t encounters (resp. encountered) at queue j.

5.3.3 Global evolution of delay

We now have the necessary tools to define the evolution of delays at any node of the network with respect to the flows at any upstream node in the network.

Definition 5.21. *First active upstream node*
Let v be an internal node. We define the first active upstream node of v as

$$\Upsilon^j_v(t) = \max_{\succ}\left\{u | u \prec v, \eta^j_u(t) = 1\right\} \tag{5.72}$$

For notational convenience we also define the following:

$$\hat{\gamma}_v^j(t) \doteq \gamma^j_{\Upsilon_v^j(t)}(t) \tag{5.73}$$

$$\hat{\Gamma}_v^j(t) \doteq \Gamma^j_{\Upsilon_v^j(t)}(t) \tag{5.74}$$

$$\hat{Q}_v^j(t) \doteq Q^j_{\Upsilon_v^j(t)}(t) \tag{5.75}$$

$$\hat{\eta}_v^j(t) \doteq \eta^j_{\Upsilon_v^j(t)}(t) \tag{5.76}$$

Theorem 5.2. ***Evolution law for delay at an arbitrary internal node*** v ***mapped to any node*** j

Given an arbitrary internal node $v \in V \backslash (S \cup \{0\})$ such that queue v is active, if the flows at the origin are acceptable departure curves and the model requirements are satisfied, the evolution law for delay mapped to any upstream node $j \in V \backslash S$ is

$$\left.\frac{d\delta_v^j}{dt}\right|_t = \begin{cases} \dfrac{\sum_{p\in\Gamma_v^j(t)} \lambda_p^j(t)}{Q_v^j(t)} - \left.\dfrac{dT^{0,j}}{dt}\right|_t & \\ & \text{if } v \text{ is the first active queue} \in p \\ \dfrac{\sum_{p\in\Gamma_v^j(t)} \lambda_p^j(t)}{Q_v^j(t)} - \dfrac{\sum_{p\in\hat{\Gamma}_p^j(t)} \lambda_p^j(t)}{\hat{Q}_v^j(t)} & \text{otherwise} \end{cases} \tag{5.77}$$

Proof: Let t be a time and v be a node. Evolution law (5.23) in Proposition 5.2 gives

$$\left.\frac{d\delta_v}{dt}\right|_t = \frac{\sum_{p\in\Gamma_v(t)} \lambda_p^{\pi_v}(t)}{Q_v(t)} - 1 \tag{5.78}$$

By the definition of the time mapping functions we have, $\delta_v^{\pi_v}(t) \doteq \delta_v(t)$, $Q_v^{\pi_v}(t) \doteq Q_v(t)$, $\Gamma_v^{\pi_v}(t) \doteq \Gamma_v(t)$. Thus, equation (5.78) becomes:

$$\left.\frac{d\delta_v^{\pi_v}}{dt}\right|_t = \frac{\sum_{p\in\Gamma_v^{\pi_v}(t)} \lambda_p^{\pi_v}(t)}{Q_v^{\pi_v}(t)} - 1 \tag{5.79}$$

Case 1: If node v is not the first active node of path p and $\Upsilon_v(t)$ exists.

Let for an arbitrary node j, $t^j = T^{j,\pi_v}(t)$. Since all the nodes between $\Upsilon_v(t)$ and π_v are inactive by the definition of $\Upsilon_v(t)$, we have

$$t^{\Upsilon_v(t^{\pi_v})} = t^{\pi_v} = t \tag{5.80}$$

Furthermore, since $\hat{\eta}_v^{\pi_v}(t) = 1$, and the full capacity discharge of active links (assumption 5.3), we have

$$\sum_{p \in \hat{\Gamma}_v^{\pi_v}(t)} \lambda_p^{\Upsilon_v(t)}(t) = \hat{Q}_v^{\pi_v}(t) \tag{5.81}$$

$$\sum_{p \in \hat{\Gamma}_v^{\pi_v}(t)} \lambda_p^{\pi_v}(t) = \hat{Q}_v^{\pi_v}(t) \tag{5.82}$$

Thus:

$$\frac{\sum_{p \in \hat{\Gamma}_v^{\pi_v}(t)} \lambda_p^{\pi_v}(t)}{\hat{Q}_v^{\pi_v}(t)} = 1 \tag{5.83}$$

By replacing the constant 1 in equation (5.79) with the above result we get,

$$\left.\frac{d\delta_v^{\pi_v}}{dt}\right|_t = \frac{\sum_{p \in \Gamma_v^{\pi_v}(t)} \lambda_p^{\pi_v}(t)}{Q_v^{\pi_v}(t)} - \frac{\sum_{p \in \hat{\Gamma}_v^{\pi_v}(t)} \lambda_p^{\pi_v}(t)}{\hat{Q}_v^{\pi_v}(t)} \tag{5.84}$$

This gives us the result for $j = \pi_v$. We will now map this result to any node $j \in V \backslash S$. By definition of time mapping, we have

$$\delta_v^j = \delta_v^{\pi_v} \circ T^{\pi_v, j} \tag{5.85}$$

Taking its derivative with respect to time, we obtain

$$\frac{d\delta_v^j}{dt} = \left[\frac{d\delta_v^{\pi_v}}{dt} \circ T^{\pi_v, j}\right] \cdot \frac{dT^{\pi_v, j}}{dt} \tag{5.86}$$

$$\left.\frac{d\delta_v^j}{dt}\right|_t = \left[\frac{\sum_{p \in \Gamma_v^{\pi_v} \circ T^{\pi_v, j}(t)} \lambda_p^{\pi_v} \circ T^{\pi_v, j}(t)}{Q_v^{\pi_v} \circ T^{\pi_v, j}(t)} - \frac{\sum_{p \in \Gamma_{\Upsilon_v t}^{\pi_v} \circ T^{\pi_v, j}(t)} \lambda_p^{\pi_v} \circ T^{\pi_v, j}(t)}{Q_{\Upsilon_v(t)}^{\pi_v} \circ T^{\pi_v, j}(t)}\right] \cdot \left.\frac{dT^{\pi_v, j}}{dt}\right|_t \tag{5.87}$$

Equation (5.46) on flow mapping gives

$$\left(\lambda_p^{\pi_v} \circ T^{\pi_v, j}(t)\right) \cdot \left.\frac{dT^{\pi_v, j}}{dt}\right|_t = \lambda_p^j(t) \tag{5.88}$$

Substituting this result and the simple time mapping transformations of λ and Q into equation (5.87) gives the final result

$$\left.\frac{d\delta_v^j}{dt}\right|_t = \frac{\sum_{p \in \Gamma_v^j(t)} \lambda_p^j(t)}{Q_v^j(t)} - \frac{\sum_{p \in \Gamma_{\Upsilon_v(t)}^j(t)} \lambda_p^j(t)}{Q_{\Upsilon_v(t)}^j(t)} \tag{5.89}$$

$$= \frac{\sum_{p \in \Gamma_v^j(t)} \lambda_p^j(t)}{Q_v^j(t)} - \frac{\sum_{p \in \hat{\Gamma}_v^j(t)} \lambda_p^j(t)}{\hat{Q}_v^j(t)} \tag{5.90}$$

Case 2: If node v is the first active node of path p, we leave the constant 1 in equation (5.79) and follow the same remaining steps as in case 1 to obtain the result.

$$\left.\frac{d\delta_v^j}{dt}\right|_t = \frac{\sum_{p\in\Gamma_v^j(t)}\lambda_p^j(t)}{Q_v^j(t)} - \left.\frac{dT^{0,j}}{dt}\right|_t \tag{5.91}$$

□

Applying Theorem 5.2 with $j = 0$, we see that the delays with respect to the flows at the origin δ_v^0 are solutions to the ordinary differential equations in definition 5.22.

Definition 5.22. *Time mapped delay evolution differential equation*

- *If v is not an active node and the flow on its active link γ_0^v is within capacity, then* $\frac{d\delta_v^0}{dt} = 0$.
- *If v is an active node or its active link γ_0^v is over capacity, then*

$$\left.\frac{d\delta_v^0}{dt}\right|_t = \begin{cases} \dfrac{\sum_{p\in\Gamma_v^0(t)}\lambda_p^0(t)}{Q_v^0(t)} - 1 \\ \qquad\qquad \text{if } v \text{ is the first active queue} \in p \\ \dfrac{\sum_{p\in\Gamma_v^0(t)}\lambda_p^0(t)}{Q_v^0(t)} - \dfrac{\sum_{p\in\hat{\Gamma}_v^0(t)}\lambda_p^0(t)}{\hat{Q}_v^0(t)} \quad \text{otherwise} \end{cases} \tag{5.92}$$

where the time mapping functions are redefined from delays as follows:

$$T^{j,0} = \sum_{0\prec i\preccurlyeq j}\delta_i^0 \tag{5.93}$$

Proposition 5.9. *Delay evolution does not depend on departure curves*
All the time mapped quantities in equations (5.92) *can be computed using only the initial delays, departure curve at the origin and the link capacities. It does not require the departure curves for any internal nodes $v \in V\backslash v_0$.*

Proof: The time mapping function only depends on the delay functions from definition 5.9. The time mapped flows can be obtained using the time mapping function using Proposition 5.6. The other time mapped quantities are by definition constructed using the time mapping function as given in section 5.3.2. □

5.3.4 Equivalence of departure curves and delays

We prove Theorem 5.1 on the existence and uniqueness of Problem 1 by first showing the equivalence between Problem 1 and Problem 2 (defined below), and then proving the existence and uniqueness of Problem 2 in the next section.

Problem 2: General delay problem
Input. An arborescence (V, L) with source v_0 and sink set S, capacities $\mu_l(t), \forall l \in L,\ t \in$

$[t_{\text{initial}}, t_{\text{final}}]$, departure functions from the source $D_p^{v_0} \in \mathbf{D}(t_{\text{initial}}, t_{\text{final}})\ \forall p \in P_{v_0}$ and initial delays $\delta_v(t_{\text{initial}}) \geq 0,\ \forall v \in V \backslash (S \cup \{v_0\})$
Question. Does a solution to the time mapped delay function from definition 5.22 for each node $v \in V \backslash v_0$ exist and is it unique?

Theorem 5.3. *Problem (1) and problem (2) are equivalent*

Proof. The inputs to both problems are identical. Therefore, we only need to prove that the existence of a solution to one problem implies a unique and feasible corresponding solution to the other problem.

($\Rightarrow$) Suppose first that Problem 1 admits a solution.
By the definition of delay,

$$\delta_v(t) = [D_p^v]^{-1}\left(D_p^{\pi_v}(t)\right) - t, \tag{5.94}$$

By the definition of time mapped delay we obtain,

$$\begin{aligned}
\delta_v^0(t) &= \delta_v^{\pi_v}(t) \circ T^{\pi_v,0}(t) && (5.95)\\
&= \delta_v^{\pi_v}\left([D_p^v]^{-1}\left(D_p^{\pi_v}(t)\right) - [D_p^0]^{-1}\left(D_p^{\pi_v}(t)\right)\right) && (5.96)\\
&= \delta_v\left([D_p^v]^{-1}\left(D_p^{\pi_v}(t)\right) - [D_p^0]^{-1}\left(D_p^{\pi_v}(t)\right)\right), && (5.97)
\end{aligned}$$

which can be made a function of only D_p^v by equation (5.94).

Theorem 5.2 then ensures that the delay functions thus defined satisfy the time mapped delay evolution from definition 5.22, i.e. a feasible solution to problem 2. Furthermore, the solution is unique from equation (5.97), since D_p^v is a strictly increasing function.

($\Leftarrow$) Suppose now that Problem 2 admits a solution $\delta_v^0(t)$. We can build the corresponding departure curves $D_p^v(t)$ as follows.

$$D_p^0(t) = \delta_0^0(t) \tag{5.98}$$

The inverse departure curve $[D_p^0]^{-1}(x)$ can be constructed from $D_p^0(t)$, since the departure curve is strictly increasing.

$$[D_p^v]^{-1}(x) = [D_p^0]^{-1}(x) + T^{v,0}\left([D_p^0]^{-1}(x)\right) \tag{5.99}$$

The departure curve $D_p^v(t)$ can also be constructed from $[D_p^v]^{-1}(x)$ due to the strictly increasing nature of the functions.

We now show that the departure curves thus defined are feasible departure curves, i.e. a feasible solution to problem 1.

1. D_p^0 is continuous and piecewise C_1 because λ_p^0 is piecewise continuous. Furthermore, since $T^{j,0}$ is strictly increasing for all nodes j, D_p^v is continuous and piecewise C_1.
2. The capacity constraint on links is imposed by equation (5.92) due to Proposition 5.8.
3. The FIFO condition is satisfied by construction since the delay δ_v^0 is not a function of the path p.
4. The full capacity discharge of the active queues is also imposed by by equation (5.92) due to Proposition 5.8.

□

5.3.5 Existence and uniqueness of the time mapped delay evolution

This section proves Theorem 5.4 on the existence and uniqueness of the solution to Problem 2.

Theorem 5.4. ***Existence and uniqueness of the solution to problem (2)***
The solution to problem (2) exists and is unique on the time interval of the problem $[t_{initial}, t_{final}]$*, if the following conditions are satisfied.*

1. *the path flows at the origin* $\lambda_p^0(t)$ *are piecewise polynomial,*
2. *link capacities* μ_l *are piecewise constant over time.*

The proof of this theorem is fairly technical and requires several definitions and lemmas. Theorem 5.1 is a direct corollary of this result due to Theorem 5.3 on the equivalence of the two problems.

The main goal of the proof of Theorem 5.4 is to show that there are a finite number of possible transitions, and to integrate equation (5.92) across the transitions. The next definitions and lemmas enables to establish these properties.

Definition 5.23. *Depth of a node* $d(v)$
We define the depth $d(v)$ *of a node* v *as the number of links on the unique path from the origin* v_0 *to node* v

Definition 5.24. *Link constraint comparators* $B_{(c_{l_1},c_{l_2})}(t)$ *and* $B_{c_l}(t)$
Given a node v *and two distinct links* (l_1, l_2)*, we define the boolean comparator* $B_{(c_{l_1},c_{l_2})}(t)$ *as follows:*

$$B_{(c_{l_1},c_{l_2})}(t) = \begin{cases} 1 & \text{if } \frac{\sum_{p\in P_{l_1}} \lambda_p^0(t)}{\mu_{l_1}^0(t)} > \frac{\sum_{p\in P_{l_2}} \lambda_p^0(t)}{\mu_{l_2}^0(t)} \\ 0 & \text{otherwise} \end{cases} \tag{5.100}$$

Given a node v *and link* $l \in L_v^{out}$*, we define the boolean comparator* $B_{c_l}(t)$ *as follows:*

$$B_{c_l}(t) = \begin{cases} 1 & \text{if } \frac{\sum_{p\in P_l} \lambda_p^0(t)}{\mu_l^0(t)} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.101}$$

Definition 5.25. *Time segment of constant link constraint J*
A time segment J is a segment of constant link constraint if and only if

1. *for each each $l \in L$, the boolean $B_{c_l}(t)$ is constant on J,*
2. *for each each pair of nodes $(l_1, l_2) \in L$, the boolean $B_{(c_{l_1}, c_{l_2})}(t)$ is constant on J.*
3. *for each each $l \in L$, the time mapped link capacity $\mu_l^0(t)$ is constant on J.*

Lemma 5.3. *Under the assumptions on flows and capacities, there are a finite number of segments of constant link constraint*

Proof: Consider a pair of links (l_1, l_2). Since capacities are piecewise constant and flows are piecewise polynomial, there are a finite number of segments on which the capacities are constant and flows are polynomial. On any such a segment, $\frac{\sum_{p \in P_{l_1}} \lambda_p^0(t)}{\mu_{l_1}^0(t)}$ and $\frac{\sum_{p \in P_{l_1}} \lambda_p^0(t)}{\mu_{l_1}^0(t)} - \frac{\sum_{p \in P_{l_2}} \lambda_p^0(t)}{\mu_{l_2}^0(t)}$ are polynomials. Therefore, the number of times each expression crosses zero is bounded by the degree of the polynomial, which implies that there are a finite number of segments of constant link constraint. □

Lemma 5.4. *Constant active link*
If J is a segment of constant link constraint, the active link γ_v^0 of any node v is constant on J.

Proof: The result comes directly from the definition of a segment of constant constraint. □

Definition 5.26. *Solution of depth n*
A solution of problem (2) for depth n is a set of solutions δ_v for all nodes v such that $d(v) < n$. It can be rigorously defined because the equations for δ_v only depend on variables associated with nodes of depth less than n.

Definition 5.27. *Elementary time segment $T^e(v)$*
Given a node v and a solution of depth $d(v) - 1$ (if v is not the origin), an elementary segment for node v is a time segment $T^e(v)$ such that

- *$T^e(v)$ is a segment of constant constraint,*
- *If v is not the origin, for each node $j \in V$ such that $d(j) < d(v)$, the node state $\eta_j(t)$ is constant on $T^e(v)$.*

Lemma 5.5. *Single transition of node state on an elementary segment*
If there exists a solution to problem (2) up to depth $d(v-1)$, and if $T^e(v) = [t_0, t_f]$ is an elementary segment for node v, then there is a solution δ_v^0 of the problem and node v admits at most one transition in $T^e(v)$.

Proof: As for each node $j \in V$ such that $d(j) < d(v)$, the node state $\eta_j(t)$ is constant on $T^e(v)$, the first active upstream node Υ_v is constant over time. Moreover, as $T^e(v)$ is a segment of constant constraint, Lemma 5.4 gives that active link γ_v and first active upstream link $\hat{\gamma}_v$ are constant on $T^e(v)$, and the sign of $\frac{\sum_{p\in\Gamma_v^0(t)}\lambda_p^0(t)}{Q_v^0(t)} - \frac{\sum_{p\in\hat{\Gamma}_v^0(t)}\lambda_p^0(t)}{\hat{Q}_v^0(t)}$ is constant on $T^e(v)$.

Let us now consider the following four cases:

1. $B_{(c_{\hat{\gamma}_v}, c_{\gamma_v})}(t_0) = 1, \eta_v^0(t) = 1$
   $\implies \left.\frac{d\delta_v^0}{dt}\right|_t > 0$ and since the queue state is already active no transition will occur.

2. $B_{(c_{\hat{\gamma}_v}, c_{\gamma_v})}(t_0) = 1, \eta_v^0(t) = 0$
   $\implies \left.\frac{d\delta_v^0}{dt}\right|_t > 0$ and the queue state will immediately transition to being active $\eta_v^0(t) = 1$. No further transitions will occur as shown above.

3. $B_{(c_{\hat{\gamma}_v}, c_{\gamma_v})}(t_0) = 0, \eta_v^0(t) = 1$
   $\implies \left.\frac{d\delta_v^0}{dt}\right|_t \leq 0$ and the queue at node v starts dissipating. There will be a transition in the queue state to inactive $\eta_v^0(t) = 0$ if the queue dissipates by time t_f and the queue state will remain active otherwise.

4. $B_{(c_{\hat{\gamma}_v}, c_{\gamma_v})}(t_0) = 0, \eta_v^0(t) = 0$
   $\implies \left.\frac{d\delta_v^0}{dt}\right|_t \leq 0$ and the only possibility is the strict equality case and the queue state remains inactive.

□

Lemma 5.6. *Unique solution on an elementary segment*
Let $T^e(v)$ be an elementary segment for node v. Assuming a solution of depth $d(v)-1$ (if v is not the origin), then solution of equation (5.92) for node v exists is unique on $T^e(v)$.

Proof: By Lemma 5.5, there can be at most one state transition of node v in $T^e(v)$. This splits $T^e(v)$ into at most two sub-segments where $\eta_v = 0$ or $\eta_v = 1$. From Lemma 5.4 we have that active link γ_v and the first active upstream link Υ_v are constant on $T^e(v)$. Therefore, the quantities $\Gamma_v, \hat{\Gamma}_v, Q_v$ and $\hat{Q}_v$ are constant on $T^e(v)$. Equation (5.92) states that

- If v is not an active node ($\eta_v = 0$) and the flow on its active link γ_v is within capacity, then $\frac{d\delta_v^0}{dt} = 0$.
- If v is an active node ($\eta_v = 1$) or it's active link γ_v is over capacity,

$$\left.\frac{d\delta_v^0}{dt}\right|_t = \begin{cases} \dfrac{\sum_{p\in\Gamma_v^0(t)}\lambda_p^0(t)}{Q_v^0(t)} - 1 \\ \qquad\qquad \text{if } v \text{ is the first active queue} \in p \\ \dfrac{\sum_{p\in\Gamma_v^0(t)}\lambda_p^0(t)}{Q_v^0(t)} - \dfrac{\sum_{p\in\hat{\Gamma}_v^0(t)}\lambda_p^0(t)}{\hat{Q}_v^0(t)} \text{ otherwise} \end{cases} \tag{5.102}$$

As all the variables in equation (5.102) other than the flow $\lambda_p^0(t)$ are constant during an elementary segment $T^e(v)$ and the flow $\lambda_p^0(t)$ is continuous in t for all $t \in T^e(v)$, we can show that equation (5.102) admits a unique solution on the interval $T^e(v)$ by the Picard-Lindelöf theorem. □

We have now all the ingredients to prove Theorem 5.4.

Theorem 5.4: The time interval of interest $(t_{initial}, t_{final}]$ can be partitioned into a finite set of elementary segments, and the solution to problem (2) exists and is unique

Proof: The proof is done inductively over the depth of the network. If the network contains a single node v_0, $[t_{\text{initial}}, t_{\text{final}}]$ is an elementary segment for v_0, $(t_{\text{initial}}, t_{\text{final}}] \in T^e(v_0)$ and there is a unique solution by Lemma 5.6. By the induction hypothesis, let us now assume that $(t_{\text{initial}}, t_{\text{final}}]$ can be partitioned into a finite number of elementary segments with respect to all nodes of depth n and that the solution exists and is unique. Let $t_0, t_1, \cdots, t_m$ be times such that $E_n = \{(t_i, t_{i+1}], \forall i \in [0, m-1]\}$ is the set of elementary segments for nodes of depth n, and let δ_v for all $v \in \{V | d(v) \leq n\}$ be the unique solution of depth n.

Let K_n be the non-empty set of nodes of depth n, and let $v \in K_n$ be a node in this set. Lemma 5.5 gives that for each $v \in K_n$, there is at most one state transition on $(t_i, t_{i+1}]$. Let $F_n(v)$ be the set of times at which these transitions occur for node v. Since there are m elementary segments, there can at most be $|F_n(v)| \leq m$ transitions. If F_n is the set of times at which the transitions for all nodes of depth n happen, $|F_n| \leq m \cdot K_n$.

Let $\{t'_0, t'_1, \cdots, t'_{m'}\} = \{t_0, t_1, \cdots, t_m\} \cup F_n$ be the m' segments created by splitting E_n at each of the state transitions for nodes of depth n. The total number of segments m' satisfies $m' \leq m \cdot (K_n + 1)$, since $|F_n| \leq m \cdot K_n$. By the definition of the t'_i, for each $i \in [0, m']$ we have

- for all $v \in K_n$, η_v is constant on $(t'_i, t'_{i+1}]$,
- $(t'_i, t'_{i+1}]$ is a segment of constant constraint J, since it is subset of an elementary segment, which is already by definition a segment of constant constraint.

Thus, $[t'_i, t'_{i+1}]$ is an elementary segment for all nodes of depth $(n+1)$. Furthermore, by Lemma 5.6, this implies that there is an unique solution to all nodes of depth $n+1$, which concludes the proof. □

This also completes the proof of Theorem 5.1.

Theorem 5.1: Problem 1 admits a unique solution under the following conditions.
1) the path flows at the origin $\lambda_p^0(t)$ are piecewise polynomial,
2) link capacities μ_l are piecewise constant over time.

Proof: Problem 1 is equivalent to Problem 2 by Theorem 5.3 and Problem 2 admits a unique solution by Theorem 5.4. □

In some applications, it is also important to be able to computing the total delay experienced by an agents that takes a particular path. 5.3.6 provides analytical expressions for the total delay along a path.

5.3.6 Total path delay

In some applications, it is also important to be able to computing the total delay experienced by an agents that takes a particular path. In this section, we provide analytical expressions for the total delay along a path.

Definition 5.28. *Total delay of a path p*
We define the total delay Δ_p^0 encountered on a path p at time t as the total delay encountered by agent on path p that enters on the network at t throughout its entire path to the sink node.

$$\Delta_p^0(t) = [D_p^{v_{p_N}}]^{-1}\left(D_p^0(t)\right) - t \tag{5.103}$$

where v_{p_N} is the last non-sink node on path p. We define the time mapped total delay Δ_p^j as the total delay in path p as seen by an agent that is at node j at time t.

$$\Delta_p^j = \mathbf{T}^{j,0}\left(\Delta_p^0\right) \tag{5.104}$$

Proposition 5.10. *Total delay Δ_p^j as a function of queue delay δ*
The time mapped total delay Δ_p^j encountered on a path is equal to the sum of delay encountered by the agent on its path.

$$\Delta_p^j\left(t^j\right) = \sum_{v \in V_p \backslash (\{0\} \cup S)} \delta_v^j\left(t^j\right) \tag{5.105}$$

where t^j is the time that the agent is at node j.

Proof. Let $t^i = T^{i,j}(t^j)$. We obtain the result as follows using the definition of delay and a series of time mappings.

$$\begin{aligned}
LHS &= \Delta_p^j\left(t^j\right) \\
&= \mathbf{T}^{j,0}\left(\Delta_p^0\left(t^j\right)\right) \\
&= \Delta_p^0\left(T^{0,j}\left(t^j\right)\right) \\
&= \Delta_p^0\left(t^0\right) \\
&= [D_p^{v_{p_N}}]^{-1}\left(D_p^0\left(t^0\right)\right) - t^0
\end{aligned}$$

$$\begin{aligned}
RHS &= \sum_{v\in V_p\setminus(\{0\}\cup S)} \delta_v^j\left(t^j\right) \quad = \sum_{v\in V_p\setminus(\{0\}\cup S)} \delta_v^{\pi_v}\left(t^{\pi_v}\right) \\
&= \sum_{v\in V_p\setminus(\{0\}\cup S)} [D_p^v]^{-1}\left(D_p^{\pi_v}\left(t^{\pi_v}\right)\right) - t^{\pi_v} \\
&= \sum_{v\in V_p\setminus(\{0\}\cup S)} [D_p^v]^{-1}\left(D_p^{\pi_v}\left(t^{\pi_v}\right)\right) - [D_p^{\pi_v}]^{-1}\left(D_p^{\pi_v}\left(t^{\pi_v}\right)\right) \\
&= \sum_{v\in V_p\setminus(\{0\}\cup S)} [D_p^v]^{-1}\left(D_p^0\left(t^0\right)\right) - [D_p^{\pi_v}]^{-1}\left(D_p^0\left(t^0\right)\right) \\
&= [D_p^{v_{p_N}}]^{-1}\left(D_p^0\left(t^0\right)\right) - [D_p^0]^{-1}\left(D_p^0\left(t^0\right)\right) \\
&= [D_p^{v_{p_N}}]^{-1}\left(D_p^0\left(t^0\right)\right) - t^0
\end{aligned}$$

□

Definition 5.29. *Active link of the last active queue of a path p at time t ($a_p(t)$)*
Let p be a path and t be the time that an agent departs node j. We define the last active queue of the path p time mapped to passing node j at time t as

$$a_p^j(t) = \max_{\succ}\left\{v \in V_p | \eta_v^j(t) = 1\right\} \tag{5.106}$$

For notational convenience we also define the following:

$$\tilde{\gamma}_p^j(t) = \gamma_{a_p^j(t)}^j(t) \tag{5.107}$$

$$\tilde{\Gamma}_p^j(t) = \Gamma_{a_p^j(t)}^j(t) \tag{5.108}$$

$$\tilde{Q}_p^j(t) = \mu_{\tilde{\gamma}_p^j(t)}^j(t) \tag{5.109}$$

Theorem 5.5. ***Evolution law for total delay Δ_p^0***
Let p be a path, t be a time. The evolution law for total delay at time t is

$$\left.\frac{d\Delta_p^0}{dt}\right|_t = \begin{cases} \dfrac{\sum_{p'\in\tilde{\Gamma}_p^0(t)} \lambda_{p'}^0(t)}{\tilde{Q}_p^0(t)} - 1 & \text{if } p \text{ has an active queue} \\ 0 & \text{otherwise} \end{cases} \tag{5.110}$$

Proof: Taking the derivative of equation (5.105) for $j = 0$, we obtain

$$\left.\frac{d\Delta_p^0}{dt}\right|_t = \sum_{v \in V_p \setminus (S \cup \{0\})} \left.\frac{d\delta_v^0}{dt}\right|_t \tag{5.111}$$

$$= \sum_{\{v | v \in V_p \setminus (S \cup \{0\}), \gamma_v^0(t)=1\}} \left.\frac{d\delta_v^0}{dt}\right|_t \tag{5.112}$$

Note that $\gamma_v^0(t) = 1$ implies node v is active when the source flow at time t reaches node v. From Theorem 5.2 with $j = 0$ we have,

$$\left.\frac{d\delta_v^0}{dt}\right|_t = \begin{cases} \dfrac{\sum_{p \in \Gamma_v^0(t)} \lambda_p^0(t)}{Q_v^0(t)} - 1 & \\ & \text{if } v \text{ is the first active queue} \in p \\ \dfrac{\sum_{p \in \Gamma_v^0(t)} \lambda_p^0(t)}{Q_v^0(t)} - \dfrac{\sum_{p \in \hat{\Gamma}_v^0(t)} \lambda_p^0(t)}{\hat{Q}_v^0(t)} & \text{otherwise} \end{cases} \tag{5.113}$$

Plugging this into equation (5.112) gives a telescopic series, since it only considers the active nodes of the path and $\tilde{Q}_p^0(t)$ gives the capacity of the last active link of path p. Thus, we obtain

$$\left.\frac{d\Delta_p^0}{dt}\right|_t = \frac{\sum_{p' \in \tilde{\Gamma}_p^0(t)} \lambda_{p'}^0(t)}{\tilde{Q}_p^0(t)} - 1 \tag{5.114}$$

If p does not contain an active queue there is no queuing in the path, which means there is no change in the queue length and therefore no change in the delay. □

Remark 5.9. *Note that this theorem can be extended to any subpath $p_{ij} \in p$ such that*

$$\left.\frac{d\Delta_{p_{i,j}}^i}{dt}\right|_t = \frac{\sum_{p' \in \tilde{\Gamma}_{p_{ij}}^i(t)} \lambda_{p'}^i(t)}{\tilde{Q}_{p_{ij}}^i(t)} - 1 \tag{5.115}$$

5.4 Applications

The solution to problem (1) models the flows in the network given the departure time functions at the origin and the initial delays by providing the departure time functions for all the other nodes in the network. The solution can be obtained by first solving problem (2), which provides the agent delay function at each node. Practically, problem (2) easier to directly solve than problem (1), because it corresponds to an explicit automaton that is easy to implement for numerical simulations.

Given a discretization time step Δt and the initial conditions $\delta_v(0)$, algorithm 5.1 gives a numerical solution to the discretized problem (2). The algorithm numerically integrates the

ordinary differential equation (ODE) given in equation (5.92) over time to obtain the solution. The algorithm relies on the fact that each discretized time step is an elementary segment, because the path flows and capacities are assumed to be constant (discrete approximation) during each time step.

Algorithm 5.1 Calculate approximate solution of problem (2)

solveDelays(*sourceFlow:* λ^0, *initialDelays:* $\delta^0[0]$, *capacities:* μ)
  for $l \in L_0^{\text{out}}$ **do**
    for $t = 1$ to T **do**
      update$(v_l^{\text{out}}, t, 1, 0)$
    end for
  end for

update(*node:* v, *timeStep:* t, *lastActiveConstraint:* $\hat{\omega}$)
  if $v \notin S$ **then**
    $\Delta^0_{0,v}[t] = \Delta^0_{0,\pi_v}[t] + \delta^0_v[t-1]$
    for $l \in L_v$ **do**
      $\mu_l^0[t] = \mu_l(t + \Delta^0_{0,v}[t])$
      $c_l^0[t] = \dfrac{\sum_{p \in P_l} \lambda_p^0[t]}{\mu_l^0[t]}$
    end for
    $\gamma_v[t] = \arg\max_{l \in L_v^{\text{out}}} c_{v,l}(t)$
    $\Gamma_v[t] = P_{\gamma_v(t)}$
    $\omega_v[t] = \dfrac{\sum_{p \in \Gamma_v[t]} \lambda_p^0[t]}{\mu_l^0[t]}$
    $\delta_v^0[t] = max\,(0,\ \ (\omega_v - \hat{\omega}) \cdot \Delta t)$
    for $l \in L_v^{out}$ **do**
      if $\delta_v^0[t] > 0$ **then**
        $update(v_l^{\text{out}},\ t,\ \omega_v)$
      else
        $update(v_l^{\text{out}},\ t,\ \hat{\omega})$
      end if
    end for
  end if

5.4.1 Single route with multiple bottlenecks

The first case we will study is that of a simple single path network with multiple queues due to several capacity bottlenecks, as illustrated in figure 5.3. This network can be modeled as a tree with a single sink, i.e. a single path. Thus, we will remove the path index from the

notation in this section. Each internal node v has a unique child, thus the internal nodes can be indexed by the integers $v_0, \cdots, {}_v\, n$ and the unique path of the tree is $[v_0, {}_v\, 1, \cdots, v_n, v_s]$. Moreover, as they model a succession of queues on the same road, we can assume that the capacity of each link (v_i, v_{i+1}) is constant and equal to capacity of the corresponding road segment $\forall v, \mu_{v_i,v_{i+1}} = \mu$. From Theorem 5.5, we know that the evolution of delay is given by

$$\left.\frac{d\Delta_p^0}{dt}\right|_t = \begin{cases} \frac{\sum_{p' \in \tilde{\Gamma}_p^0(t)} \lambda_{p'}^0(t)}{\tilde{Q}_p^0(t)} - 1 & \text{if p has an active queue} \\ 0 & \text{otherwise} \end{cases} \tag{5.116}$$

Since, the link with the smallest capacity will always be the last active link $\tilde{\mu} = \min(\mu_{v_i,v_{i+1}})$:

$$\left.\frac{d\Delta^0}{dt}\right|_t = \begin{cases} \frac{\lambda^0(t)}{\tilde{\mu}} - 1 & \text{if there is an active queue} \\ 0 & \text{otherwise} \end{cases} \tag{5.117}$$

Thus, the evolution of total delay is equivalent to the evolution law for one queue of capacity $\tilde{\mu}$, and the network can be simplified to a unique internal node v followed by a link of capacity $\tilde{\mu}$.

If the capacity of the links is time varying and $\bar{\mu}(t)$ is the capacity of the most constrained link that the agent entering the network at time t is subjected to,

$$\left.\frac{d\Delta^0}{dt}\right|_t = \begin{cases} \frac{\lambda^0(t)}{\bar{\mu}(t)} - 1 & \text{if p has an active queue} \\ 0 & \text{otherwise} \end{cases} \tag{5.118}$$

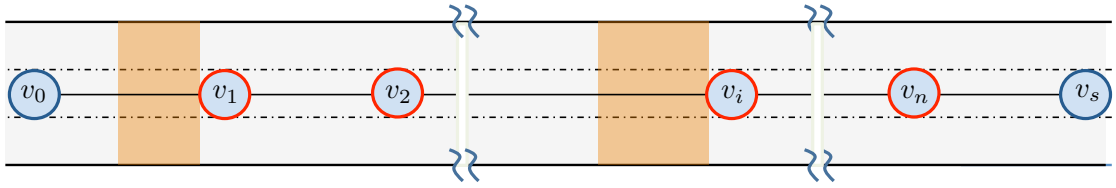


Figure 5.3: MultipleBottlenecks on a road.

5.4.2 Off-Ramp bottleneck

The next application is to compute the the dynamics of a congested freeway off-ramp, using the off-ramp model presented by Newell [101]. This example shows the versatility of our framework, since Newell's the model includes non-FIFO dynamics at the off-ramp. This is accommodated by introducing an additional node and state dependent capacities on two links. The description of the model is as follows. As seen in figure 5.4(a), there are two flows

λ_h and λ_e that enter the network, which has a capacity of μ_h. Therefore, $\lambda_h(t) + \lambda_e(t) \leq \mu_h$. The exiting flow λ_e is restricted by a capacity constraint of μ_e at the exit. There are four possible states of queuing dynamics that can occur based on the flow values. Figure 5.5 illustrates the transitions between the states.

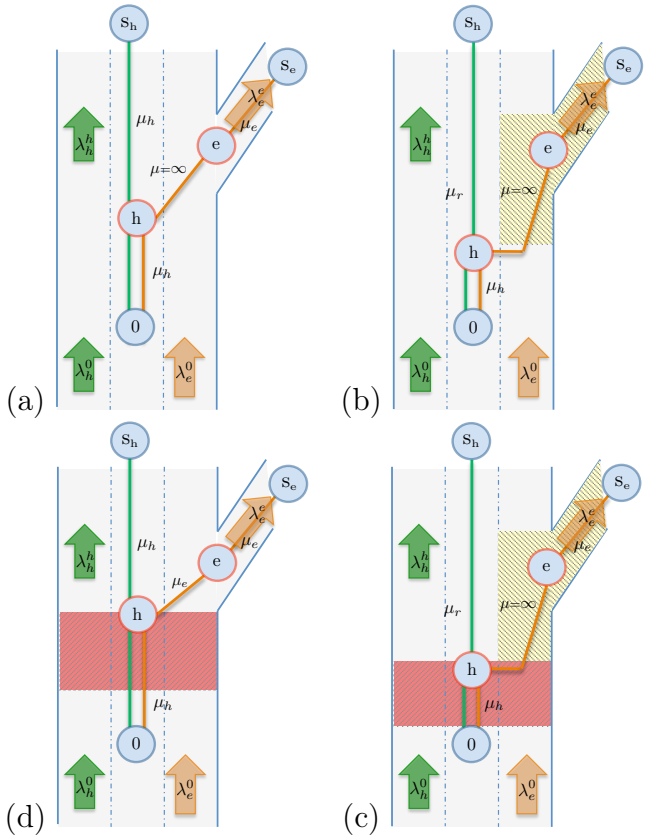


Figure 5.4: Off-Ramp model - (a) state 00 (b) state 01 (a) state 10 (a) state 11

Case 1: $\lambda_e \leq \mu_e$. If $\lambda_e(t) \leq \mu_e$, no queues will form in the network and there will be no delay.
Case 2: $\lambda_e > \mu_e$ and $\lambda_h \leq \mu_r$. If $\lambda_e(t) > \mu_e$, an exit queue will start forming at the entrance to the exit as seen in figure 5.4(b), which will then restrict the capacity of the freeway from μ_h to μ_r.
Case 3: $\lambda_e > \mu_e$, $\lambda_h > \mu_r$ and $\frac{\mu_r}{\lambda_h} \cdot \lambda_e \geq \mu_e$. If the freeway flow $\lambda_h > \mu_r$, then a second freeway queue will start forming behind the exiting agent queue, as seen in figure 5.4(c), since the freeway demand is greater than the new reduced freeway capacity μ_r. This second freeway queue will contain both freeway and exiting agents and therefore the flow exiting the queue will be subject to the *first-in-first-out* (FIFO) condition. As a result, since the freeway flow λ_h is restricted to a rate of μ_r, the exiting agent flow at the freeway queue will be restricted to $\lambda'_e = \frac{\mu_r}{\lambda_h} \cdot \lambda_e$.
Case 4: $\lambda_e > \mu_e$, $\lambda_h > \mu_r$ and $\frac{\mu_r}{\lambda_h} \cdot \lambda_e < \mu_e$. Now, if $\lambda'_e < \mu_e$, then the off-ramp queue

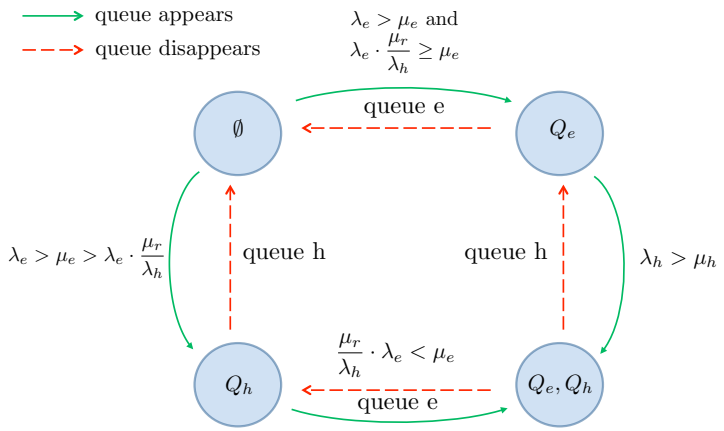


Figure 5.5: State transitions in the off-ramp model. The four states $\emptyset$, Q_e, (Q_e, Q_h) and Q_h correspond respectively to the cases (a), (b), (c) and (d) from figure 5.4.

will start decreasing since the flow is less than the capacity and the queue will disappear. Thus, in this case, an off-ramp bottleneck created a second bottleneck that in turn removed the off-ramp bottleneck, which is an unstable equilibrium. Therefore, as explained in [101], there will be a single queue of both freeway and exiting agents that occurs at the off-ramp, as seen in figure 5.4(d), and the freeway flow through the bottleneck will be $\lambda_h^{out} = \frac{\mu_e}{\lambda_e} \cdot \lambda_h$ according to the FIFO condition.

The uniqueness and existing properties hold even with the state dependent capacities, since the flows are assumed to be piecewise polynomial and therefore lead to a finite number of state transitions. This implies that the link capacities are piecewise constant. Therefore, we can solve for the delays in this network using algorithm 5.1. Furthermore, this subnetwork can be part of a larger network over which we wish to compute the system delays.

Figure 5.6 shows the flow and delay profiles for a numerical example of the off ramp network with the following link capacities: $\mu_E = 5, \mu_H = 30$ and $\mu = 45$. We can observe the following state transitions during the simulated time window.

- At `t=92` Appearance of exiting agent queue.
- At `t=121` Appearance of freeway queue.
- At `t=222` Disappearance of exiting agent queue.
- At `t=372` Disappearance of freeway queue.

One interesting observation is that freeway congestion caused by the exiting agent bottleneck persists well beyond the time at which the exiting agent queue disappears.

The next chapter utilizes the delay analysis framework presented in this chapter to compute the user equilibrium departure time allocation at a freeway off-ramp.

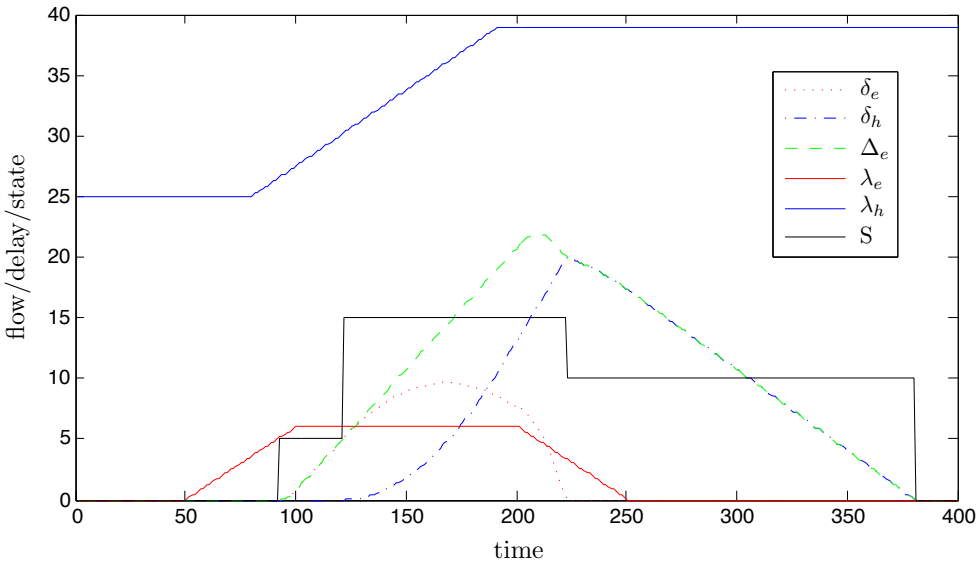


Figure 5.6: Simulation of states and delays (δ_E, δ_H) as functions of time t, given the incoming flows at the off ramp, and road parameters: $\mu_E = 5, \mu_H = 30$ and $\mu = 45$

Chapter 6

Solving the user equilibrium departure time problem at an off-ramp with incentive compatible cost functions

6.1 Introduction

In this chapter, we consider the problem of finding an equilibrium of departure times for a group of vehicles that travel through a capacity restricted network, also known as the morning commute problem. The vehicles share a common desired arrival time and incur a cost for both queuing delays and not meeting their desired arrival time, with penalties being imposed for both early and late arrivals. We wish to determine a set of departure times, such that they form a user equilibrium with respect to the total cost incurred by each vehicle. A brief introduction to the morning commute problem is given in Section 1.4. In particular, we consider the morning commute problem at an under-capacitated off-ramp, which is a common occurrence during the morning rush-hour, at freeway exits with heavy demands. Spill-back from an under-capacitated off-ramp can block the freeway and reduce the freeway capacity available for vehicles that are passing through, and thereby lead to additional delays for these vehicles.

This problem can be addressed by either building additional capacity at the off-ramp to accommodate the peak flow or by using demand management strategies such as tolling. Adding new capacity requires construction work that is extremely disruptive to the network in the short term and incurs a large monetary cost. In addition, the peak flow demands can change rapidly due to many reasons, but the road capacities can not be altered rapidly to adapt to these demand changes. Therefore, in this chapter, we study the ability to manipulate the equilibrium departure-times of the off-ramp demand by augmenting the arrival-time cost function, with tolls or incentives, and thereby mitigate the negative impact on the freeway capacity.

Existing solutions to the morning commute problem assume that the arrival time cost

function of the exiting vehicles is convex and continuous. However, in practice, it is difficult to implement an incentive or tolling strategy where the value of the incentive or toll is continuous in time. These values are most likely to be piece-wise constant. Therefore, the arrival-cost function of the exiting vehicles can no longer be assumed to be continuous. Our main result is to show the existence and uniqueness properties of the departure-time equilibrium for a general class of cost functions that allow for discontinuities and local minima. We present these results in the context of a network with an off-ramp, but they also apply to the standard Vickrey equilibrium [131], since it is a special case of our network. Using these results, we analyze a number of incentive and tolling strategies that can be used by a transportation planning authority to achieve different objectives.

The rest of this chapter is organized as follows. Section 6.2 describes the network and demand models. The main contributions are presented in Section 6.3, which proves the existence and uniqueness properties of the departure time equilibrium. Finally, Section 6.4 presents an analysis of some incentive/tolling strategies.

6.2 Network and demand model

6.2.1 Network

We consider a highway segment with an off-ramp, where the number of vehicles that exit at the off-ramp exceeds its capacity during a peak congestion period. The dynamics of the network are modeled using the point queue model described in Chapter 5. The network contains two types of vehicles; a) vehicles that drive past the off-ramp and stay on the highway that are called highway vehicles (denoted with the subscript h) and b) vehicles that exit the highway at the off-ramp are called exiting vehicles (denoted with the subscript e). The flow of each of these vehicles types is constrained by the capacity limitations of the network, and a queuing delay occurs when the inflow is greater than the bottleneck capacity.

Definition 6.1. ***Flow***
The flow of highway vehicles (and resp. exiting vehicles) entering the network at time is t is $\lambda_h(t)$ (and resp. $\lambda_e(t)$). The flow of highway vehicles (and resp. exiting vehicles) that exit the network at time t is $\lambda_h^{out}(t)$ (and resp. $\lambda_e^{out}(t)$).

Definition 6.2. ***Capacity***
The capacity $\mu_b(t)$ of a bottleneck b is the maximum flow that can enter the link from its input node at time t.

Definition 6.3. ***Delay***
The queuing delay $\delta_q(t)$ at queue q at time t is the waiting time at queue q due to the capacity constraints of the outgoing links from the queue. The total delay $\Delta_g(t)$ for a vehicles of type g entering the network at time t is the total delay that the vehicle experiences across all queues prior to exiting the network[1].

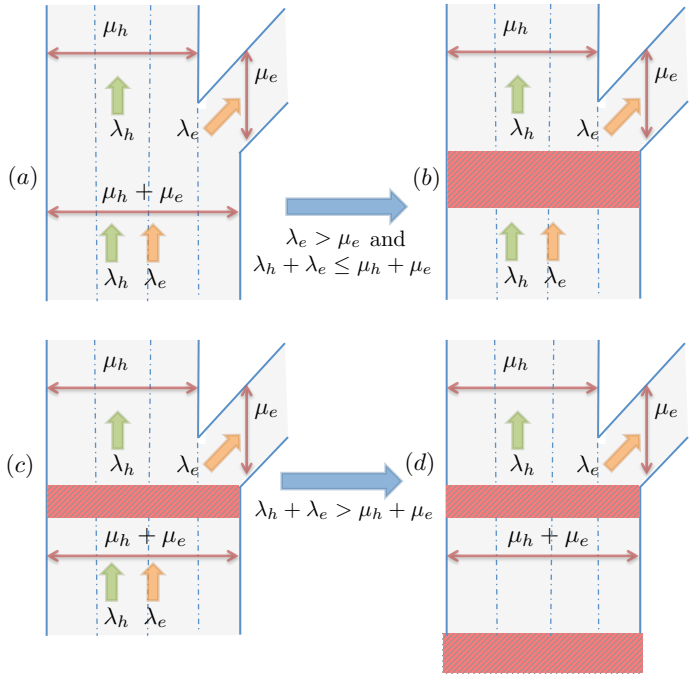


Figure 6.1: Network model. There are two types of queues that can form; i) (a) → (b): a queue forms at the off-ramp entrance if the exiting flow is greater than the off-ramp capacity, ii) (c) → (d) a queue forms at the entrance to the network if the total demand is greater than the capacity freeway capacity at the entrance.

For analyzing the equilibrium departure flows, we need to quantify the delay characteristics of the off-ramp model under different boundary flows. Chapter 5 formally derives the delay characteristics of single source networks that satisfies the following properties, which we will utilize in our analysis.

1. The flow on any link is constrained by the maximum link flow μ[2].
2. The flows through each junction (node) of the network satisfy the first-in-first-out (FIFO) property, i.e. vehicles going to different destinations can not overtake either other.
3. The most constrained exit flow through each junction saturates the corresponding outgoing link, i.e. flow through each junction is maximized subject to the FIFO condition.

We can now analytically express the delays observed by the exiting vehicles as they travel through the network using the framework developed in Chapter 5.

[1]For simplicity of presentation, without loss of generality, we remove the free flow travel-time from our analysis. This can be done because the free flow travel time seen by every vehicle of a given type is the same.

[2]We assume that the capacities are time invariant. Our analysis can be extended to piecewise constant time varying capacities, but we limit this discussion to the time invariant case clarity and conciseness in presenting our contributions.

Proposition 6.1. *Exiting vehicle delay*
If highway flow is restricted to $\lambda_h \leq \mu_h$, i.e. there is no bottleneck purely due to the highway vehicles, the delay seen by the exiting vehicles that enter the network at time t is given by the following differential equation:

$$\left.\frac{d\Delta_e}{dt}\right|_t = \begin{cases} \dfrac{\lambda_e(t)}{\mu_e} - 1 & \text{if there is an active queue} \\ 0 & \text{otherwise, i.e. no queing} \end{cases} \tag{6.1}$$

where queue q being active implies that $\Delta_q(t) > 0$ when the exiting vehicle that enters the network at time t reaches queue q.

Proof. Due to the FIFO condition, the flow through a congested junction is determined by the most constrained outgoing flow. See Chapter 5 for a detailed treatment of the network dynamics. Since there is no capacity drop on the highway at the off-ramp, the most constrained outgoing flow type for the off-ramp queue is always the exiting flow. Furthermore, the queuing caused at the entrance to the network is due to the total flow entering the network. Therefore, following the general derivation for the total delay experienced by vehicles on a particular path from Theorem 5.5 we obtain:

$$\begin{aligned} \left.\frac{d\Delta_e}{dt}\right|_t &= \begin{cases} \dfrac{\lambda_h(t)+\lambda_e(t)}{\mu_e} - 1 + \dfrac{\lambda_e(t)}{\mu_e} - \dfrac{\lambda_h(t)+\lambda_e(t)}{\mu_e} & \text{if the off-ramp queue is the last active queue} \\ \dfrac{\lambda_h(t)+\lambda_e(t)}{\mu_e} - 1 & \text{if the highway queue is the last active queue} \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} \dfrac{\lambda_e(t)}{\mu_e} - 1 & \text{if the off-ramp queue is the last active queue} \\ \dfrac{\lambda_h(t)+\lambda_e(t)}{\mu_e} - 1 & \text{if the highway queue is the last active queue} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.2}$$

□

where the time t is with respect to the time that a vehicle enters the network.

Now let us consider the case where the highway queue is activated. This means that $\lambda_e + \lambda_h > \mu_e + \mu_h$ and since $\lambda_h \leq \mu_h$ implies that $\lambda_e > \mu_e$. This flow exits the highway queue at a rate $\tilde{\lambda}_e = \frac{\lambda_e}{\lambda_e+\lambda_h} \cdot (\mu_e + \mu_h)$ by the FIFO condition. For this flow to no cause an off-ramp

queue, we must have that

$$\begin{aligned}
& & \tilde{\lambda}_e &\leq \mu_e \\
\implies & & \frac{\lambda_e}{\lambda_e + \lambda_h} \cdot (\mu_e + \mu_h) &\leq \mu_e \\
\implies & & \lambda_e \cdot (\mu_e + \mu_h) &\leq \mu_e (\lambda_e + \lambda_h) \\
\implies & & \lambda_e \cdot \mu_h &\leq \mu_e \cdot \lambda_h \\
\implies & & \lambda_h &\geq \frac{\lambda_e}{\mu_e} \cdot \mu_h
\end{aligned}$$

However, since the existence of a highway queue implies that $\lambda_e > \mu_e$ and we know that $\lambda_h \leq \mu_h$, this is not possible. Therefore, if a highway queue forms the flow exiting the highway queue will form a off-ramp queue. This implies that the off-ramp queue is the last active queue. Therefore we have,

$$\left.\frac{d\Delta_e}{dt}\right|_t = \begin{cases} \dfrac{\lambda_e(t)}{\mu_e} - 1 & \text{if there is an active queue} \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

6.2.2 Demand model

Assumption 6.1. ***Exiting vehicle demand***
We assume that the exiting vehicles are free to choose their departure times and do so in a selfish manner to minimize a cost function C. Therefore, the demand for the exiting vehicles $\lambda_e(t)$ will form a Nash equilibrium (or user equilibrium) with respect to the cost function C.

Assumption 6.2. ***highway vehicle demand***
We assume that the highway vehicle demand $\lambda_h(t)$ such that $\lambda_h(t) \leq \mu_h(t)$ is fixed (exogenous) and not a function of the exiting vehicle demand distribution.

The cost function C consists of a cost related to the queuing delay on the network and a cost related to the arrival time at the destination.

Definition 6.4. ***Delay cost function***
The delay cost function C_δ assigns a cost $C_\delta(\Delta(t))$ corresponding to a queuing delay of $\Delta(t)$. The delay cost encountered by an exiting vehicle that enters the network at time t is given by $C_\delta(\Delta_e(t))$.

Definition 6.5. ***Schedule time cost function***
The exiting vehicles have an expected arrival time at the destination and the schedule time cost function C_S assigns a penalty $C_S(t_a)$ corresponding to the actual arrival time t_a. The schedule time cost encountered by the an exiting vehicle that enter the network at time t is given by $C_S(t + \Delta_e(t))$.

As our goal is to analyze the impact of incentive and tolling strategies on the departure time equilibrium of the exiting vehicles and the resulting impact on overall congestion, we also define an incentive/toll cost function with a toll being modeled as a negative incentive.

Definition 6.6. ***Incentive/toll cost***
The incentive/toll cost function C_I assigns a cost $C_I(t_a)$ corresponding to the incentive/toll for arriving at the destination at time t_a.
If $C_I \leq 0$, $|C_I(t_a)|$ represents the incentive or negative toll given to the vehicles that exit the network at time t_a.
If $C_I > 0$, $C_I(t_a)$ represents the toll or negative incentive charged to the vehicles that exit the network at time t_a.

Definition 6.7. ***Arrival cost***
The arrival cost C_A is the total cost experienced by a vehicle due to its arrival time.

$$C_A(t_a) = C_S(t_a) + C_I(t_a) \tag{6.4}$$

The delay cost is a function of the queuing delay, while both the schedule time and incentive/toll costs are functions of the arrival time. The total cost can now be defined as follows.

Definition 6.8. ***Total cost*** C
The total cost $C(t)$ is the sum of the delay cost and the arrival cost for a vehicles that enters the network at time t.

$$C(t) = C_\delta(\Delta_e(t)) + C_A(t + \Delta_e(t)) \tag{6.5}$$

We will now model the behavior of the existing vehicles with respect to the network model and the cost functions.

Definition 6.9. ***Exiting vehicle equilibrium***
Given a network with an exit and a fixed number of exiting vehicles N, cost functions (C_δ, C_A, C_I) and highway vehicle demand $\lambda_h(t)$, λ_e is an exiting vehicle equilibrium if and only if

$$\begin{cases} \lambda_e(t) & \geq & 0 & \text{is piecewise continuous} \\ \int_{\mathbb{R}} \lambda_e(\tau)\, d\tau & = & N & \\ \lambda_e(t) > 0 & \Rightarrow & C(t) \leq C(t'),\ \forall t' & \end{cases} \tag{6.6}$$

where $C(t)$ is the total cost and $\Delta_e(t)$ is the total delay in the network given $\lambda_e(\cdot)$ and $\lambda_h(\cdot)$. The equilibrium cost for each vehicle is denoted by C_E.

6.3 Existence and uniqueness of the exiting vehicle equilibrium

In this section, we will prove the existence and uniqueness of the exiting vehicle equilibrium for a general class of cost functions within the dynamics of our network model. We first introduce the general class of cost functions that we consider.

6.3.1 Equilibrium compatible cost functions

The classical single route single bottleneck equilibrium departure time problem was first introduced by Vickrey [131] in 1969. Smith [127] proved the existence of an equilibrium for convex arrival cost functions, and Daganzo [32] proved the uniqueness of this solution. Convex cost functions imply that the marginal cost of earliness (or lateness) increases as commuters arrive earlier (or later), which is a reasonable assumption. However, convex cost functions by themselves are not adequate in our setting. To design time dependent incentives and tolls, we require the ability to use schedule cost functions $C_A = C_S + C_I$. This introduces a more complex set of cost functions we must be able to accommodate. The following generalizations are required:

- *Local maximums:* An incentive/toll is intended for pushing commuters out of the peak congestion period. Thus, it is reasonable to envision incentives/tolls that are proportional to the peak congestion pattern and therefore inversely proportional to the schedule cost function C_S. This could result in an arrival time function C_A that admits local maximums, which we must be able to support.

- *Discontinuity:* A fixed incentive/toll to encourage commuters to arrive before some time t_I could take the form

$$C_I(t) = \begin{cases} I < 0 & \text{if } t < t_I \\ 0 & \text{if } t \geq t_I \end{cases} \tag{6.7}$$

  Thus a discontinuity $C_A(t_I^+) - C_A(t_I^-) = I$ will appear in the arrival cost function $C_A(t)$ at $t = t_I$, where I is the value of the incentive/toll.

Definition 6.10. ***Equilibrium compatible cost functions***
The functions (C_A, C_δ) are equilibrium compatible cost functions if they satisfy the following requirements.

1. *C_δ is convex on $\mathbb{R}^+$, C^1 and admits a unique minimum at 0.*
2. *C_A is C^1 on the right and piecewise C^1, with a finite number of positive discontinuities such that $C_A(t^+) \geq C_E$ and no negative discontinuities in the support of the solution.*
3. *C_A' has a finite number of sign changes $\Leftrightarrow$ C_A has a finite number of local maximums.*
4. $\lim_{t \to \pm\infty} C_A(t) = +\infty$
5. $\exists t_0 : -\frac{dC_A(t)}{dt} < \frac{dC_\delta(0)}{d\delta},\ \forall t > t_0$

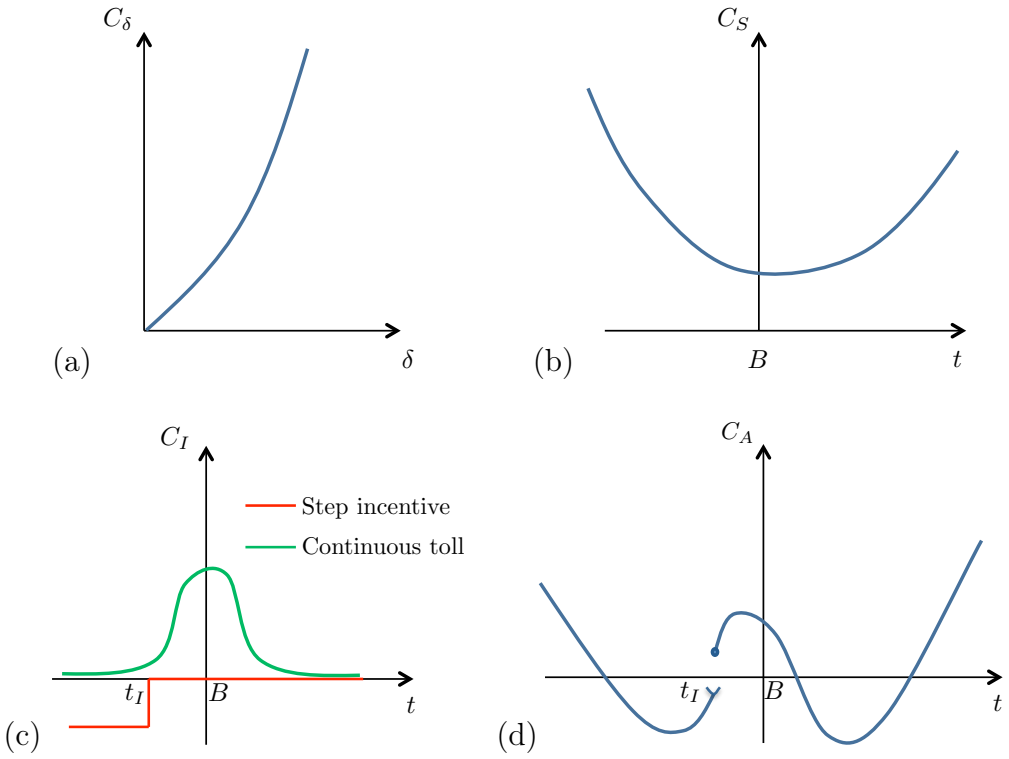


Figure 6.2: Illustration of equilibrium compatible cost functions - (a) Equilibrium compatible delay cost functions C_δ are convex on $\mathbb{R}^+$, C^1 and admit a unique minimum at 0. (b) A classical convex schedule cost function C_S (c) A continuous toll that induces a local minimum and a step incentive/toll that induces a discontinuity in the arrival cost function d) The resulting arrival cost function C_A.

The first condition ensures that C_δ penalizes queuing delay and that the marginal cost of delay is monotonically increasing. The second condition allows for a finite number of positive discontinuities in C_A due to step incentives or tolls. The restriction on a finite number discontinuities is not a practical limitation, since there will be finite number of incentives/tolls implemented in practice. The third and the fourth conditions replace the convexity assumption of the arrival cost function with something more general that allows for a finite number of local maximums in C_A, as long as very large early (or late) arrivals still result in large penalties. The last assumption ensures that marginal cost of delay is greater than the marginal cost of the arrival time. Figure 6.2 illustrates the different cost functions that our extended framework can accommodate.

Proposition 6.2. ***Equilibrium compatible cost functions allow local maximums and discontinuities in the arrival time function.***

Proof. The proposition follows directly from definition 6.10 by construction. □

6.3.2 Fixed cost equilibrium

Solving for an exiting vehicle equilibrium directly is difficult due to the flow conservation constraint $\int_{\mathbb{R}} \lambda_e(\tau)\, d\tau = N$. Therefore, we will first consider the simpler problem of finding the equilibrium for a fixed cost C_E, where the total number of exiting vehicles $\int_{\mathbb{R}} \lambda_e(\tau)\, d\tau > 0$ is not fixed and is a function of the cost C_E.

Definition 6.11. ***Fixed cost equilibrium***
The fixed cost equilibrium $E(C_E)$ for a given cost C_E is given by $\lambda_e(t)$ that satisfies the following equations.

$$\begin{cases} \lambda_e(t) \geq 0 & \text{is piecewise continuous} \\ \lambda_e(t) > 0 \;\Rightarrow\; C(t) = C_E \\ \lambda_e(t) = 0 \;\Rightarrow\; C(t) \geq C_E \end{cases} \tag{6.8}$$

Proposition 6.3. ***Exiting vehicle equilibrium for a fixed cost***
If $\lambda_e(t)$ is the solution to the fixed cost equilibrium $E(C_E)$, then $\lambda_e(t)$ is also an exiting vehicle equilibrium for $N = \int \lambda_e(\tau)\, d\tau$ exiting vehicles.

Proof. The solution satisfies the requirements of definition 6.9. The first requirement follows directly from definition 6.11. The second requirement is true by construction. The third requirement enforces that $C(t) \leq C(t')$ for all t' if t is in the support of $\lambda_e(t)$. From definition 6.11 we know that $C(t) = C_E$ in the support of $\lambda_e(t)$ and $C(t) \geq C_E$ outside. Therefore, the third requirement is also satisfied. □

In this section, we will consider the existence and uniqueness of the solution to the fixed cost equilibrium under equilibrium compatible cost functions. We first present some definitions that will be used in the analysis.

Definition 6.12. ***Plateau***
A plateau P is an interval $[t_a, t_b)$ such that $C_A(t) = C_E, \forall t \in P$ and $|P| > 0$. The arrival time cost for all vehicles that arrive at the destination during this interval is C_E.

Definition 6.13. ***Valley***
A valley V is an interval $[t_a, t_b)$ such that $C_A(t_a) = C_E$ and $C_A(t) < C_E,\ \forall t \in (t_a, t_b)$. The arrival time cost for a vehicle that arrives at the destination at time t_a is C_E and the cost is strictly less than C_E for all $t \in (t_a, t_b)$.

Figure 6.3 gives a graphical illustration of how an arrival time function is split into valleys and plateaus.

Definition 6.14. ***Dominant plateaus and valleys***
A plateau or valley $[t_a, t_b)$ is dominant if $\Delta(t_a) = 0$. A vehicle that arrives at the beginning of a dominant plateau or valley has zero queuing delay. A plateau or valley that is not dominant is called dominated plateau or valley.

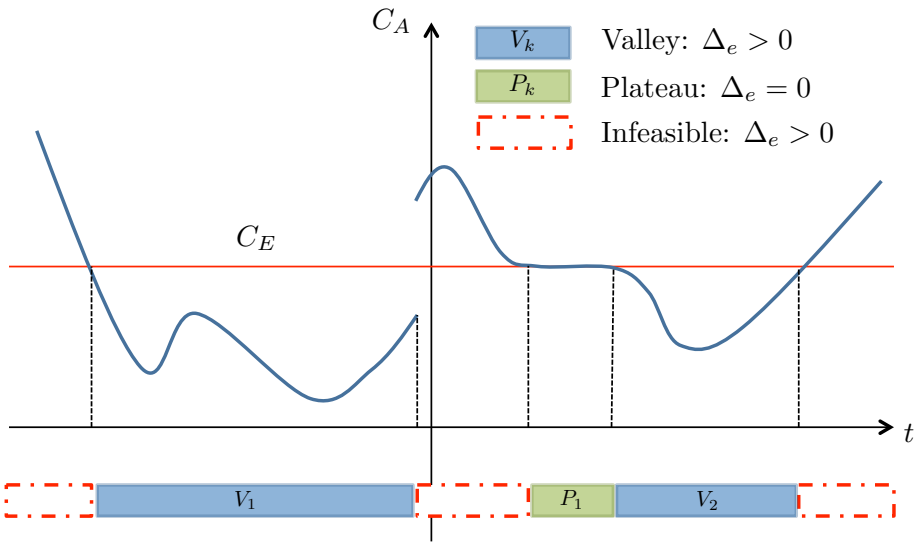


Figure 6.3: Set of windows and plateaus

Proposition 6.4. ***All valleys and plateaus are dominant***

Proof. The first valley or plateau is dominant because no flow can enter the network when $C_A > C_E$. Furthermore, each valley $V = [t_a, t_b)$ ends with either $C_A(t_b) = C_E$ or a positive discontinuity such that $C_A(t_b) > C_E$. In either case, the queue must be empty at $t = t_b$ because $C_A(t_b) \geq C_E$ and any queuing delay will result in a total cost greater than C_E, which violates the equilibrium. Also, since $C_A(t) = C_E$ during any plateau, there can not be any queuing delay during the plateau for the same reason. Therefore, since the first valley or plateau is dominant and all valleys and plateaus end with no queue, all valleys and plateaus are dominant. □

Proposition 6.5. ***Window of feasible arrival times for a fixed cost equilibrium***
The window of feasible arrival times F (i.e. times at which the exiting vehicles leave the network) for a fixed cost equilibrium with cost C_E is the union of a finite number of plateaus and valleys.

$$F = \{\cup_{i=1}^{n_P} P_i\} \cup \{\cup_{j=1}^{n_V} V_j\} \tag{6.9}$$

Proof. The arrival time cost $C_A(t)$ must be less than or equal to C_E for all feasible arrival times with an equilibrium cost of C_E. Furthermore, $\lim_{t \to \pm\infty} C_A(t) = +\infty$ from definition 6.10, which implies that there is some t_{min} such that $C_A(t_{min}) > C_E$ and t_{max} such that $C_A(t_{max}) > C_E$, and the feasible arrival times are bounded by (t_{min}, t_{max}). Also from definition 6.10, we know that C_A is piecewise C^1 with a finite number of discontinuities and that C'_A has a finite number of sign changes. Therefore, there can only be a finite number of plateaus and valleys, since plateaus and valleys begin and end with either $C_A(t) = C_E$ or with a discontinuity in $C_A(t)$. □

Proposition 6.6. ***Vehicles only enter the network inside the window of feasible arrival times***
If $\lambda_e(t)$ is the solution to the fixed cost equilibrium $E(C_E)$), the support of $\lambda_e(t)$ (i.e. times at which the exiting vehicles enter the network) is limited to the window of feasible arrival times, i.e. $t \notin F \Rightarrow \lambda_e(t) = 0$

Proof. Let t_0 be the time at which the first exiting vehicle enters the network. The queuing delay for this vehicle $\Delta_e(t_0) = 0$, since there are no other exiting vehicles already in the network. If $t_0 \notin F$, the first exiting vehicle will reach the destination at time $t_0 + \Delta_e(t_0) = t_0$ and be subject to an arrival time cost of $C_A(t_0)$. However, since $C_A(t_0) > C_E,\ \forall t_0 \notin F$ this violates the fixed cost equilibrium and therefore t_0 must be in F. Furthermore, this means the no exiting vehicle can enter the network at any time $t \notin F$ if $\Delta_e(t) = 0$, which means that the exiting vehicles must enter during a plateau or valley. Since F is the set of plateaus and valleys this concludes the proof. □

Lemma 6.1. ***Existence and uniqueness of the solution of fixed cost equilibrium on a valley***
Let $J = [t_a, t_b)$ be a valley. Given the boundary condition $\Delta_e(t_a)$ on the left of the valley J, equations (6.8) *have a unique continuous solution on a dominant valley. A solution exists but is not unique for a dominated valley.*

Proof. From definition 6.8 we have:

$$C(t) = C_\delta(\Delta_e(t)) + C_A(t + \Delta_e(t)) \tag{6.10}$$

$$\Rightarrow \frac{dC(t)}{dt} = \left.\frac{dC_\delta}{dt}\right|_{\Delta_e(t)} \cdot \frac{d(\Delta_e(t))}{dt} + \left.\frac{dC_A}{dt}\right|_{t+\Delta_e(t)} \cdot \left[1 + \frac{d(\Delta_e(t))}{dt}\right] \tag{6.11}$$

From definition 6.11 for a fixed cost equilibrium, we know that $C(t)$ is constant for all t such that $\lambda_e(t) > 0$.

If $\frac{dC(t)}{dt} = 0$ and $C(t) = C_E$,

$$\left.\frac{dC_\delta}{dt}\right|_{\Delta_e(t)} \cdot \frac{d\Delta_e(t))}{dt} + \left.\frac{dC_A}{dt}\right|_{t+\Delta_e(t)} \cdot \left[1 + \frac{d\Delta_e(t))}{dt}\right] = 0 \tag{6.12}$$

$$\frac{d\Delta_e(t))}{dt}\left[\left.\frac{dC_\delta}{dt}\right|_{\Delta_e(t)} + \left.\frac{dC_A}{dt}\right|_{t+\Delta_e(t)}\right] = -\left.\frac{dC_A(t)}{dt}\right|_{t+\Delta_e(t)} \tag{6.13}$$

$$\frac{d\Delta_e(t)}{dt} = \frac{-\left.\frac{dC_A}{dt}\right|_{t+\Delta_e(t)}}{\left.\frac{dC_\delta}{dt}\right|_{\Delta_e(t)} + \left.\frac{dC_A}{dt}\right|_{t+\Delta_e(t)}} \tag{6.14}$$

Let t_a^0 be the time at which the vehicle that reaches the exit queue (E) at time t_a had left the origin at. i.e. $t_a^0 + \Delta_e(t_a^0) = t_a$. By the definition of a dominant valley, we know that $\Delta_e(t_a^0) = 0$, $t_a = t_a^0$ and $C_A(t_a) = C_E$. We consider the initial value problem specified by equation (6.14) and $\Delta_e(t_a^0) = 0$ for the interval $J = [t_a, t_b)$.

Assume that $\lambda_h(t)$ and $\lambda_e(t)$ are continuous functions for all t such that $t + \Delta_e(t) \in J$. Under this assumption, the function $\frac{d\Delta_e(t)}{dt}$ is continuous in $t, \forall t \in J$, since $\Delta_e(t)$ is continuous in t from equation (6.2) and C_δ, C_A are C^1 in J by definition 6.10. The function $\frac{d\Delta_e(t)}{dt}$ is also Lipschitz continuous in $\Delta_e(t), \forall t \in J$ because every continuously differentiable function is locally Lipschitz and C_δ and C_A are C^1 in J. Therefore, by the Picard-Lindelöf Theorem $\Delta_e(t)$ admits a unique solution on J.

From Proposition 6.1, the delay evolution for exiting vehicles is given by,

$$\left.\frac{d\Delta_e}{dt}\right|_t = \begin{cases} \dfrac{\lambda_e(t)}{\mu_e} - 1 & \text{if there is an active queue} \\ 0 & \text{otherwise, i.e. no queing} \end{cases} \tag{6.15}$$

Let us now consider the solution $\lambda_e(t)$ to equations (6.8). If there is an active queue at the exit, the unique exiting vehicle flow is given directly by equation (6.15),

$$\lambda_e(t) = \left(\left.\frac{d\Delta_e}{dt}\right|_t + 1 \right) \cdot \mu_e \tag{6.16}$$

The exiting vehicle queue is active in J because $C_A(t) < C_E \in (t_a, t_b)$ and the equilibrium would be violated if there was no queuing delay (i.e. if $\Delta_e(t) = 0$). Also, note that the solution is continuous since $\Delta_e(t)$ is C^1. Therefore, the solution to $\lambda_e(t)$ given by equation (6.16) is both continuous and unique $\forall t \in J$.

What remains to be shown is that the transitions between the queuing states is also unique. Since the flows within any queuing state are unique and the state transitions are only depend on the previous flows, the state transitions are also unique.

Finally, we show that the unique fixed cost equilibrium $\lambda(t)$ is also physically acceptable, i.e. $\lambda_e(t) \geq 0$. From Proposition 6.1, we know that the off-ramp queue is always the last active queue. Therefore, $\lambda_e \geq 0$ requires $\lambda_e = \left(1 + \left.\frac{d\Delta_e}{dt}\right|_t\right) \cdot \mu_e \geq 0 \implies \left.\frac{d\Delta_e}{dt}\right|_t \geq -1$. This follows directly from equation (6.14) because from definition 6.10 we know that $-\frac{dC_A(t)}{dt} < \frac{dC_\delta(0)}{dt}$.

□

Corollary 1. ***Solution to the fixed cost equilibrium on a valley***
The solution to the fixed cost equilibrium C_E for a valley $V = [t_a, t_b)$ can be found by solving the ordinary differential equation (6.14) *with the initial condition $\Delta_e(t_a) = 0$ and plugging it into equation* (6.15).

Proposition 6.7. ***Feasible flow in plateaus***
On a plateau $P = [t_a, t_b)$, any flow $\lambda_e(t) \in [0, \mu_e]$ is a feasible flow.

Proof. Since all plateaus are dominant, $\Delta_e(t_a) = 0$ and $C_A(t) = C_E, \forall t \in P$. Since the condition $C(t) = C_E$ must hold for an equilibrium solution and $C_A(t) = C_E$, $\Delta_e(t)$ must be zero for all $t \in P$. Therefore, $0 \leq \lambda_e(t) \leq \mu_e, \forall t \in P$. □

Proposition 6.8. ***Unique flow in a valley***
Let $N(V)$ be the unique number of vehicles that pass the exit during a valley $V = [t_a, t_b)$.

$$N(V) = \mu_e |V| \tag{6.17}$$

Proof. Since $C_A(t_a) = C_E$ and $C_A(t') \leq C_E, \forall t' \in [t_a, t_b)$, all the vehicles exiting the valley in the interval $[t_a, t_b)$ must have all seen some queuing delay and the exit queue is active during the entire interval. Therefore, the flow of exiting vehicles is equal to the bottleneck flow of μ_e during the entire interval and the result follows directly. □

Theorem 6.1. ***Existence and uniqueness of fixed cost equilibrium***
If λ_h is piece-wise continuous, C_A and C_δ are equilibrium compatible cost functions, the fixed cost equilibrium $E(C_E)$ exists and the solution is unique if C_A does not contain any plateaus.

Proof. If C_A does not contain any plateaus or valleys and the window of feasible arrival times $F = \emptyset$, then the solution to the fixed cost equilibrium $E(C_E)$ is $\lambda_e(t) = 0, \forall t$, since vehicles can only enter the network within the F from Proposition 6.6. If $F \neq \emptyset$, then F is the union of a disjoint set of plateaus and valleys, and we know that the flow $\lambda_e(t) = 0, \forall t \notin F$. From Lemma 6.1 we know that a solution to the equilibrium $E(C_E)$ exists and is unique for each valley. Also, we know from Proposition 6.7 that a solution to the equilibrium exists, but is not unique for plateaus.

Therefore, a fixed cost equilibrium exists when λ_h is piece-wise continuous, C_A and C_δ are equilibrium compatible cost functions. The solution is unique if C_A does not contain any plateaus of $C_A(t) = C_E$. □

6.3.3 Existence and uniqueness of exiting vehicle equilibrium

Lemma 6.2. ***Number of exiting vehicles as a function of equilibrium cost***
If C_A does not contain any plateaus for $C_E \in (C_{min}, C_{max})$, then $C_E \mapsto \Phi(C_E)$ is a continuous function for $C_E \in (C_{min}, C_{max})$. If C_A does contain a plateau for $C_E \in (C_{min}, C_{max})$, then $C_E \mapsto \Phi(C_E)$ is a set valued map for $C_E \in (C_{min}, C_{max})$. This property is illustrated in figure 6.4.

Proof. From Theorem 6.1 we know that an equilibrium solution exists for any fixed cost C_E. There is no non-zero equilibrium with cost $C < \min(C_A)$, since $C_\delta \geq 0$. For $C = \min(C_A)$, there can only be a non-zero equilibrium solution if there exists a plateau at $C = \min(C_A)$.

For $C > \min(C_A)$ a plateau must exist by definition. As C_A is piecewise C^1, the boundaries of each valley grow as a continuous function of C_E. Also, from Proposition 6.8 we know that the number of vehicles that exit the network during some valley V, $N(V)$

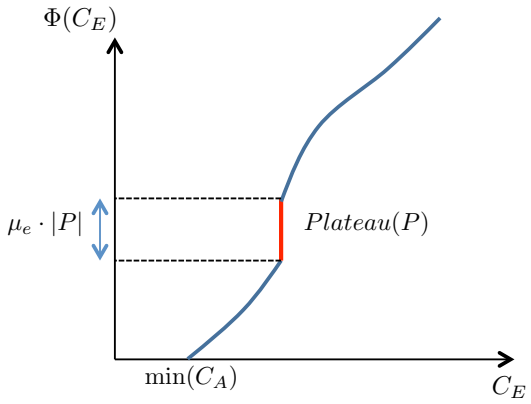


Figure 6.4: Number of employees which can arrive as a continuous correspondence of fixed cost C_E

is equal to $\mu_e|V|$. Therefore, $N(V)$ is a continuous function of C_E. In the case where two valleys V_1, V_2 merge, the sum $|V_1| + |V_2|$ grows continuously.

Let $V(C_E)$ be the set of valleys at $C_A = C_E$. If there are no plateaus in C_A, $\Phi(C_E) = \sum_{W \in V(C_E)} N(W)$ is also a continuous function of C_E.

If C_A includes a plateau at $C_A = C_E$, the number of vehicles that can exit during the plateau can be any value in the range $(0, \mu_e)$. Therefore, the number of vehicles that exit the network as a function of C_E is then a set valued map. □

Theorem 6.2. ***Existence and uniqueness of exiting vehicle equilibrium***
For any total demand of exiting vehicles N, an exiting vehicle equilibrium $\lambda_e(t)$ that satisfies definition 6.10 exists. The equilibrium is unique if there are no plateaus at the equilibrium cost C_E.

Proof. From Lemma 6.2 we know that the number of exiting vehicles N is a continuous function of the equilibrium cost C_E. Therefore, for any N there is a corresponding equilibrium cost $C_E = \Phi^{-1}(N)$ and the resulting fixed cost equilibrium flow distribution is an exiting vehicle equilibrium by Proposition 6.3. □

Corollary 2. ***Solution to the exiting vehicle equilibrium***
The solution to the exiting vehicle equilibrium with N vehicles can be found as follows.

1. *Find the equilibrium cost C_E, which is the minimum cost C such that the length of the support of arrival cost function $\{C_A : C_A \leq C_E\}$ is greater than or equal to $\frac{N}{\mu_e}$, i.e. $C_E = \min\{C : |C(t) \leq C| \cdot \mu_e \geq N\}$. The condition holds with equality if there are no plateaus at $C = \Phi^{-1}(N)$.*

2. *For each valley in $V \in V(C_E)$, solve equation (6.14) with the initial condition $\Delta_e(t_a) = 0$ and plug it into equation (6.15) to obtain the solution on V.*

The equilibrium departure flows are solved in practice (approximately) by numerically integrating equation (6.14). Algorithm 6.1 shows how this numerical integration can be done for each valley V.

Algorithm 6.1 Calculate λ_e

Require: $\{t_a, t_b\}\ \forall V \in V(C_E)$ and unit time discretization[3]
optFlow$(\mathbf{t_a}, \mathbf{t_b})$
  for $V \in V(C_E)$ **do**
    $\Delta_e[t_a(V)] = 0$
    for $t = t_a(V)$ to $t_b(V)$ **do**

$$d\Delta_e[t] = \frac{-\left.\frac{dC_A(t)}{dt}\right|_{t+\Delta(t)}}{\left.\frac{dC_\delta(t)}{dt}\right|_{\Delta(t)} + \left.\frac{dC_A(t)}{dt}\right|_{t+\Delta(t)}} \cdot \Delta t$$

$$\Delta_e[t+1] = \Delta_e[t] + d\Delta_e[t]$$

$$\lambda_e[t] = \mu_e \cdot \left(1 + \frac{d\Delta_e[t]}{\Delta t}\right)$$

    end for
  end for
  return λ_e

6.4 Analysis of incentive/tolling functions

We will now analyze different incentive/tolling functions that reduce the highway congestion caused by the bottleneck at the off-ramp with respect to congestion reduction, cost efficiency and robustness of the solution.

Definition 6.15. ***highway vehicle cost***
The highway vehicle cost σ_h is the cost imposed on the highway vehicles due to the congestion caused by the exiting vehicles.

$$\sigma_h = \int \lambda_h^0(\tau) \cdot C_\delta\left(\Delta_h(\tau)\right)\ d\tau \tag{6.18}$$

Definition 6.16. ***Exiting vehicle cost***
The exiting vehicle cost σ_e is the total cost for the exiting vehicles due to the queuing at the

[4]The problem can be normalized to achieve a unit time discretization without any loss of generality.

bottleneck and the arrival time cost.

$$\begin{aligned} \sigma_e &= \int \lambda_e^0(\tau) \cdot \left[C_\delta\left(\Delta_h\left(\tau\right)\right) + C_A\left(t + \Delta_e\left(\tau\right)\right)\right] \, d\tau && (6.19) \\ &= \int \lambda_e^0(\tau) \cdot C_E \, d\tau && (6.20) \\ &= N \cdot C_E && (6.21) \end{aligned}$$

where N is the total number of exiting vehicles and C_E is the equilibrium cost. The exiting vehicle cost only depends on the equilibrium cost.

Definition 6.17. ***Cost of incentives/tolls***
The cost of incentives/tolls is the total amount of incentives and tolls distributed to the exiting vehicles.

$$\sigma_I = -\int \lambda_e^0(\tau) \cdot C_I(t + \Delta_e(\tau)) \, d\tau \tag{6.22}$$

A incentive/tolling function that results in $\sigma_I = 0$ is called a revenue neutral incentive/tolling function.

Definition 6.18. ***Social cost***
The social cost σ is the total cost to the system due to both highway inefficiency and the cost of incentives/tolls.

$$\sigma = \sigma_h + \sigma_I + \sigma_e \tag{6.23}$$

6.4.1 Zero-congestion incentives/tolls

Definition 6.19. ***highway optimal incentive/toll***
The highway optimal incentive/toll is the incentive/toll required to eliminate congestion on the highway due to the exiting vehicles during the exiting vehicle equilibrium.

Proposition 6.9. ***Computing the highway optimal incentive***
Let C_E be the equilibrium cost without any incentives for N exiting vehicles and $\text{supp}(\lambda)$ *be the support of the equilibrium flow of exiting vehicles. The highway optimal incentive is:*

$$C_I(t) = \begin{cases} \min(C_S) - C_S & \text{if } t \in \text{supp}(\lambda) \\ 0 & \text{if } t \notin \text{supp}(\lambda) \end{cases} \tag{6.24}$$

The new equilibrium cost will be $\min(C_S)$.

Proof. The arrival cost function C_A given the highway optimal incentive C_I is

$$\begin{aligned} C_A(t) &= \min(C_S) \quad \text{if } t \in \text{supp}(\lambda) && (6.25) \\ C_A(t) &\geq \min(C_S) \quad \text{if } t \notin \text{supp}(\lambda) && (6.26) \end{aligned}$$

since $C_A = C_S + C_I$. Therefore, $\{t : C_A(t) = min(C_A)\} = \text{supp}(\lambda)$ and there is a plateau of length $\text{supp}(\lambda)$ at the minimum arrival time cost. A plateau of length equal to the support of the original equilibrium flow distribution allows for new equilibrium departure distribution that does not cause any congestion, since $C_A = C_E$ for all $t \in \text{supp}(\lambda)$. Figure 6.5 illustrates the highway optimal incentive for a simple schedule cost function. □

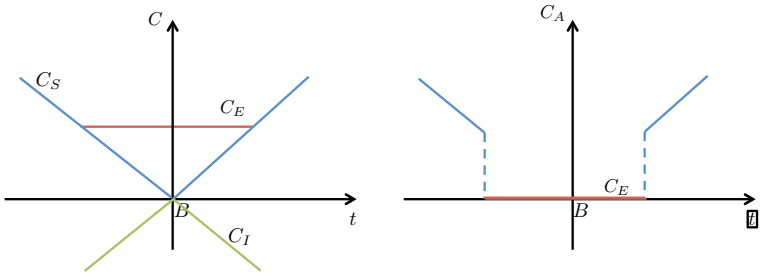


Figure 6.5: The highway optimal incentive for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and corresponding highway optimal incentive C_I. **Right:** the corresponding arrival cost function C_A with the new equilibrium C_E that leads to no queuing.

For each feasible incentive C_I, there is a corresponding toll C_I^T such that both the incentive and tolling function lead to the same equilibrium flow distribution.

Definition 6.20. ***Complementary toll***
Given a bounded incentive C_I, the complementary toll for this incentive C_I^T is

$$\begin{aligned} C_I^T(t) &= -min\left(C_I\left(t\right)\right) + C_I(t) && (6.27) \\ &\geq 0 && (6.28) \end{aligned}$$

Proposition 6.10. ***Incentives and tolls***
For a fixed schedule cost function C_S and delay function C_δ, both the incentive function C_I and the tolling function C_I^T lead to the same equilibrium.

Proof. The exiting vehicle equilibrium only depends on the shape of the arrival cost function $C_A(t)$, i.e. the relative cost, so adding a constant $-min(C_I)$ will not alter the equilibrium. □

Corollary 3. *The highway optimal toll is:*

$$C_I(t) = \begin{cases} C_E - C_S & \text{if } t \in \text{supp}(\lambda) \\ 0 & \text{if } t \notin \text{supp}(\lambda) \end{cases} \quad (6.29)$$

The new equilibrium cost will be C_E. Figure 6.6 illustrates the highway optimal toll for a simple schedule cost function.

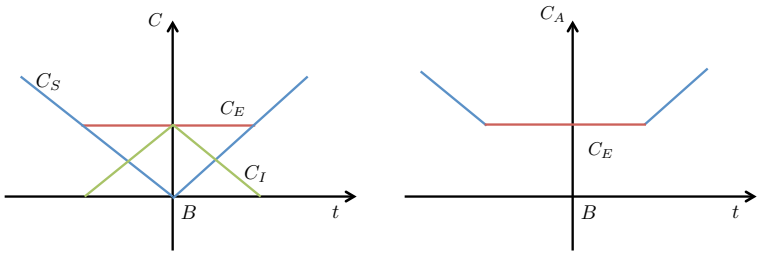


Figure 6.6: The highway optimal toll for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and corresponding highway optimal toll C_I. **Right:** the corresponding arrival cost function C_A with the new equilibrium C_E that leads to no queuing.

Note that the equilibrium cost for the exiting vehicles and who bares the cost of moving the equilibrium is different in the two cases. In the case of an incentive, the controlling agency will bear the entire cost of the demand shift, while in the case of a toll, the exiting vehicles will bear the entire cost of the demand shift.

Corollary 4. *Any equilibrium that is achieved via a incentive or toll can also be achieved via a combination of incentives and tolls. Figure 6.7 illustrates a incentive/toll combination that achieves an highway optimal flow allocation for a simple schedule cost function.*

This allows the controlling agency to distribute the cost of the demand shift in an equitable manner. For example, the distribution can be such that the tolls charged to the exiting vehicles is equal to the incentive, which means that the controlling agency has no net gains or losses (i.e. the control strategy is revenue neutral). Therefore, it is possible to reduce both σ_h and σ_e while the net incentive/tolling cost σ_I is zero, which reduces the total social cost σ.

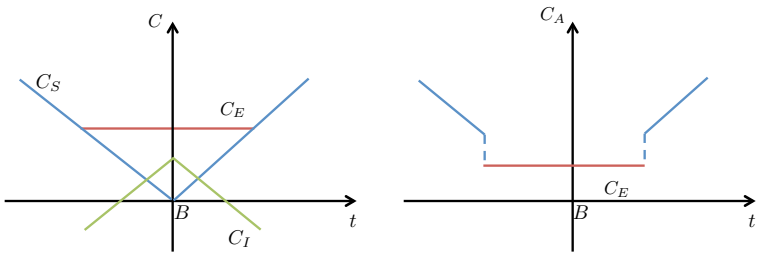


Figure 6.7: A combined incentive and tolling strategy that achieves a highway optimal flow allocation for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and corresponding highway optimal incentive/toll C_I. **Right:** the corresponding arrival cost function C_A with the new equilibrium C_E that leads to no queuing. The vehicles that arrive within t_I of the scheduled arrival time B are tolled, while the vehicles that arrive outside this window are given an incentive.

Proposition 6.11. ***Shifting the equilibrium***
The support of the equilibrium can be shifted by t_s with the combination of tolls and incentives $C_I(t) = C_S(t + t_s) - C_S(t)$ to achieve a new equilibrium with the same equilibrium cost.

Proof. To shift the equilibrium by some value t_s, we need to modify C_A such that $C_A(t) = C_S(t + t_s)$. By definition $C_A(t) = C_S(t) + C_I(t)$. Therefore, we require

$$C_S(t + t_s) = C_S(t) + C_I(t) \tag{6.30}$$
$$\Rightarrow \; C_I(t) = C_S(t + t_s) - C_S(t) \tag{6.31}$$

Figure 6.8 illustrates a left shift of the equilibrium by t_s. □

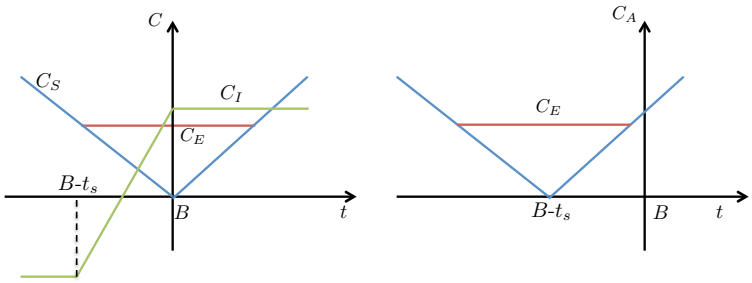


Figure 6.8: A combined incentive and tolling strategy that shifts the equilibrium left by t_s for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and incentive/toll C_I corresponding to a left shift of the equilibrium by t_s. **Right:** the corresponding arrival cost function C_A with the new flow distribution and unchanged equilibrium cost C_E.

Corollary 5. *A shifted highway optimal equilibrium can be achieved by combining a shift incentive/toll with a highway optimal incentive/toll. Figure 6.9 illustrates a highway optimal incentive with a left shift of the equilibrium by t_s.*

An highway optimal incentive/toll eliminates highway congestion. Furthermore, the building blocks described above provide great flexibility in both shifting the equilibrium flows across time and distributing the cost of the demand shift between the vehicles and the controlling agency. However, there are some a couple of drawbacks to this approach.

- A continuous time varying incentive/toll is extremely difficult to implement in practice.
- The precise arrival and delay cost function are not known.

Therefore, it is unlikely that such an incentive/toll will be used in practice. However, these optimal strategies serve as a useful reference for implementing the simpler piecewise constant incentives/tolls that are most commonly used in practice.

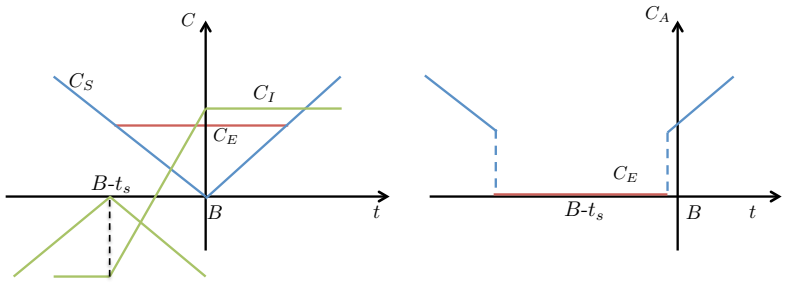


Figure 6.9: A combined incentive and tolling strategy that achieves a highway optimal flow and shifts the equilibrium left by t_s for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and two incentive/toll functions C_I^1, C_I^2 that correspond respectively to a left shift of the equilibrium by t_s and an highway optimal flow. **Right:** the corresponding arrival cost function C_A with the new flow distribution with equilibrium cost C_E.

6.4.2 Step incentives/tolls

Definition 6.21. ***Step incentive/toll***
A step incentive/toll is an incentive/tolling function with a constant value up to a given time t_I and zero after that.

$$C_I(t) = \begin{cases} I < 0 & \text{if } t < t_I \\ 0 & \text{if } t \geq t_I \end{cases} \tag{6.32}$$

Thus, since the schedule cost function C_S is continuous, a step incentive/toll will impose a discontinuity $C_A(t_I^+) - C_A(t_I^-) = I$ in the arrival cost function $C_A(t)$ at $t = t_I$, where I is the value of the incentive/toll.

From the definition of equilibrium compatible cost functions (definition 6.10), we know that the arrival time cost function C_A can admit positive discontinuities as long as $C_A(t^+) \geq C_E$, i.e. the right side of the discontinuity is not less than the equilibrium cost. Therefore, arrival time functions with step incentives/tolls still admit equilibrium solutions, as long as C_I is picked such that $C_A(t^+) \geq C_E$ at each discontinuity.

Proposition 6.12. ***Demand shift with step incentives***
The exiting vehicle equilibrium can be shifted such that the support of the equilibrium flow is either to the left of some time t_{min} or the right of some time t_{max} using the following step incentives.

- *Left shift:*

$$C_I = \begin{cases} I < 0 & \text{if } t_{min} - t_{\text{supp}(\lambda)} \leq t \leq t_{min} \\ 0 & \text{otherwise} \end{cases} \tag{6.33}$$

where $t_{\text{supp}(\lambda)}$ is the length of the support of λ and $I = -\max(C_A(t) : t \in (t_{min} - t_{\text{supp}(\lambda)},\ t_{min})) - \min(C_A)$.

- *Right shift:*

$$C_I = \begin{cases} I < 0 & \text{if } t_{max} \leq t \leq t_{max} + t_{\text{supp}(\lambda)} \\ 0 & \text{otherwise} \end{cases} \tag{6.34}$$

where $t_{\text{supp}(\lambda)}$ *is the length of the support of* λ *and* $I = -\max(C_A(t) : t \in (t_{max},\ t_{max} + t_{\text{supp}(\lambda)})) - \min(C_A)$.

Proof. Consider the case of the left shift. Let C_A be the original arrival cost function and C_A^* be the new arrival cost function. Given the equilibrium λ, we know that the total number of exiting vehicles is $N \leq t_{\text{supp}(\lambda)} \cdot \mu_e$. If an incentive of $I = \max(C_A(t) : t \in (t_{min} - t_{\text{supp}(\lambda)},\ t_{min})) - \min(C_A(t))$ is given during the interval $t_{min} - t_{\text{supp}(\lambda)} \leq t \leq t_{min}$, the maximum arrival time cost C_A^* during this interval is $\min(C_A)$. Therefore, C_A^* contains an interval of $t_{\text{supp}(\lambda)}$ such that $C_A^* < \min(C_A)$ during this interval, and the $N \leq t_{\text{supp}(\lambda)} \cdot \mu_e$ vehicles can exit during this interval, which makes this an unique equilibrium solution. A similar argument can be used to prove the case of the right shift as well. □

Corollary 6. *Demand shift with step tolls We can show that the demand shift can also be achieved via a step toll using a similar argument.*

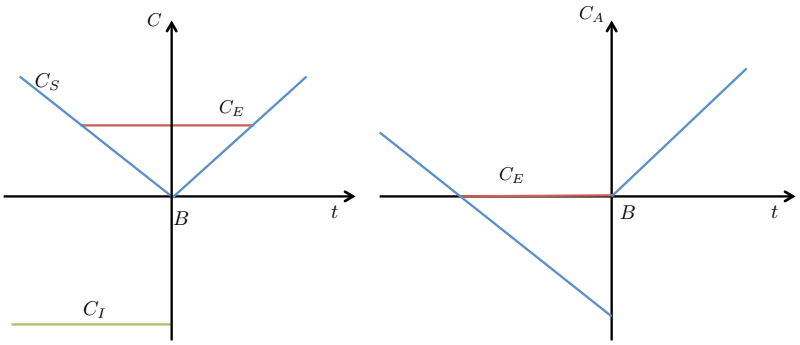


Figure 6.10: A step incentive strategy that shifts the the exiting vehicle flow to the left of the scheduled arrival time B for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and step incentive function C_I that corresponds the left shift of the equilibrium. **Right:** the corresponding arrival cost function C_A with the new flow distribution with equilibrium cost C_E.

Step incentives/tolls are inefficient for multiple reasons. As all the exiting vehicles in the incentive window must be given the same incentive, the vehicles that arrive close to the desired arrival are given a much larger incentive than needed. Consequently, the equilibrium solution requires that these vehicles occur a large queuing delay to compensate for the incentive. In fact, a step incentive can increase the total delay in the network, causing undesirable side effects such as increasing emissions in addition to the additional cost incurred. Furthermore, step incentives can not move a congested equilibrium to a congestion-free equilibrium.

However, the efficiency of step incentives/tolls can be improved by combining step incentives/tolls. A sequence of step incentives can be used to approximate the highway optimal incentive and obtain an equilibrium with a lower total incentive/toll cost σ_I. Figure 6.11 illustrates this. However, this still does not allow for a congestion free equilibrium.

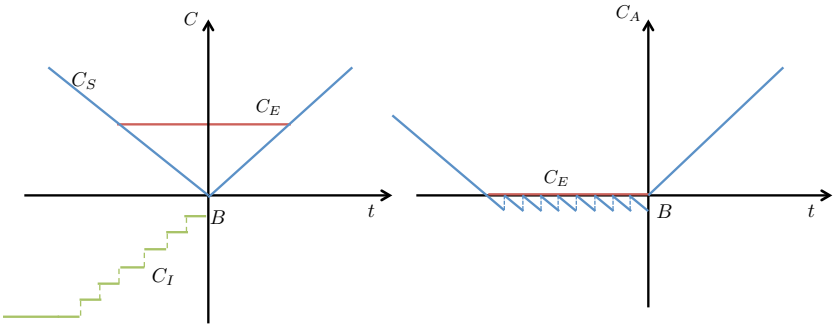


Figure 6.11: A more efficient step incentive strategy that shifts the the exiting vehicle flow to the left of the scheduled arrival time B for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and piecewise constant step incentive/toll function C_I that corresponds the left shift of the equilibrium. **Right:** the corresponding arrival cost function C_A with the new flow distribution with equilibrium cost C_E.

Furthermore, step incentives can also be mixed with step tolls to shift the equilibrium cost between the vehicles and the controlling agency for any shift. Figure 6.12 illustrates this.

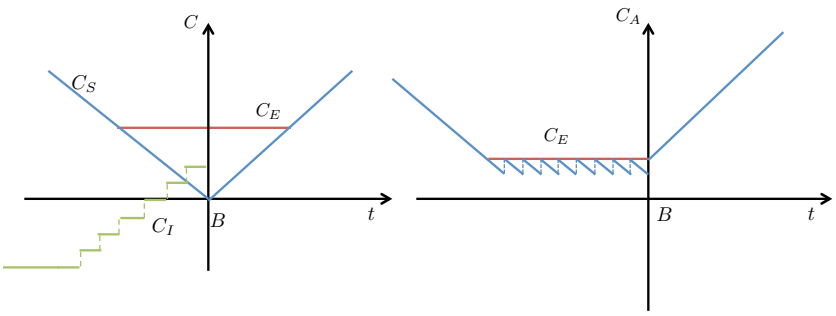


Figure 6.12: A step incentive/toll strategy that shifts the the exiting vehicle flow to the left of the scheduled arrival time B for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and piecewise constant step incentive/toll function C_I that corresponds the left shift of the equilibrium. **Right:** the corresponding arrival cost function C_A with the new flow distribution with equilibrium cost C_E.

Step schedule cost functions

The inefficiency of step incentives/tolls is a direct result of the assumption that the schedule time cost function C_S is continuous. However, in reality the actual schedule time cost incurred by commuters (imposed by employers) is likely to be discrete. In the event of discrete schedule time cost functions, a sequence of step incentives/tolls can be used to obtain a equilibrium that is congestion free for the highway traffic. Figure 6.13 illustrates this.

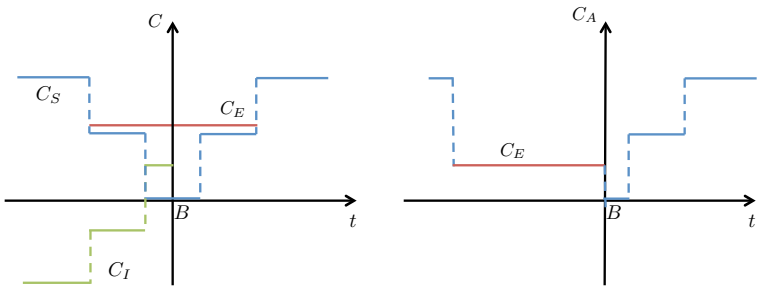


Figure 6.13: A step incentive/toll strategy that shifts the the exiting vehicle flow to the left of the scheduled arrival time B for a simple schedule cost function. **Left:** a simple schedule cost function C_S with linear earliness and lateness costs, an equilibrium cost C_E and piecewise constant step incentive/toll function C_I that corresponds the left shift of the equilibrium. **Right:** the corresponding arrival cost function C_A with the new flow distribution with equilibrium cost C_E.

In conclusion, we can make the following observations on controlling the departure time equilibrium using incentives and tolls.

- Continuous incentives/tolls can be used to obtain a congestion free equilibrium, time shift the exiting vehicle demand and to allocate the cost of the control between the controlling agency and the drivers at any ratio.
- Step incentives/tolls can be used to time shift the exiting vehicle demand and control the cost allocation, but can not be used to obtain a congestion free equilibrium for general schedule cost functions.
- If the schedule cost function is piecewise constant, then step incentives/tolls can be used to obtain a congestion free equilibrium.

Chapter 7

Conclusions and future work

The research pursued in this dissertation was motivated by the need for computationally tractable algorithms for vehicle routing and demand management in road networks, with the aim of improving the efficiency of existing network infrastructure, in the context of the *Connected Corridors* and *Mobile Millennium* projects. The contributions are spread across three research topics, 1) route planning with reliability guarantees, 2) system optimal dynamic traffic assignment, and 3) controlling user equilibrium departure times. We conclude this dissertation with some final remarks on each of these research topics and a discussion of possible extensions.

Route planning with reliability guarantees. The first part of this dissertation (Chapters 2 and 3) considers the computationally challenging reliable routing problem of maximizing the probability of on-time arrival, and presents algorithmic methods that improve the tractability of the problem over existing methods. This reasearch provides a step towards the eventual goal of implementing a real-time stochastic router in an operational setting.

The optimization techniques discussed in chapter 2 are based on the existence of an uniform strictly positive minimum link travel-time that allows us to compute the SOTA solution using a label-setting algorithm instead of a label-correcting successive approximations scheme. It also allows for batch computation of the convolution integrals, which is a key component of the optimization techniques. It is seen that the heterogeneity of the minimum link travel-times on a network can make the SOTA algorithm very sensitive to the order in which the nodes are treated. Thus, an optimal ordering algorithm is developed to find the update order that minimizes the computation time. Finally, a technique for efficiently computing convolutions of streaming signals, zero-delay convolution (ZDC), is combined with the optimal ordering to reduce the time complexity of each convolution product to $O(T\left(\log^2 T - \log^2 \delta_i\right)$, where δ_i is the minimum strictly positive loop travel-time for node i. Experimental results are provided to numerically justify the theoretical contributions.

Chapter 3 presents, what are to the best of our knowledge, the first results on using preprocessing techniques for the stochastic on-time arrival (SOTA) problem. We discuss the difficulties in applying the preprocessing techniques commonly used in the deterministic

setting to the SOTA problem, and identify two techniques (*reach* and *arc-flags*) that can be adapted to the stochastic setting. We also present an extension of the *reach* technique that enables more aggressive pruning of the search space at the cost of some additional memory. Experimental results show that the preprocessing methods can provide up to a order of magnitude improvement in runtime for the networks we have considered and time budgets on the order of 2000 seconds. The main limitation of this work is the inability to perform the preprocessing in a computationally efficient manner, making the technique intractable for large networks with large time budgets. However, we discuss the potential for faster precomputation using efficient heuristic schemes, and present one such example for *arc-flags*. Furthermore, reimplementing the algorithms with more low level code optimization and memory efficiency on more powerful hardware systems should allow us to understand the behavior of these algorithms on larger networks for larger time budgets. In addition, we are confident that refinements to the preprocessing schemes, such as more targeted reach partitioning schemes and better strategies for selecting Arc-flag regions, will also provide further gains. While these precomputation techniques reduce the computation time for the SOTA problem, there are no theoretical guarantees of the speedup that can be achieved. In the deterministic shortest path problem, it has been shown that precomputation techniques are provably efficient in networks with a low *Highway Dimension* [2]. Ideally, we would like to obtain a set of characteristics for both the network structure and the underlying probability distributions such that the precomputation techniques can be shown to provably improve the query time. Such a result would of course be more complex than the result in [2], due to the influence of the stochastic edge weights in addition to the network structure.

While most of the results presented in this section focus on exact solutions to the SOTA problem, practical routing applications rarely require the problem to be solved exactly. The tractability of the problem has the potential to be improved significantly using approximation algorithms. Furthermore, initial experiments on solving the SOTA problem in parallel on a GPU [1] have been promising and combining this with the preprocessing methods described in this section are the interesting area for future exploration. The current solutions to the SOTA problem also assume that the network satisfies the first-in-first-out (FIFO) property. However, transit networks for example do not satisfy this property, since the optimal strategy might involve waiting at a station for an express bus or train to arrive. Therefore, the solution methods need to be adapted to accommodate such networks. Finally, the solution to the SOTA problem can also be used as a efficient lower bound for computing the much harder path-based SOTA problem [104], where we wish to obtain a fixed path as opposed to a policy. This provides another natural extension to the work presented in this dissertation.

System optimal traffic assignment with partial control. The second part of this dissertation (Chapter 4) presents a model and optimization framework for solving the system optimal dynamic traffic assignment (SO-DTA) problem with partial control for general networks with horizontal queuing dynamics. The model only requires full origin-destination (OD) information for the fraction of the demand that is controllable, with aggregate split

ratios being sufficient for the non-controllable (selfish) demand.

One assumption of the current setup is that behavior of the selfish demand is prescribed by fixed aggregate split ratios that do not react to the control. This is a reasonable assumption in non-recurrent situations such as rerouting vehicles due to an accident, but unrealistic if the routing control is done on the repeated basis, for example during the daily rush hours. In such a situation, the selfish demand would react to the new state of the network and change their routes accordingly to minimize individual travel-times, with the potential of once again creating an inefficient network utilization. These dynamics are typically modeled as a leader-follower or Stackelberg game, where the leader tries to optimize the efficiency of the network utilization with the knowledge that the follower will act selfishly. As mentioned in the introduction, finding the optimal control for a Stackelberg game is NP-Hard in the size of the network for the class of increasing latency functions even in the case of the static problem [113] and it is common to use approximate strategies [113, 130]. Adapting these strategies to efficiently compute the solution to the SO-DTA-PC problem is an important next step to address.

The traffic dynamics in the proposed model are given by a Godunov discretization of the Lighthill-Williams-Richards (LWR) partial differential equation with a triangular flux function and a corresponding multi-commodity junction solver. The sparsity pattern of the resulting forward system enables computing the gradient of the system with linear computational complexity and memory using the discrete adjoint method. The junction solver presented in the dissertation considers $(1 \times m), (n \times 1)$ and $(2 \times m)$ junctions because a unique solution does not always exist for a general $(n \times m)$ junction. There are existing junction solvers that can accommodate general $(n \times m)$ junctions, but require a parameter tuning step to determine the weights corresponding to the dual objectives at the junction, maximizing flow and minimizing the violation of the priority rule. One important extension to the model presented in this dissertation is to develop a junction model that can accommodate general $(n \times m)$ junctions, but uses a natural trade-off between flow maximization and the priority rule, eliminating the need to prescribe weights.

The proposed framework for solving the SO-DTA-PC problem is used to find the optimal vehicle rerouting strategy in response to a capacity loss such as an accident and show the congestions reductions that can be achieved. Numerical results are presented for a test network and Interstate 210 in Southern California. The system is implemented in the *Connected Corridors* system at UC Berkeley, a partnership between the California Department of Transportation (CalTrans) and California Partners for advanced transportation technology (PATH) for *Integrated Corridor Management* (ICM).

The morning commute problem. The third and final part of the dissertation (Chapters 5, 6) tackle the problem of spill-back from a congested off-ramp during the morning commute and incentive/tolling strategies to minimize the negative impact of this local phenomenon on the rest of the network.

To solve the morning commute problem at an off-ramp, we first need to be able to analytically prescribe the delays at the off-ramp as a function of the demand at the source. Chapter 5 presents a mathematical framework for modeling traffic flow through a network with a single source and multiple sinks. The model satisfies the standard laws of flow dynamics such as the FIFO property and is shown to lead to a well-posed problem with a unique solution. The main benefit of this framework is the ability to analytically describe the delays at each junction as a function of the boundary flows at any other upstream junction and the delay over any sub-path with respect to the boundary flow at the source node of the sub-path. This is a critical requirement when solving control and optimization problems over a network, since solving an optimization problem over simulation models is generally intractable in terms of computational complexity. The versatility of computing the delays as a function of the inflow at any point in the network is achieved though a mathematical framework for time mapping the delays. While this framework is sufficient for solving the morning commute problem at an off-ramp, it is fairly limited by the single source assumption. The time mapping framework that is used can, however, be generalized to any non-cyclic (tree) network. Thus, the next step would be to introduce merging dynamics into the framework to obtain a more general network model.

Chapter 6 considers the spill-back from a congested off-ramp and the resulting throughput loss on a highway when the departure times of the exiting vehicles form an equilibrium with respect to the total cost incurred by the exiting vehicles. Existence and uniqueness properties are proved for a general class of cost functions that allow for local minima and discontinuities, which is a new result for the equilibrium departure time problem, even in the case of a single bottleneck as opposed to an off-ramp junction.

However, the junction considered in this problem takes a very specific form, where there is an exit lane prior to the off-ramp and the non-exiting freeway demand is assumed to be less than the capacity of the freeway (not counting the exit lane). A natural extension of this work is to extend the analysis to off-ramp junctions without an exit lane on the freeway and to the Newell off-ramp model [102], where the through traffic does not satisfy the FIFO property. We have already shown that the framework from chapter 5 can be used to model the Newell off-ramp junction (see section 5.4.2). What remains to be shown is the existence and uniqueness of the departure time equilibrium for these junctions, which is a more involved due to the more complicated queuing dynamics that arise.

The analysis provided in this chapter, informs demand side congestion management strategies via congestion pricing (tolling) or incentives at an off-ramp. We also show how tolling and incentives can be used in tandem to achieve a wide variety of demand shifts for the vehicles that exit the highway at the congested off-ramp and thereby decrease congestion for the vehicles that continue on the freeway. The cost of the demand shift can be distributed in any ratio between the traffic management authority and the commuters by picking the appropriate incentive/tolling function. This allows for revenue neutral management strategies that are viewed more favorably with respect to public policy considerations.

Bibliography

[1] Abeydeera, M., and Samaranayake, S. GPU parallalization of the stochastic on-time arrival problem. In *Proceedings of the 21st IEEE Conference on High Performance Computing (HiPC), Goa, India* (2014). 9, 162

[2] Abraham, I., Fiat, A., Goldberg, A. V., and Werneck, R. F. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA10), Society for Industrial and Applied Mathematics* (2010). 2, 32, 162

[3] Adas, A. Traffic models in broadband networks. *Communications Magazine, IEEE 35*, 7 (1997), 82–89. 104

[4] Arnott, R., de Palma, A., Lindsey, R., and Palma, A. D. Departure time and route choice for the morning commute. *Transportation Research Part B: Methodological 24*, 3 (1990), 209–228. 5

[5] Assaf, D., and Levikson, B. Closure of Phase Type Distributions Under Operations Arising in Reliability Theory. *The Annals of Probability 10*, 1 (1982), 265–269. 18

[6] Astarita, V. A continuous time link model for dynamic network loading based on travel time function. In *13th International Symposium on Transportation and Traffic Theory* (Lyon, France, 1996), pp. 79–102. 3

[7] Aswani, A., and Tomlin, C. Game-theoretic routing of GPS-assisted vehicles for energy efficiency. In *American Control Conference (ACC), 2011* (2011), IEEE, pp. 3375–3380. 4

[8] Aubin, J. P. *Viability Theory*. Springer, 2001. 32

[9] Bast, H., Funke, S., and Matijevic, D. Transit-ultrafast shortest-path queries with linear-time preprocessing. *9th DIMACS Implementation Challenge* (2006). 49, 51

[10] Bauer, R., and Delling, D. Sharc: Fast and robust unidirectional routing. *Journal of Experimental Algorithmics (JEA) 14* (2009), 4. 49, 51

[11] Bayen, A. M., Raffard, R. L., and Tomlin, C. J. Adjoint-based control of a new Eulerian network model of air traffic flow. *Control Systems Technology, IEEE Transactions on 14*, 5 (2006), 804–818. 63

[12] Beckman, M., McGuire, C. B., and Winsten, C. B. *Studies in the Economics of Transportation*. Yale University Press, New Haven, 1956. 4

[13] Bellman, R. E., and Kalaba, R. E. *Numerical Inversion of the Laplace Transform.*

American Elsevier Publishing Company, 1966. 18

[14] Bertsekas, D. P. *Nonlinear programming.* Athena Scientific, 1999. 62, 63

[15] Bertsekas, D. P. *Dynamic Programming and Optimal Control.* Athena Scientific, 2005. 3

[16] Bertsekas, D. P., and Tsitsiklis, J. N. *Neuro-dynamic Programming.* Athena Scientific, 1996. 16

[17] Block, H. W., and Savits, T. H. The IFRA Closure Problem. *The Annals of Probability 4*, 6 (1976), 1030–1032. 18

[18] Boese, K. D., Kahng, A. B., and Muddu, S. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters 16*, 2 (1994), 101 – 113. 85, 98

[19] Borokhov, P., Blandin, S., Samaranayake, S., Goldschmidt, O., and Bayen, A. An adaptive routing system for location-aware mobile devices on the road network. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (Oct. 2011), IEEE, pp. 1839–1845. 9

[20] Boyd, S., and Vandenberghe, L. *Convex optimization.* Cambridge university press, 2004. 62

[21] Braess, D. Über ein Paradoxon aus der Verkehrsplanung. *Mathematical Methods of Operations Research 12*, 1 (1968), 258–268. 1, 4

[22] Carey, M. Nonconvexity of the dynamic traffic assignment problem. *Transportation Research Part B: Methodological 26*, 2 (1992), 127–133. 62

[23] Cassidy, M. J., Anani, S. B., and Haigwood, J. M. Study of freeway traffic near an off-ramp. *Transportation Research Part A: Policy and Practice 36*, 6 (2002), 563–572. 5

[24] Chang, C.-S. *Performance guarantees in communication networks.* Springer, 2000. 104

[25] Chang, G.-L., and Mahmassani, H. S. Travel time prediction and departure time adjustment behavior dynamics in a congested traffic system. *Transportation Research Part B: Methodological 22B* (1988), 217–232. 5

[26] Chen, C., Petty, K., Skabardonis, A., Varaiya, P., and Jia, Z. Freeway performance measurement system: mining loop detector data. *Transportation Research Record: Journal of the Transportation Research Board 1748*, 1 (2001), 96–102. 99

[27] Chorus, C. G., Molin, E. J. E., and Van Wee, B. Use and effects of Advanced Traveller Information Services (ATIS): a review of the literature. *Transport Reviews*, 26.2 (2006), 127–149. 2

[28] Connected Corridors. `http://connected-corridors.berkeley.edu/`. Accessed: 2013-05-27. 2, 6

[29] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms.* The MIT Press, 2001. 25, 31, 32

[30] Cronin, B., Mortensen, S., and Thompson, D. The shortest route through a network with time-dependent internodal transit times. *Public Roads 71.5* (2008). 6

[31] Daganzo, C. The cell transmission model: A dynamic representation of highway

traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological 28*, 4 (1994), 269–287. 4, 63

[32] Daganzo, C. F. The uniqueness of a time-dependent equilibrium distribution of arrivals at a single bottleneck. *Transportation science 19*, 1 (1985), 29–37. 5, 144

[33] Daganzo, C. F. THE CELL TRANSMISSION MODEL , PART II : NETWORK TRAFFIC. 79–93. 104, 105

[34] Daganzo, C. F. The cell transmission model, part II: Network traffic. *Transportation Research Part B 29*, 2 (1995), 79–93. 4, 63

[35] D'Angelo, G., Frigioni, D., and Vitale, C. Dynamic arc-flags in road networks. In *Experimental Algorithms*. Springer, 2011, pp. 88–99. 54

[36] Dean, B. C. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. *Networks 44* (2004), 41–46. 3, 19, 20

[37] Dean, B. C. Speeding up Stochastic Dynamic Programming with Zero-Delay Convolution. *Algorithmic Operations Research 5*, 2 (2010). 8, 14, 41, 42

[38] Delle Monache, M. L., Reilly, J., Samaranayake, S., Krichene, W., Goatin, P., and Bayen, A. A PDE-ODE Model for a Junction with Ramp Buffer. *SIAM Journal on Applied Mathematics (in review)* (2013). 9, 63

[39] Delling, D., Sanders, P., Schultes, D., and Wagner, D. Engineering Route Planning Algorithms. 117–139. 3

[40] der Zijpp, N. V., and Koolstra, K. Multiclass continuous-time equilibrium model for departure time choice on single-bottleneck network. *Transportation Research Record: Journal of the Transportation Research Board 1783*, -1 (2002), 134–141. 5

[41] Dervisoglu, G., Gomes, G., Kwon, J., Horowitz, R., and Varaiya, P. Automatic Calibration of the Fundamental Diagram and Empirical Observations on Capacity. 6

[42] Dervisoglu, G., Gomes, G., Kwon, J., Horowitz, R., and Varaiya, P. Automatic calibration of the fundamental diagram and empirical observations on capacity. In *Transportation Research Board 88th Annual Meeting* (2009), no. 09-3159. 66

[43] Dijkstra, E. W. A Note on Two Problems on Connection with Graphs. *Numerishe Mathematik 1* (1959), 269–271. 2

[44] Dobbs, R., Smit, S., Remes, J., Manyika, J., Roxburgh, C., and Restrepo, A. Urban world : Mapping the economic power of cities. *Technical Report* (March 2011). 1

[45] Dreyfus, S. An appraisal of some shortest-path algorithms. *Operations Research 17* (1969), 395–412. 19

[46] Duffy, A. An Introduction to Gradient Computation by the Discrete Adjoint Method. Tech. rep., Florida State University, 2009. 8, 86

[47] Fan, Y., and Nie, Y. Optimal Routing for Maximizing the Travel Time Reliability. *Networks and Spatial Economics 6*, 3-4 (Aug. 2006), 333–344. 13, 15, 16, 35, 44

[48] Fan, Y. Y., Kalaba, R. E., and Moore, J. E. Arriving on Time. *Journal of Optimization Theory and Applications 127*, 3 (Dec. 2005), 497–513. 3, 18

[49] Flajolet, A., Blandin, S., and Jaillet, P. Robust Adaptive Routing Under

Uncertainty. *arXiv preprint arXiv:1408.3374* (2014). 3

[50] Frank, H. Shortest paths in probabilistic graphs. *Operations Research 17*, 4 (1969), 583–599. 3

[51] Frost, V. S., and Melamed, B. Traffic modeling for telecommunications networks. *Communications Magazine, IEEE 32*, 3 (1994), 70–81. 104

[52] Fu, L., and Rilett, L. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B 32*, 7 (1998), 499–516. 2

[53] Garavello, M., and Piccoli, B. *Traffic flow on networks*. American institute of mathematical sciences Springfield,, USA, 2006. 8, 65, 71

[54] Gardner, W. G. Efficient convolution without input-output delay. *Journal of the Audio Engineering Society 43*, 3 (1995), 127–136. 8, 14, 41

[55] Geisberger, R., Sanders, P., Schultes, D., and Delling, D. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)* (2008), vol. 2, Springer, pp. 319–333. 2, 32, 49, 50

[56] Giles, M. B. M., and Pierce, N. A. N. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion 65*, 3 (2000), 393–415. 8, 63

[57] Giles, M. B. M. B., and Pierce, N. A. N. Adjoint equations in CFD : duality , boundary conditions and solution behaviour. *AIAA paper 97*, 1850 (1997), 182–198. 8, 63

[58] Godunov, S. K. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik 89*, 3 (1959), 271–306. 65, 67

[59] Goldberg, A. V., Kaplan, H., and Werneck, R. F. Reach for A*: Efficient point-to-point shortest path algorithms. In *ALENEX* (2006), vol. 6, pp. 129–143. 50

[60] Goldberg, A. V., Kaplan, H., and Werneck, R. F. Better Landmarks within Reach. In *Workshop on Experimental Algorithms (WEA), Rome, Italy* (2007). 49, 52, 53, 55, 59

[61] Goldberg, A. V., and Werneck, R. F. F. Computing point-to-point shortest paths from external memory. In *ALENEX/ANALCO* (2005), pp. 26–40. 49

[62] Gomes, G., and Horowitz, R. Optimal freeway ramp metering using the asymmetric cell transmission model. *Transportation Research Part C: Emerging Technologies 14*, 4 (2006), 244–262. 62

[63] Hall, R. W. The fastest path through a network with random time-dependent travel times. *Transportation Science 20*, 3 (1986), 182. 2

[64] Hendrickson, C., and Kocur, G. Schedule delay and departure time decisions in a deterministic model. *Transportation Science 15*, 1 (1981), 62–77. 5

[65] Herring, R., Hofleitner, A., Amin, S., Nasr, T. A., Khalek, A. A., Abbeel, P., and Bayen, A. Using Mobile Phones to Forecast Arterial Traffic Through Statistical Learning. In *Transportation Research Board 89th Annyal Meeting, Washington D.C., January 10-14, 2010*. 6

[66] Herring, R., Hofleitner, A., Bayen, A., and Abbeel, P. Estimating arterial

traffic conditions using sparse probe data. In *13th International Conference on Intelligent Transportation Systems* (Madeira Island, Portugal, Sept. 2010), Ieee, pp. 929–936. 33, 39, 47

[67] Hilger, M., Köhler, E., Möhring, R. H., and Schilling, H. Fast point-to-point shortest path computations with arc-flags. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge 74* (2009), 41–72. 49, 50, 54, 59

[68] Hunter, T., Abbeel, P., and Bayen, A. M. The path inference filter: Model-based low-latency map matching of probe vehicle data. In *Algorithmic Foundations of Robotics X* (Heidelberg, Germany, 2012), Springer Tracts in Advanced Robotics, Springer-Verlag, pp. 591:1–607:17. 6, 57

[69] Hunter, T., Hofleitner, A., Reilly, J., Krichene, W., Thai, J., Kouvelas, A., Abbeel, P., and Bayen, A. Arriving on time: estimating travel time distributions on large-scale road networks. *arXiv preprint arXiv:1302.6617* (2013). 6

[70] Jain, S., Fall, K., and Patra, R. Routing in a delay tolerant network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2004), SIGCOMM '04, ACM, pp. 145–158. 104

[71] Jameson, A., and Martinelli, L. *Aerodynamic shape optimization techniques based on control theory*. Springer, 2000. 63

[72] Kelly, F. P. Network routing. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences 337*, 1647 (1991), 343–367. 4

[73] Korilis, Y. A., Lazar, A. A., and Orda, A. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking (TON) 5*, 1 (1997), 161–173. 4

[74] Koutsoupias, E., and Papadimitriou, C. Worst-case equilibria. In *Proceedings of the 16th annual conference on Theoretical aspects of computer science* (1999), Springer-Verlag, pp. 404–413. 4

[75] Krichene, W., Reilly, J., Amin, S., and Bayen, A. M. Stackelberg Routing on Parallel Networks with Horizontal Queues. *IEEE Transactions on Automatic Control (in review)* (2013). 4

[76] Kuwahara, M. Equilibrium queueing patterns at a two-tandem bottleneck during the morning peak. *Transportation Science 24*, 3 (1990), 217–229. 5

[77] Lago, A., and Daganzo, C. F. Spillovers, merging traffic and the morning commute. *Transportation Research Part B: Methodological 41*, 6 (2007), 670–683. 5

[78] Lebacque, J. P. The godunov scheme and what it means for first order traffic flow models. In *Proceedings of the 13th International Symposium on Transportation and Traffic Theory* (1996), pp. 647–678. 104, 105

[79] L'Ecuyer, P. Stochastic Simulation in Java, `http://www.iro.umontreal.ca/~{}simardr/ssj/indexe.html`, (2008). 33, 44

[80] Lespiau, J.-B., Samaranayake, S., and Bayen, A. Solving the dynamic user equilibrium problem via sequential convex optimization for parallel horizontal queuing networks. In *Transportation Research Board 88th Annual Meeting* (2015). 9

[81] Leveque, R. *Finite volume methods for hyperbolic problems*. Cambridge University Press, Cambridge, UK, 2002. 67

[82] Lighthill, M. J., and Whitham, G. B. 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 (1955), 317–345. 4, 63, 65

[83] Lindsey, R. Existence, uniqueness, and trip cost function properties of user equilibrium in the bottleneck model with multiple user classes. *Transportation science 38*, 3 (2004), 293–314. 5

[84] Lo, H. K., Chang, E., and Chan, Y. C. Dynamic network traffic control. *Transportation Research Part A: Policy and Practice 35*, 8 (2001), 721 – 744. 104

[85] Loui, R. P. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM. 26*, 9 (1983), 670–676. 2

[86] M. Kobitzsch, S. Samaranayake, D. S. Pruning Techniques for the Stochastic on-time Arrival Problem - An Experiemental Study. 9

[87] Mahmassani, H., and Herman, R. Dynamic user equlibrium departure ime and route choice on idealized traffic arterials. *Transportation Science 18* (1984), 362–384. 5

[88] Mahmassani, H. S., and Chang, G.-L. Experiments with departur time choice dynamics of urban commuters. *Transportation Research Part B: Methodological 20B* (1986), 297–320. 5

[89] Marti, R. Multi-start methods. In *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger, Eds., vol. 57 of *International Series in Operations Research and Management Science*. Springer US, 2003, pp. 355–368. 85, 98

[90] Merchant, D. K., and Nemhauser, G. L. A Model and an Algorithm for the Dynamic Traffic Assignment Problems. *Transportation science 12*, 3 (1978), 183–199. 3, 104

[91] Merchant, D. K., and Nemhauser, G. L. Optimality conditions for a dynamic traffic assignment model. *Transportation Science 12*, 3 (1978), 183–199. 3

[92] Miller, M. A., and Skabardonis, A. *San Diego I-15 Integrated Corridor Management (ICM) System: Stage II (analysis, Modeling, and Simulation*. California PATH Program, Institute of Transportation Studies, University of California at Berkeley, 2010. 6

[93] Miller-Hooks, E. D., and Mahmassani, H. S. Least Expected Time Paths in Stochastic, Time-Varying Transportation Networks. *Transportation Science 34*, 2 (2000), 198–215. 2

[94] Mitchell, I. M., Bayen, A. M., and Tomlin, C. J. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on 50*, 7 (2005), 947–957. 104

[95] Mobile Millennium. http://traffic.berkeley.edu, 2008. 2, 7, 14, 47

[96] Morton, K. W., and Mayers, D. F. *Numerical solution of partial differential equations: an introduction*. Cambridge university press, 2005. 100

[97] Muralidharan, A., Dervisoglu, G., and Horowitz, R. Freeway traffic flow

simulation using the link node cell transmission model. In *American Control Conference, 2009. ACC'09.* (2009), IEEE, pp. 2916–2921. 6

[98] Nagurney, A., Dong, J., and Zhang, D. A supply chain network equilibrium model. *Transportation Research Part E: Logistics and Transportation Review 38*, 5 (2002), 281–303. 104

[99] Newell, G. F. The morning commute for nonidentical travelers. *Transportation Science 21*, 2 (1987), 74–88. 5

[100] Newell, G. F. Traffic flow for the morning commute. *Transportation Science 22* (1988), 47–58. 5

[101] Newell, G. F. Delays caused by a queue at a freeway exit ramp. *Transportation Research Part B: Methodological 33*, 5 (1999), 337 – 350. 134, 136

[102] Newell, G. F. Delays caused by a queue at a freeway exit ramp. *Transportation Research Part B: Methodological 33*, 5 (1999), 337–350. 164

[103] Nie, Y., and Fan, Y. Arriving-on-Time Problem: Discrete Algorithm That Ensures Convergence. *Transportation Research Record 1964*, 1 (Jan. 2006), 193–200. 13, 16, 17, 19, 24, 35, 44

[104] Niknami, M., Samaranayake, S., and Bayen, A. Tractable Pathfinding for the Stochastic On-Time Arrival Problem. *arXiv preprint arXiv:1408.4490* (2014). 9, 162

[105] Nikolova, E., Brand, M., and Karger, D. R. Optimal route planning under uncertainty. In *ICAPS* (2006), vol. 6, pp. 131–141. 3

[106] Orda, A., and Rom, R. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM) 37*, 3 (1990), 607–625. 19

[107] Parmentier, A., Samaranayake, S., Xuan, Y., and Bayen, A. M. A mathematical framework for delay analysis in single source networks. *Technical report* (2014. `http://dx.doi.org/10.7922/G2RN35S6`). 9

[108] Peeta, S., and Ziliaskopoulos, A. Foundations of Dynamic Traffic Assignment : The Past , the Present and the Future. *Networks and Spatial Economics* (2001), 233–265. 4, 104

[109] Raffard, R. An adjoint-based parameter identification algorithm applied to planar cell polarity signaling. *Automatic Control*, January (2008), 109–121. 8, 63

[110] Reilly, J., Monache, M. L. D., Samaranayake, S., Krichene, W., Gaotin, P., and Bayen, A. An efficient method for coordinated ramp metering using the discrete adjoint method. *Journal of Optimization Theory and Applications, in review* (2013). 6, 7, 9, 65, 104

[111] Richards, P. I. Shock waves on the highway. *Operations research 4*, 1 (1956), 42–51. 4, 63, 65

[112] Riedmiller, M., and Braun, H. Rprop-a fast adaptive learning algorithm. In *Proceedings of the International Symposium on Computer and Information Science VII), Universitat* (1992), Citeseer. 96

[113] Roughgarden, T. Stackelberg scheduling strategies. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing* (2001), ACM, pp. 104–113. 4,

63, 163

[114] Roughgarden, T. The Price of Anarchy is Independent of the Network. *Computer*, May (2002), 1–24. 4

[115] Roughgarden, T. On the severity of Braess's paradox: designing networks for selfish users is hard. *Journal of Computer and System Sciences 72*, 5 (2006), 922–953. 4

[116] Roughgarden, T., and Tardos, É. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior 47*, 2 (2004), 389–403. 4

[117] Sabran, G., Samaranayake, S., and Bayen, A. Precomputation techniques for the stochastic on-time arrival problem. In *SIAM Meeting on Algorithm Engineering and Experiments* (2014). 8

[118] Samaranayake, S., Blandin, S., and Bayen, A. A tractable class of algorithms for reliable routing in stochastic networks. *Transportation Research Part C: Emerging Technologies 20*, 1 (2011), 199–217. 8, 55

[119] Samaranayake, S., Blandin, S., and Bayen, A. Speedup Techniques for the Stochastic on-time Arrival Problem. In *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (2012), vol. 25 of *OpenAccess Series in Informatics (OASIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 83–96. 8

[120] Samaranayake, S., Blandin, S., Bayen, A., Samitha Samaranayake, Sébastien Blandin, and Alexandre Bayen. Learning the dependency structure of highway networks for traffic forecast. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on* (Dec. 2011), pp. 5983–5988. 10

[121] Samaranayake, S., Holstius, D., Monteil, J., Tracton, K., Glaser, S., Seto, E., and Bayen, A. Real-time estimation of pollution emissions and dispersion from highway traffic. *Computer-Aided Civil and Infrastructure Engineering 29* (August 2014), 546–558. 10

[122] Samaranayake, S., Krichene, W., Reilly, J., Delle Monache, M. L., Gaotin, P., and Bayen, A. System Optimal Dynamic Traffic Assignment with Partial Compliance (SO-DTA-PC). *Technical report* (2014. `http://dx.doi.org/10.7922/G23X84KV`). 7, 8, 104

[123] Samaranayake, S., Parmentier, A., Xuan, Y., and Bayen, A. M. Solving the user equilibrium departure time problem at an off-ramp with incentive compatible cost functions. *Technical report* (2014. `http://dx.doi.org/10.7922/G2057CVF`). 9

[124] Schrank, D., Lomax, T., and Turner, S. Texas Transportation Institute 2010 urban mobility report. *College Station, TX: Texas Transportation Institute, A&M University* (2010). 1

[125] Sedgewick, R. *Algorithms in C*. Addison Wesley Publishing Company, 1990. 31

[126] Simaiakis, I., and Balakrishnan, H. Queuing models of airport departure processes for emissions reduction. In *AIAA Guidance, Navigation and Control Conference and Exhibit* (2009). 104

[127] Smith, M. J., and Smith, J. The existence of a time-dependent equilibrium distri-

bution of arrivals at a single bottleneck. *Transportation science 18*, 4 (1984), 385–394. 5, 144

[128] Stone, H. S. Multiprocessor scheduling with the aid of network flow algorithms. *Software Engineering, IEEE Transactions on*, 1 (1977), 85–93. 104

[129] Stone, M. H. The generalized weierstrass approximation theorem. *Mathematics Magazine 21*, 5 (1948), 237–254. 112

[130] Swamy, C. The effectiveness of Stackelberg strategies and tolls for network congestion games. In *Symposium on Discrete Algorithms: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), vol. 7, pp. 1133–1142. 4, 63, 163

[131] Vickrey, W. S., and William S. Vickrey. Congestion theory and transport investment. *The American Economic Review 59*, 2 (1969), 251–260. 4, 5, 9, 139, 144

[132] Wachter, A., and Biegler, L. T. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming.* 2005. 86, 98

[133] Waller, S. T., and Ziliaskopoulos, A. K. On the Online Shortest Path Problem with Limited Arc Cost Dependencies. *Networks 40*, 4 (2002), 216–227. 2

[134] Wardrop, J. G. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers 1* (1952), 325–378. 4

[135] Wendykier, P. JTransforms library, `http://sites.google.com/site/piotrwendykier/software/jtransforms`, (2009). 33, 44

[136] Work, D., Blandin, S., Tossavainen, O.-P., Piccoli, B., and Bayen, A. A traffic model for velocity data assimilation. *AMRX Applied Mathematics Research eXpress 1* (2010), 1–35. 33, 104

[137] Work, D. B. A traffic model for velocity data assimilation. *Cell 00*, 0000, 1–21. 6

[138] Wu, C.-J., Schreiter, T., and Horowitz, R. Multiple-clustering armax-based predictor and its application to freeway traffic flow prediction. In *American Control Conference (ACC), 2014* (2014), IEEE, pp. 4397–4403. 6

[139] Yperman, I., Logghe, S., and Immers, B. Dynamic congestion pricing in a network with queue spillover. In *Proc. 12th World Congress on Intelligent Transportation Systems, San Francisco* (2005). 5

[140] Ziliaskopoulos, A. K. A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science 34*, 1 (2000), 37–49. v, 62, 96