Dynamic Route Guidance Algorithms for Robust Roadway Networks

by

Agop Koulakezian

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2016 by Agop Koulakezian
Abstract

Dynamic Route Guidance Algorithms for Robust Roadway Networks

Agop Koulakezian

Doctor of Philosophy

Graduate Department of Electrical and Computer Engineering
University of Toronto
2016

This Thesis focuses on developing robust dynamic route guidance algorithms to reduce traffic congestion in roadway networks. While recurring traffic congestion is normally the focus in planning and investment decisions, almost half of traffic congestion is caused by non-recurring traffic disturbances, primarily caused by incidents, vehicle breakdowns, extreme weather events, etc. In order to reduce traffic congestion, we focus on the problem of understanding the effect of traffic disturbances and reducing their impact on roadway networks. We introduce a systematic framework for defining the context of robustness based on the severity, frequency and predictability of traffic disturbances and for developing a robust design for roadway networks based on network design goals. We also present methods to speed up traffic assignment algorithms through compiler optimizations and parallelism, to efficiently measure the effect of traffic disturbances, and enable real-time ITS applications including dynamic route guidance systems. Next, we introduce a hybrid metric for measuring robustness in a roadway network by extending the shortest-path betweenness metric from network science, and augmenting it with links weights based on dynamic traffic flow metrics. Finally, we implement a robust dynamic traffic assignment algorithm for roadway networks based on this metric and test it on a large-scale calibrated network model for the Greater Toronto Area. Performance results show that the robust traffic assignment algorithm reduces vehicle travel times compared to existing traffic assignment algorithms, with and without the presence of disturbances in the form of traffic incidents. This makes a strong case for traffic planners and operators to use robust dynamic route guidance systems within actual implementations of real-time ITS strategies to help proactively alleviate traffic congestion due to disturbances.
To my Loving Parents and Caring Brothers

To Hillary

Your motivation and support make everything possible!
Acknowledgements

I would like to express my deep appreciation to my Supervisor, Professor Alberto Leon-Garcia, for being an amazing guide and teacher throughout the course of my graduate studies. His belief in me and his valuable questions have helped me grow more independent in my research and to focus on solving challenging real-life problems. I will forever be thankful for his supervision and contribution to my academic and professional life.

I would also like to thank my committee members, Professors Ben Liang, Elvino Sousa, David Michelson and Baher Abdulhai for their time and valuable feedback.

I express grate gratitude to my parents for their continuous motivation and support and to my brothers who always gave their all so that I earn the best education.

I am very appreciative for Dr. Ali Tizghadam for his weekly support and creative ideas that have helped me develop my research and Thesis and for Professor Baher Abdulhai and Dr. Hossam Abdelgawad for sharing their ITS industry experiences and providing great help in reviewing many of my research papers.

I would like to thank Billy Graydon, Professor Yi-Chang Chiu and Ye Tian for their support with the DynusT traffic simulator used in this thesis.

In addition, I would like to thank Hadi Bannazadeh, Armin Ghayoori, Houman Rastegarfar, Weiwei Li, Nadeem Abji, Hazem Soliman, Mo Faraji, Ali Shariat, Thomas Lin, Aya Aboudina, Kasra Rezaee, and Mohamed Elshenawy for their creative research ideas and support in the lab.

Lastly, I would like to say a special thank you to Hillary Armstrong, who supported me throughout this journey, and along with Elias Bou Harb and Saleem Khouri, have been the inspiration and rock I needed in developing this thesis.
# Contents

1 Introduction ........................................ 1
   1.1 Motivation ........................................ 1
   1.2 Research Objectives ................................ 4
   1.3 Thesis Contributions ............................... 5
   1.4 Thesis Outline .................................... 7

2 Background .......................................... 8
   2.1 Advanced Traveller Information Systems ............ 8
   2.2 Network Science ................................... 10
       2.2.1 Graph Theory .................................. 10
       2.2.2 Network Properties .............................. 11
       2.2.3 Network Models .................................. 13
   2.3 Basics of Traffic Assignment in Transportation Networks ......................... 14
   2.4 Classical Traffic Assignment Methods for Transportation Networks ............ 17

3 Robust Network Design for Roadway Networks ........... 21
   3.1 The Need for a Systematic Framework for Robust Network Design ............... 21
   3.2 Context of Robustness ............................... 22
       3.2.1 Sensitivity of Roadway Networks ............... 22
       3.2.2 Severity of Traffic Disturbances ............... 23
       3.2.3 Frequency of Traffic Disturbances .............. 24
       3.2.4 Robustness of Roadway Networks ............... 24
   3.3 Robust Network Design .............................. 26
       3.3.1 Long-term Robust Network Design ................ 26
       3.3.2 Short-term Robust Network Design ............... 29
5.3 Developing Algorithms to Efficiently Calculate the PK-BC Metric . . . . . . . . . . . . . 67
5.4 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
  5.4.1 Performance of Algorithms Calculating the PK-BC Metric . . . . . . . . . . . . . . 69
  5.4.2 Analyzing the PK-BC Metric in the GTA Network . . . . . . . . . . . . . . . . . . . 70
  5.4.3 Analyzing the HCI Metric in the GTA network . . . . . . . . . . . . . . . . . . . . 72
5.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

6 Implementation of Robust Dynamic Traffic Assignment for Roadway Networks 77
  6.1 The Need for a Large-scale Robust DTA Implementation . . . . . . . . . . . . . . . . . . 77
  6.2 DTA Model in DynusT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
    6.2.1 Components of the DynusT DTA model . . . . . . . . . . . . . . . . . . . . . . . . 79
    6.2.2 DTA Traffic Assignment Process in DynusT . . . . . . . . . . . . . . . . . . . . . . 81
  6.3 Robust DTA Implementation Within DynusT . . . . . . . . . . . . . . . . . . . . . . . . 82
    6.3.1 Requirements for Robust DTA Implementation in Roadway Networks . . . . . . . 82
    6.3.2 HCI-based Robust DTA Implementation in DynusT . . . . . . . . . . . . . . . . . . 82
  6.4 Performance Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
    6.4.1 MPL Network Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
    6.4.2 GTA Network Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
  6.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95

7 Conclusions and Future Work 96
  7.1 Contribution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
    7.1.1 Robust Network Design for Roadway Networks . . . . . . . . . . . . . . . . . . . 97
    7.1.2 Speedup of Dynamic Traffic Assignment Simulation for Real-Time ITS Applications 97
    7.1.3 A Hybrid Approach for Robustness in Roadway Networks . . . . . . . . . . . . . 98
  7.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
    7.2.1 Studying the Effect of Increasing Demand . . . . . . . . . . . . . . . . . . . . . . 100
    7.2.2 Mobile Application Providing Travel Time Reliability Based on HCI . . . . . . . 100
    7.2.3 City-Based Robust ATIS system . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

Bibliography 101
## List of Tables

2.1 Classical traffic assignment methods for transportation networks ........................................ 18
3.1 Robustness metrics for long-term roadway network design .................................................. 27
3.2 Robustness Metrics for Short-Term roadway network Design .............................................. 30
4.1 Sizes for the GTHA and GTA networks .................................................................................. 51
4.2 DynusT execution times on the GTHA network ................................................................... 55
4.3 DynusT execution times on the GTA network ....................................................................... 56
5.1 SP Betweenness calculation in the fish network ..................................................................... 61
5.2 PK-BC calculation in the fish network ................................................................................... 63
5.3 Betweenness calculation in the PL network ............................................................................ 65
5.4 Betweenness calculation in the MPL network ........................................................................ 66
6.1 PK-BC values in the MPL network ......................................................................................... 84
6.2 MPL Incident Scenarios ........................................................................................................ 85
6.3 Total Travel Times (hrs) in MPL network in 6 incident scenarios for UE assignment and
    HCl-based assignment using inverse of link travel time as weight ........................................ 86
6.4 Total Travel Times (hrs) in MPL network in 6 incident scenarios for UE assignment and
    HCl-based assignment using link speed as weight ................................................................. 87
6.5 GTA Incident Scenarios ........................................................................................................ 89
6.6 Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and
    HCl-based assignment using inverse of link travel time as weight ........................................ 89
6.7 Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and
    HCl-based assignment using link speed as weight ................................................................. 89
6.8 Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and
    HCl-based assignment using SP-BC and inverse of link travel time as weight ..................... 94
6.9 Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and HCI-based assignment using SP-BC and link speed as weight
# List of Figures

1.1 Economics of cloud computing for a data center [13] ........................................ 3
1.2 System Architecture ......................................................................................... 5

2.1 Concept of an Advanced Traveller Information System [2] ............................. 9

3.1 Impact of Traffic Disturbances [65] ................................................................. 23
3.2 Greater Toronto Highway Network ................................................................. 36
3.3 Percentage increase in average travel time with increasing demand ............ 37
3.4 Average travel time with increasing demand .................................................... 38
3.5 Percentage increase in average travel time with incidents ............................ 38
3.6 Increase in average travel time with incidents ................................................ 39

4.1 Iterations in a DynusT simulation ................................................................. 43
4.2 Adding vehicles mid-simulation in DynusT .................................................... 44
4.3 Loop unrolling example, modified from [44] ................................................. 45
4.4 Critical sections resulting in wasted resources ............................................... 47
4.5 Asymptotic run-time change with more processors ....................................... 48
4.6 Critical sections in a loop in vehicle moving code ......................................... 50
4.7 Layout of the GTHA and GTA networks ....................................................... 52
4.8 DynusT CPU usage for 12 iterations ............................................................. 52
4.9 DynusT Memory usage for 12 iterations ....................................................... 53
4.10 CPU and memory usage during one iteration. Dashed: CPU in %, and solid: memory in GB/100. ................................................................. 54

5.1 Example fish network ...................................................................................... 61
5.2 SP-BC values for Fish network .................................................................. 61
5.3 AK-BC values for Fish network .................................................................. 62
5.4 PK-BC values for Fish network ......................................................... 64
5.5 Parking Lot network ................................................................. 64
5.6 Modified Parking Lot network ..................................................... 64
5.7 Layout of the GTA network ......................................................... 69
5.8 PK-BC metric calculation times using OD- and O-Based SP Algorithms ........................................................................... 70
5.9 Location of the GTA network ....................................................... 71
5.10 Comparison between speeds for the base and incident cases and PK-BC values in the GTA network - each location is represented by a series of upstream and downstream links covering nearby interchanges ........................................ 72
5.11 HCI for the incident case in the GTA network ............................... 73
5.12 Camera snapshot of 401 incident .................................................. 74
5.13 Space-time graph of speed for 401 incident case ............................. 75
5.14 Space-time graph of Speeds and HCI for 401 partial-closure incident case .......................................................... 76

6.1 Traffic simulation and assignment in DynusT [20] (this is a more detailed version of Fig. 4.1) ......................................................... 80
6.2 Method of Isochronal Vehicle Assignment in DynusT [57] .............. 81
6.3 HCI-Based Robust DTA Implementation in DynusT ......................... 83
6.4 Modified Parking Lot network .......................................................... 84
6.5 Max HCI values in MPL network with Incident 1 ............................. 87
6.6 GTA network and Incident Scenario locations .................................. 88
6.7 Space-time graph of speed with UE Assignment with Incident 1 ........ 91
6.8 Space-time graph of speed with HCI-based Assignment with Incident 1 .................................................................................. 91
6.9 GTA major highway locations for Inc1 scenario ................................ 92
6.10 Difference of link travel times between UE and HCI-based assignments in Inc1 case at t=100 min for the GTA network ......................................................... 93
6.11 Difference of link travel times in the Toronto area between UE and HCI-based assignments in Inc1 case at t=100 min for the GTA network ......................................................... 93
Chapter 1

Introduction

1.1 Motivation

In an era of smart phones, smart devices, smart healthcare, smart TVs, smart infrastructure and smart cities, it is important to stop for a moment and ask ourselves: What does “smart” mean? Why is this word being overused and what is its use intended to signify?

From a system’s point of view, one possible definition of a smart system is one that is adaptive, i.e. can adjust to changes in the environment and have optimized performance over time. Therefore, in order for a system to be smart/adaptive, it needs to have the following components (or a possible variation of these components):

- Sensors that help measure the state of the system by capturing data
- A communication system that transfers the acquired data to a central module
- A central module that uses the captured data, forms a global view of the field the system works in, and optimizes the performance of the system
- A communication system that transfers the resulting “decision(s)” from the central module to a controller or group of controllers in the field
- A controller or a group of controllers that perform the new action(s) to achieve optimal performance

One example of an adaptive system is an advanced traffic light at an intersection, which includes:

- Sensors measuring flow of vehicles and monitoring the pedestrian crossing buttons
- A wired/wireless system that sends the sensor data to a central module
• A central module that optimizes the traffic light phases based on sensor data and data from other traffic lights, time of day, etc.

• A wired/wireless system that sends the new traffic light cycle lengths from the central module to the traffic controller

• A traffic controller that updates its phase cycles for optimal performance

Therefore, the need for adaptiveness in a system is due to the variations that occur in the environment, which act as disturbances to the system and degrade its performance. One example of a disturbance is a traffic incident, which might block a lane or more on a road for a period of time, affecting the supply of a roadway network. Adaptiveness in a system entails accurately measuring the state of the system when affected by such disturbances, making intelligent decisions and implementing control actions to re-optimize system performance.

In cloud computing, the effect of disturbances can be resolved by scaling up, i.e. requesting more resources from the cloud infrastructure, to meet increasing application demands or to replace malfunctioning hardware resources (decrease in supply). In smart grids, sudden load increases in one part of the grid can be met by bidirectional power flow from nearby distributed energy resources through intelligent microgrids.

In addition, these adaptive systems have a common strategy that makes them successful in dealing with disturbances: they keep some of their system resources unused during normal operation, in order to maintain a robust system performance and prevent system breakdown when disturbances occur. This is called a “robust resource allocation” strategy. Examples of unused resources include additional capacity for cloud computing and backup storage units for smart grids. The case for cloud computing is shown in Fig. 1.1, where a small portion of unused resources is still kept in the cloud model (on the right) to adjust to possible disturbances affecting the system. This strategy is implemented in such intelligent systems because the costs of a system breakdown, due to the lack of cloud resources, loss of wireless connectivity or having a power outage, are far greater than the cost of backup resources allocated during normal operation for dealing with disturbances.

Things are different for transportation (roadway) networks. Intelligent Transportation Systems (ITS) aim to ensure the efficient utilization of roadway networks by controlling traffic operations and influencing driver behavior. In particular, Advanced Traveller Information Systems (ATIS) aim to provide real-time, travel time and personalized route guidance information to the road users. The core part of an ATIS system is the implementation of a Dynamic Traffic Assignment (DTA) method, which assigns vehicles to
routes on a roadway network, given the road topology and a set of vehicle trip demands. This is similar to the resource allocation method used in cloud computing, wireless networks and smart grids.

However, supply is limited in roadway networks; thus, an extra lane cannot be added on the fly on a highway to alleviate the effects of an incident upstream, at least with today’s technologies. Moreover, vehicles are limited by the topology of the roadway network; thus porting them onto a different road is not possible while they are stuck in traffic, as is the case of migration for virtual machines using cloud computing. Furthermore, although these are assumptions for implementing their DTA methods, ATIS systems today do not have full knowledge of traffic states in the network, vehicles do not necessarily follow the routes based on the resulting assignments, and users do not have full knowledge of routes and route costs in the network. These are significant differences compared to cloud computing, wireless networks, and smart grids, which give ATIS systems far less control over the operations in a transportation network and make it even more important to have robust traffic (resource) assignment strategies to deal with disturbances in a transportation network. This is evident when considering that roadway networks have more regular unexpected disturbances, such as incidents, severely degrading the performance of these networks, in addition to frequent variations in demand and varying traveller route choices to route guidance information. While the economic cost of traffic congestion in the Greater Toronto and Hamilton Area (GTHA) was estimated to be approximately $3.3 billion/year for example [6], almost half of this congestion is caused by non-recurring disturbances, primarily caused by incidents, vehicle breakdowns, road construction activities, special events, extreme weather events, etc. [24, 54]. Toronto, with 1.2 Million vehicle registrations, contributes to 32,000 collisions yearly, i.e. about 88 collisions every day [8],
making traffic disturbances due to weather conditions and incidents part of everyday commute in a city like Toronto. One reason for the failure of ATIS systems to significantly reduce congestion has been a lack of information. However, this has started to change with the emergence of Connected Vehicles and the abundance of real-time traffic data.

1.2 Research Objectives

Advanced Traveler Information Systems (ATIS) aim to provide real-time, traveller and personalized route guidance information to road users. In particular, by continuously monitoring traffic states and executing Dynamic Traffic Assignment (DTA) algorithms, ATIS systems aim to provide car-specific route updates to achieve a particular system traffic strategy. To date, the ability of ATIS systems to considerably reduce traffic congestion has been hindered by the occurrence of traffic disturbances, such as incidents or variations of traffic demand, and the stochastic behavior of travellers.

The growth of Connected Vehicles and the proliferation of smart phones, vehicle sensors, vehicular communication capabilities and participatory sensing, lead to an abundance of high quality data and real-time feedback, with considerable potential for enhancing the ability of ATIS systems to improve the convenience and efficiency of travel. The objective of this research is to investigate the traffic management methods that leverage these sources of rich state information to control the flow of vehicles, in order to achieve system-wide traffic strategies, taking into account the occurrence of traffic disturbances and the stochastic behavior of travellers.

The feedback control loop, describing this objective, is shown in Fig. 1.2, and can be explained as follows: The proliferation of high quality traffic data and real-time feedback from various sources, provides ATIS systems with real-time information about current traffic conditions, including traffic speeds, congestion, hazards, weather conditions, and traveller route choices, etc. Moreover, it enables evaluating the robustness of the transportation network to traffic disturbances and traveller behavior, by analyzing real-time traffic variations and identifying temporal and spatial patterns, which can be used to recognize the potential bottlenecks in the network and predict the onset of congestion. ATIS systems can use this information and analysis to execute a global DTA algorithm to achieve a robust optimal transportation network performance. The resulting traffic assignment can be achieved by shifting traffic flow from saturated to emptier roads using individually-tailored real-time route guidance information provided to travellers. This would be possible through the use of pricing schemes or incentives to influence traveller behavior and promise sustainability. In high congestion cases, where controlling the flow is not enough to satisfy the traffic requests in the network, additional measures, such as dynamically shifting
traffic flow directions or limiting the usage of particular lanes/roads to certain classes of vehicles, e.g. High Occupancy Vehicles, might be required by ATIS systems. Additionally, this analysis would also help provide suggestions to road operators for required road upgrades in the near future to reduce the bottle-necks in the network. The end results will allow travellers to make better decisions, leading to significant reductions of travel time due to the avoidance of congestion, fuel consumption, and greenhouse gases effecting the environment.

Therefore, the objectives of this thesis are:

1. To develop metrics that quantify the robustness of roadway networks to traffic disturbances and identify critical road sections.

2. To develop DTA algorithms based on these metrics to achieve robustness in roadway networks.

### 1.3 Thesis Contributions

In order to deliver on the objectives discussed in the previous section, the contributions of this thesis are as follows:

- In Chapter 3, we introduce a systematic framework for developing a robust design for roadway networks, based on the context of traffic disturbances and design goals, resolving the issue of the many disparate robustness methods developed in the literature.

- We provide a systematic evaluation methodology for robustness metrics in roadway networks.
• In Chapter 4, we present methods to speed up traffic assignment algorithms through compiler optimizations and parallelism, reducing execution times by 50-60 % in various cases, to enable real-time ITS applications.

• We also present the theoretical limits of reducing algorithm execution times by increasing the number of processors used indefinitely and show through extensive experiments that increasing the number of processors beyond a certain point actually increases execution times due to the communication latency between processors.

• We discuss the conversion of the open-source code of the large-scale multithreaded mesoscopic traffic simulator, namely DynusT, to run on Linux, enabling it to be used on more high-performance clusters, and can thus be run using the fastest available processors to enable real-time ITS applications.

• In Chapter 5, we develop a topological robustness metric for roadway networks, by extending the Shortest-Path Betweenness metric, from the field of network science, to apply it in roadway networks. This metric helps transportation planners identify a priority list of important road sections in a road network, guiding their planning initiatives and some of their budget allocation.

• We further augment this metric with dynamic traffic flow metrics, introducing a hybrid robustness metric for roadway networks, which helps dynamically identify critical locations in a road network in real-time, guiding traffic operators to apply dynamic traffic control schemes to alleviate the impact of traffic disturbances.

• In our path to calculate these metrics efficiently in the Greater Toronto Area (GTA), we develop fast K shortest-path calculation algorithms customized for large-scale networks that can be used for various types of networks and to calculate any future K shortest-path based metrics. The developed algorithms, which balance the use of sparse and full matrices to allow fast computation and maintain memory efficiency, are shown to be 5-10X faster than current K shortest-path algorithms for large-scale networks.

• In Chapter 6, we implement a robust dynamic traffic assignment algorithm within the algorithmic components of DynusT, based on the developed hybrid robustness metric, and evaluate its performance compared to existing traffic assignment algorithms. This algorithm improves vehicle travel times with and without the presence of incidents, supporting the case for operators to use such robust traffic assignment methods within real-time ITS strategies to proactively alleviate traffic congestion due to disturbances.
• In Chapters 4, 5 and 6, we perform our experiments and analysis on a large-scale calibrated network, that had used real data to generate the transportation demand and build the network geometry in the GTA. This makes the interpretation and generalization of the results sensible to be used in actual implementations of real-time ITS strategies.

1.4 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 provides a background in Network Science and Dynamic Traffic Assignment. Chapter 3 introduces a systematic framework for developing a robust design for roadway networks, based on the context of traffic disturbances and design goals. Chapter 4 presents methods to speed up traffic assignment algorithms through compiler optimizations and parallelism, to enable real-time ITS applications. Chapter 5 introduces a hybrid approach for measuring robustness in a roadway network. Chapter 6 provides an implementation of a robust dynamic traffic assignment on a large-scale network model for the GTA, based on this hybrid approach. Finally, Chapter 7 provides concluding remarks and possible extensions of our work.
Chapter 2

Background

In this chapter, we provide a background of Advanced Traveller Information Systems (ATISs), including possibly their most important component, the dynamic traffic assignment module. We also describe the mathematical models they are based on, namely graph theory and network science. Finally, we present the basics of traffic assignment, which is the core component of ATISs and review the classical traffic assignment algorithms.

2.1 Advanced Traveller Information Systems

ATISs include a variety of systems that provide real-time, traveller information, aiming to assist travellers with planning, perception, analysis and decision-making, in order to improve the convenience and efficiency of travel [59,75]. Traveller information might include information about current traffic conditions, hazard warnings and/or real-time route guidance recommendations [59]. The aim of ATIS systems is to reduce traffic congestion and ameliorate the capacity of the existing road infrastructure.

A concept of an ATIS system is shown in Fig. 2.1. The inputs to the system include raw data coming from various detection devices, such as road sensors, probe vehicles and participatory sensing from travellers. In addition, inputs include real-time and historical traffic patterns from databases, and various global roadway network information, such as weather conditions, road closures, city events, etc. These inputs make up the content data or events published into an information management layer, which filters and aggregates the collected data, and disseminates it to various specific ITS services and applications. Fig. 2.1 shows four ATIS services/modules that use this data, assessing traffic states and generating outputs, which are published back to the information management layer for dissemination. The provided outputs can be used by various public and private application providers and/or disseminated to
subscribed users and other ITS services through the information management layer.

Our focus in this thesis is on the Dynamic Traffic Assignment (DTA) algorithms, which monitor traffic conditions, and provide route guidance updates to vehicles in a roadway network to achieve a particular system traffic strategy. This will include changing the spatial distribution of traffic patterns, to optimise the usage of the road infrastructure by shifting traffic from saturated to emptier areas, aiming to achieve reductions in travel time, delay and fuel consumption.

In general, road users can be classified into two groups for ATIS systems: static and dynamic users. While static users are non-equipped with communication capabilities or just ignore all ATIS route recommendations, dynamic users are equipped with such capabilities and receive ATIS route recommendations. Therefore, while static users make route choices based on static traffic information, such as the length of the routes or road signs, dynamic users react to route recommendations in a way which optimizes their trips, i.e. they select the routes with the minimal travel times [75]. It is important to note that route recommendations are useful when the road user is able to replan en-route and provide significant

---

Figure 2.1: Concept of an Advanced Traveller Information System [2]
advantages to users with longer trips [26]. For shorter trips or when it is harder to replan en-route, local perception might be inaccurate and lead to worse performance [26]. This is due to the fact that instantaneous travel times (measured in the present, at a road section) might be different than experienced travel times (measured in the near future, when a new road user completes the same road section) [19]. For further explanation of ATIS systems, interested readers can refer to [75], [59], [49], or [43]. The next section presents a background of network science and graph theory, explaining the mathematical models used for transportation networks.

2.2 Network Science

Network Science has recently drawn enormous attention by various researchers in fields, such as physics, biology, communications engineering, and transportation engineering. Network science is the study of complex networks, such as engineered networks, biological networks and social networks. Network science usually deals with graphs with non-trivial patterns, which are structurally complex and are not easily predictable, compared to regular graphs, such as lattices, polygons and molecules. The aim of network science is to study network representations of various phenomena in such complex networks, enabling the development of predictive models for these phenomena, in an attempt to control them. Therefore, network science is sometimes referred to as a combination of graph theory and control theory. In this section, we provide a background of graph theory, which provides the mathematical models for transportation networks. We also present a glossary of terms defining network properties and network models that serve the foundation of understanding complex networks.

2.2.1 Graph Theory

Graph theory is the study of graphs, or equivalently, problems that can be modeled mathematically using vertices and edges. Edges connect vertices together and consecutive edges create paths of flows between vertices. This flow could model fluids in mechanical systems, packets in computer networks, current in electrical systems and vehicles in transportation systems, etc. Therefore, graph theory has been used extensively to model real-life networks in various fields. In practice, the terms networks and graphs are used interchangeably, although networks are graphs with additional properties. Moreover, the terms vertices and edges (used for graphs) and nodes and links (used for networks) are also used interchangeably in practice. A graph $G$ having $|V|$ vertices and $|E|$ edges can be represented with these matrices:
1. Adjacency matrix: A matrix of size $|V| \times |V|$, where the entry $[v_1,v_2]$ is 1 if vertices $v_1$ and $v_2$ are connected by an edge (are adjacent), or 0 otherwise.

2. Incidence matrix: A matrix of size $|V| \times |E|$, where the entry $[v,e]$ is 1 if the edge $e$ has vertex $v$ as one of its endpoints, or 0 otherwise.

The number of vertices and edges in a graph help define the number of cycles in a graph. For example, a complete graph, where every vertex is adjacent to every other, has exactly $|V| \cdot (|V|-1) \choose 2$ edges. On the other hand, a tree, which is a connected graph with no cycles, has exactly $|V|-1$ edges.

A weighted graph is a graph that has weights associated with its edges, usually represented with real numbers. Possible weight definitions include the geometric distance between the endpoints of an edge, the cost of traversing an edge or the benefit gained by traversing an edge. A weighted graph is usually associated with a weight matrix $W$: A matrix of size $|V| \times |V|$, where the entry $[v_1,v_2]$ has the associated weight for the edge(s) between vertices $v_1$ and $v_2$ if they are connected, or 0 otherwise. A network is often defined as a weighted graph.

Graphs can be classified into undirected and directed graphs. In undirected graphs, the edges have no orientation, i.e. edge $[v_1,v_2]$ is the same as edge $[v_2,v_1]$. This means that $W_{v_1 \rightarrow v_2} = W_{v_2 \rightarrow v_1}$ in an undirected weighted graph and the associated weight matrix $W$ is symmetric. Conversely, a directed graph has a set of directed edges $|E|$, where the edge $[v_1,v_2]$ is directed from $v_1$ to $v_2$ and there might not even be an edge $[v_2,v_1]$ from $v_2$ to $v_1$ in $|E|$. The weight matrix $W$ of a directed weighted graph is not necessarily symmetric, as there might be different costs/benefits of traversing the opposite edges $[v_1,v_2]$ and $[v_2,v_1]$, leading to unequal weights $W_{v_1 \rightarrow v_2}$ and $W_{v_2 \rightarrow v_1}$.

Transportation networks are often modeled by Directed Weighted Graphs, where the edges represent roads or sections of roads and the nodes represent intersections between roads or points on roads when the number of lanes changes. In addition, the topology of transportation networks has additional properties, including the number of lanes, the speed limit, and the capacity of each edge, which defines the maximum number of vehicles that can flow through that edge at any time.

### 2.2.2 Network Properties

In this section, we define some terms for network properties commonly used in network science and graph theory. These properties are especially important when discussing the robustness metrics of networks later in this Thesis. For notational purposes, we assume that a network has $|N|$ nodes (vertices) and $|L|$ links (edges) [27]:

- **Density or Connectivity**: This is the ratio of the number of links to the total possible number of links, $D = \frac{2|L|}{|N| \cdot (|N|-1)}$. For planar networks, where the maximum number of possible edges is
3(|N| − 2), D = \frac{L}{3|N|−6}

- **Size**: The size of a network usually refers to either the number of nodes |N| or the number of links |L| in a network.

- **Cyclomatic number**: The cyclomatic number of a network is the number of cycles in that network and is defined as \( \mu = |L| - |N| + 1 \).

- **Average degree**: The average degree is the average number of links the nodes in the network are connected to. It is defined as \( k = \frac{2L}{|N|} \).

- **Average Path Length**: Focusing only on shortest paths in a network, average path length defines the number of steps it takes, on average, to get from one node to another in the network. It is calculated by summing up the shortest paths between all pairs of nodes in the network and dividing by the number of pairs.

- **Diameter of a network**: The diameter of a network \( d \) is the longest length of the shortest path lengths between any pair of its nodes.

- **System spread**: The system spread indicator \( \pi \) is the ratio of the total length of the network, \( M \), to the diameter of the network. \( \pi = \frac{M}{d} \).

- **Average flow per node**: The average flow per node \( \theta \) is the ratio of the total traffic flow in the network, \( T \) (e.g. number of vehicles/hour), to the number of nodes in the network. \( \theta = \frac{T}{|N|} \).

- **Clustering coefficient**: The clustering coefficient (\( C_i \)) of a node \( i \) measures how well its neighboring nodes are connected to each other. It is defined as the ratio of the number of links (\( e_i \)) connecting between pairs of node \( i \)'s neighbors (\( k_i \)) to the total possible number of these links, \( C_i = \frac{2e_i}{k_i(k_i-1)} \). The clustering coefficient of a network is the average of the clustering coefficients of all its nodes.

- **Node Centrality**: Node centrality measures the importance of a node in a network. Common measures of Centrality include:
  - **Closeness**: Closeness measures the average distance that nodes are from all other nodes in the network.
  - **Betweenness**: Betweenness measures the number of shortest paths in the network that include the node.
– *Degree or Strength*: Degree measures the amount of links that a node has in a network. A node will have both an in-degree and an out-degree in a directed network. Analogously, a node’s strength (weight) is the sum of the node’s link weights in a weighted network.

### 2.2.3 Network Models

After understanding the basic properties of networks, we are ready to define special type of complex networks that can be used in comparison to real-world complex networks. Here, we will discuss Random Graphs [32], Small-world networks [77], and Scale-free networks [15].

Random graphs [32] are graphs generated by some random processes. In a random graph, the formation of a link between a pair of nodes has the same probability for all pairs of nodes in the network, defined by a probability distributed function (e.g., Poisson). Based on the law of large numbers, most of the nodes in a random graph are connected to a similar number of edges. This is the case in some transportation networks, such as the US Interstate Highway system. However, most real-life networks are not represented well using random graphs [32].

Small-world networks are networks of highly-clustered small-worlds. While a network may have a large size, it might have a small diameter (the number of links on the longest shortest path between any pair of its nodes, as explained earlier). This means that a small-world network of 1000 nodes might have a diameter of only 5 or 6, as opposed to having 999 links on the shortest path between its two most distant nodes. In comparison, regular networks are not random at all, but are highly clustered; random networks have a high degree of randomness, but are not well clustered; and small-world networks are hybrids of both, where most of their nodes form clusters and few links are connected randomly. Examples of small-world networks include the Internet, networks of brain neurons and many social networks. Mathematically, small-world networks have a high Clustering Coefficient and a small Average Path Length, compared to a random network of the same number of nodes and links. These two properties are important for transportation networks as they measure the accessibility to the transportation network and the length of trips between source-destination (origin-destination) pairs [77].

Scale-free networks are networks where the distribution of links per node follows power laws, rather than being bell-shaped or Poisson. This means that only a small number of nodes in a scale-free network are well connected (have many links), while most of the nodes are connected to a few links. Mathematically, a graph $V$ has a scale-free pattern if the probability $f$, that a node is connected to $m$ links, follows a power law distribution: $f(m) \propto m^{-\epsilon}$. On the one hand, a low scaling factor $\epsilon$, e.g., 1, describes an “aristocratic” network, where a number of nodes are connected to many links. On the
other hand, a high scaling factor, e.g. 5, describes an “egalitarian” network, where only a few nodes have many links. Large scale-free networks usually have a scaling factor in the range \(2 < \epsilon < 3\). Real-life networks tend to become scale-free networks by continually expanding with the addition of new nodes, where these new nodes have a preference to attach to well-connected nodes. Examples of such networks include popular websites, such as Google and Facebook, which have millions of links pointing to them, compared to other websites, which only have a few links pointing to them [15, 27].

2.3 Basics of Traffic Assignment in Transportation Networks

Traffic assignment is the problem of assigning vehicles onto paths in a roadway network from their origins to their desired destinations, given a road topology with network links and a set of vehicle trip demands, represented by an Origin-Destination (O/D) matrix [49]. It is a resource allocation or optimization problem, where the resource or traffic supply is the available infrastructure, such as the roads, and the objective function optimized depends on the traffic assignment algorithm used. Traffic supply is usually limited in a transportation network, unless additional roads or road lanes are added to the roadway network over a period of time. On the other hand, traffic demand, or the desired level of usage, of the roadway network, is a function of the cost incorporated while traversing the roadway network. Examples of this cost include the travel time or fuel consumption during the trip, and traffic demand usually decreases as this cost increases.

Formally, a traffic assignment problem can be modeled in these steps:

1. The transportation network is modeled as a directed weighted graph:

   (a) The links of the graph represent roads or sections of roads

   (b) The nodes \(N\) of the graph represent intersections between road sections or locations where the road properties change (e.g. change in number of lanes or maximum speed).

   (c) The links are directed, as the traffic in one direction may be very different than the traffic in the other direction between 2 nodes.

   (d) The link weights in the graph represent costs of traversing links, e.g. distance, travel time, fuel consumed, etc, or could be based on topological factors, such as the number of lanes, maximum speeds, etc.

   (e) An incidence matrix defines the roads in the network by relating the nodes and the edges together in the network.
2. The traffic demand in this network is represented by an \(|N|\)-by-\(|N|\) O/D matrix, specifying the number of vehicles going between every pair of nodes in the network during different times of the study period, e.g. afternoon peak. In some cases where the network has a large number of nodes, the traffic demand is instead defined by the number of vehicles going between zones, where each zone represents a geographical area that includes a set of nodes (e.g. 10-100 nodes). These zones are then mapped into nodes while solving the traffic assignment problem.

3. An optimization problem is setup having:
   
   (a) An objective function, such as minimizing travel times or delays; or the vulnerability of links, paths or the whole network, etc.

   (b) A traffic flow conservation constraint, which makes sure that every vehicle starting at an origin, ends up at its destination (based on the demand matrix) through the possible roads in the network.

   (c) A capacity constraint, which make sure that the flows on each link of the network do not exceed the link capacities (of the road section).

   (d) An integer constraint, which makes sure that the number of vehicles flowing through different parts of the network is in units of 1, i.e. without being divided, hence an Integer Optimization technique would need to be used to solve such a problem.

4. This optimization needs to be run once or multiple times, depending on whether or not the traffic assignment is static or dynamic, taking into account how link weights are changing (based on current traffic flows for example) and updating the routes for vehicles as required.

   Similar to the traffic demand, the performance of the roadway network is also dependent on the cost, as the performance degrades with an increase in the road usage or the number of vehicles on the roads. For example, drivers on an empty highway will experience free flow travel times (good performance), while drivers on a congested highway (higher volume of vehicles) will experience longer travel times (degraded performance). This can be modeled by using variable costs for links in the transportation network, e.g. through the link performance function, or volume-delay function (VDF), as defined by the Highway Capacity Manual 2000 [36]:

\[
t(V_l) = t_{fl} \left(1 + 0.15 \left(\frac{V_l}{C_l}\right)^4\right)
\]  

(2.1)
where $t(V_l)$ is the average vehicle travel time as a function of demand volume (flow) $V_l$ on link $l$, $t_{fl}$ is the free-flow travel time (without congestion) on link $l$, and $C_l$ is the practical capacity of the link (around 2000veh/hr/lane $\times$ number of lanes for highways). Note that lower capacity values are to be used for arterials and local roads. Using this function, if the demand $V_l$ exceeds $C_l$ by 60% on a link due to rush hour congestion for example, travel time will be double that of free-flow travel time on that link. Finally, $t_{fl}$ is calculated as the ratio of the length of the link to the average vehicle free-flow speed. Therefore, traffic assignment is the routing of vehicles based on performance to achieve a particular objective, e.g. minimize a specific cost function. Comprehensive steps for modelling a transportation network using real data to generate the traffic demand and build the network geometry can be found in [10,41].

A traffic assignment is considered to be static if it only depends on the topology and pre-calculated traffic flow parameters (e.g. speed, density and flow per road section), and it is considered to be dynamic if it explicitly considers time, i.e. dynamic traffic flow parameters in the transportation network. Here are some major assumptions that are used for most static traffic assignment methods in the literature (their validity is discussed in the next chapter):

- All vehicles/users start at the same time.
- Motorists are rational; they try to reduce their own travel cost, if possible.
- Demand is constant; although it varies within the day, day to day and with cost.
- Supply is constant; neglecting the occurrence of incidents or the building of new roads, etc.
- Users have perfect knowledge of routes and costs.

Static Traffic Assignment (STA) pre-assumes the knowledge of the transportation network topology and the exact traffic demand, i.e. where people want to go, and aims to achieve a specific objective, such as minimize total travel time, or minimize total fuel consumption, etc. before the vehicles start their trips. Thus, a STA method assigns trips to all links in the chosen path simultaneously, ignoring the time taken to travel from link to link. From a vehicle’s perspective, a STA provides a route, or a set of consecutive links, from the source of the trip to the desired destination. This is similar to a driver, using a navigation system or a map and writing down a set of roads (or road segments) to take before starting a trip.

An important issue regarding Dynamic Traffic Assignment (DTA) is the vagueness stemming from the generic use of the term “dynamic” assignment in the literature to describe any traffic assignment, which is not static. One such misnomer is the use of DTA to represent the deterministic traffic demand
scenario in the literature where vehicles cannot change their routes based on traffic conditions. This scenario, which assumes full knowledge of Origin/Destination (O/D) trip demands for the entire planning horizon is essentially a time-dependent or time-varying problem (the optimal paths vary with time) as the assignment decisions themselves can be solved a priori [59]. Therefore, from an ATIS perspective, this assumes that vehicles cannot change their routes; thus only need to be given route guidance information at the beginning of their trips; thus it is called an open-loop DTA method in the literature. Compared to STA methods, this method takes into account the interaction between various links while calculating travel costs and accurately represents the effects of traffic congestion, using link queues and reduced speeds.

However, a DTA problem in the literal sense is one in which assignment decisions are made in quasi-real time or real-time (or dynamically) and the traffic assignment decisions change during the planning horizon based on route costs in the network, at least for vehicles that have the capability to change their decisions. Thus, it is called a closed-loop DTA method in the literature, since the system has to use the error between the expected travel times and actual travel times (due to the varying traveller route choices and route costs) and re-adjust its assignment dynamically based on this error. This is essential for providing route guidance for vehicles. Therefore, in addition to knowledge of the topology and the traffic demand, this method assumes knowledge of vehicle locations and traffic flow parameters (such as speeds, densities and flows per road section) at all times. Throughout this work, the term DTA will be used to describe discussions on this field referring to closed-loop DTA.

2.4 Classical Traffic Assignment Methods for Transportation Networks

There are various methods of assigning traffic in transportation networks. Based on the description in the previous section, changing the objective function or any of the constraints leads to a different assignment method. Table 2.1 presents a description of the classical traffic assignment methods in the literature, both static (STA) and dynamic (DTA), as described in Section 2.3. The classical methods for traffic assignment include the methods found in any transportation textbook, such as [49]. They lay the framework for all traffic assignment methods, especially because some of them are still used extensively in practice today. They have been classified into User-controlled and System-controlled in Table 2.1 and are further explained below.

One of the simplest ways of assigning traffic in a transportation network is the All-or-Nothing (AON)
Table 2.1: Classical traffic assignment methods for transportation networks

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimization Principle &amp; Procedure</th>
<th>Type</th>
<th>Control Level</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>All-or-Nothing (AON) Assignment</td>
<td>All traffic between each O/D pair is assigned to shortest path</td>
<td>STA</td>
<td>System</td>
<td>Does not take advantage of alternate routes</td>
</tr>
<tr>
<td>Capacity-Restraint Assignment</td>
<td>Use varying cost with capacity restraint in AON assignment</td>
<td>STA</td>
<td>System</td>
<td>Method is only an approximation for actual link cost</td>
</tr>
<tr>
<td>System Optimum</td>
<td>Minimize total system travel time by equating marginal travel times</td>
<td>STA &amp; DTA</td>
<td>System</td>
<td>Not sustainable without incentives</td>
</tr>
<tr>
<td>User Equilibrium</td>
<td>Minimize user travel time by equating travel times</td>
<td>STA &amp; DTA</td>
<td>User</td>
<td>Performance degradation at high congestion</td>
</tr>
</tbody>
</table>

method [49]. In this method, trips (the traffic demand) between an O/D pair are assigned on the shortest route, which has the minimum travel time or cost between that O/D pair. This is repeated for all O/D pairs in the network until all trips have been assigned. This method does not take into account the existence of alternate routes between an O/D pair, assigning all the traffic to one path, without accounting for congestion effects on that path. This is especially problematic if various shortest paths between O/D pairs share the same link in the network, making it very highly congested. Therefore, this method creates unrealistic flow patterns, only valid in free flow conditions, when there is a very low traffic demand for the transportation network.

Given the short-comings of the AON method, the Capacity-Restraint Assignment was introduced in an attempt to incorporate congestion within the assignment method [49]. This was done by using variable costs for links in the transportation network, e.g. through the link performance function, such as the one defined in Equation 2.1. In the Capacity-Restraint Assignment, several assignments are made using the AON method. However, after each assignment, the travel times are recalculated based on the assigned volume on each link using Equation 2.1 and a new AON assignment is made. This process can be repeated several times, although four iterations are usually adequate to produce acceptable results. It is important to note here that this method is only an approximation for the actual link costs and is thus rarely used in practice today [49]. Moreover, the link performance function used in Equation 2.1 is not valid under hypercongestion cases, since it allows $V_i$ to be higher than $C_i$, which is not possible in practice.

From a system’s point of you, it would be ideal if traffic was assigned in such a way that minimizes the total travel cost in the network. This would align with the “social equilibrium” principle as it is
Chapter 2. Background

19

the best assignment for the society as a whole. This is referred to as the static System Optimum (SO) Assignment method [49] as it uses the transportation system in the most efficient way. The objective function for SO is \( \text{minimize } \sum_l t(V_l) \times V_l \), where \( t(V_l) \) is the travel time on link \( l \) given volume \( V_l \), as explained above and the minimization is done over all links in the network, subject to the constraints discussed in Section 2.3. SO leads to an assignment where all alternate routes between an O/D pair have the same marginal travel time (but not necessarily the same average travel time). Therefore, removing any vehicle from one route and assigning it to another route would increase the overall system travel time; hence SO minimizes the total travel time for all vehicles in the system. However, SO assumes that vehicles follow its resulting assignment (or that vehicle routes are controlled by the system). This makes SO unsustainable or hard to implement in practice, since users of the system are usually not willing to follow a route that has a longer travel time for the sake of achieving social equilibrium or welfare. Users usually opt to take the route with the smallest travel time unless certain incentive or disincentive (e.g. road pricing) methods are used to change that behavior [49].

Unlike SO, which seeks to maximize the social warfare, the static User Equilibrium (UE) Assignment method seeks to maximize each user’s welfare, by minimizing individual travel times of all users in the system. This is done through an assignment where all alternate routes between an O/D pair have the same travel time. As no user can reduce his or her travel time by choosing another route, UE is an equilibrium assignment. The objective function for UE is \( \text{minimize } \sum_l \int_0^{V_l} t_l(x) \, dx \), where \( \int_0^{V_l} t_l(x) \, dx \) is the area under the volume-delay curve given volume \( V_l \) on link \( l \) and the minimization is done over all links in the network, subject to the constraints discussed in Section 2.3. One limitation of UE is highlighted using Braess’ paradox, which shows that adding a highway can sometimes degrade the performance of a transportation network [34], when traffic is assigned by UE (a selfish technique), rather than a SO assignment (an unselfish technique). Another limitation is the fact that network performance (in terms of total travel time) in a UE assignment can be twice as bad as SO assignment under high congestion [49].

The DTA methods of UE and SO try to achieve the same objectives as their STA counterparts, but additionally taking into account the time taken to travel along each link during the assignment. Therefore, for a DTA UE assignment, all vehicles departing at the same time on routes between an O/D pair will have the same travel time and for a dynamic SO assignment, all routes between an O/D pair during the same period will have the same marginal travel time. Another important difference with STA methods is due to their usage of the VDF, defined in Equation 2.1. This allows the volume on a link to increase indefinitely and exceed the physical capacity of the link (in vehicles per hour), represented by a volume-to-capacity (V/C) ratio > 1, which does not have intuitive traffic meaning. Conversely, a
Chapter 2. Background

DTA ensures that this doesn’t happen through explicit modeling of traffic flow dynamics: by matching the increase of the link density on a congested link by a decrease in speed in the link based on the fundamental speed-density relationship [19]. Furthermore, VDFs in a STA method prevent overtaking because they assume link FIFO, do not distinguish between different lanes on a road section, and cannot represent the phenomenon of congestion spillback between links [19]. For both UE and SO, DTA method is more realistic, but much more complex and computationally intensive [40]. Dynamic traffic simulators implement dynamic UE since it best models traffic behavior of users on real transportation networks. They also relax some of the other assumptions listed above, such as varying supply with incidents, specifying sets of vehicles that have knowledge of routes and costs, varying demand, etc., to model different traffic scenarios.

A comprehensive review of various mathematical formulations for UE and SO assignment methods can be found in [59]. In his PhD thesis, Srinivas considers the situation where some vehicles are not equipped with communication capabilities; thus, do not receive routing information from a centralized controller. Therefore, the dynamic UE and SO problems are modeled in such a way that only a percentage of the users receive updated traveller guidance. Results show that in the case where 75% of the vehicles are not equipped, thus do not follow system recommendations, vehicles following a UE assignment have slightly shorter travel times that vehicles following an SO assignment, which explains the rationale behind user behavior in a UE assignment when most other users are not following or cannot receive travel guidance [59].

In summary, classical traffic assignment methods, such as UE and SO rely on the following assumptions, which are unrealistic to date:

1. The assumptions that the system has full knowledge of traffic states in the network and vehicles follow the routes based on the resulting assignment.

2. The assumption that users have full (global) knowledge of routes and route costs in the network.

These assumptions lead to performance degradation for ATIS systems based on such DTA methods, when met with unexpected traffic disturbances, such as incidents or variations in demand, partial user knowledge or varying traveller choices (which make the assumptions invalid). The next Chapter discusses how to design robust roadway networks that reduce the requirement for the system to have full knowledge of traffic states and traveller route choices by leading to a design, which is relatively robust (insensitive) to variations in these, given the partial knowledge that it has.
Chapter 3

Robust Network Design for Roadway Networks

This chapter provides a systematic framework for developing a robust design for roadway networks, based on the context of traffic disturbances and design goals. Moreover, it presents examples of robust network designs from the literature and illustrates their potential benefits through a simple case study.

3.1 The Need for a Systematic Framework for Robust Network Design

Recently, significant attention has been devoted to designing robust real-world networks that can handle the possible impacts caused by disturbances, such as system malfunctions, weather conditions, cyber attacks, terrorist activities, unexpected demands, incidents, etc. [45, 62, 78]. Examples of networks that can be affected by these disturbances include electric grids, computer networks, communication networks, financial networks, safe water networks, and transportation networks. In order to assess the robustness of a network, it is crucial to quantify the importance of its nodes and links to the performance of the overall network under such disturbances [65]. For example, the New York JFK airport is a very important node in the US air traffic network, because it acts as a hub between hundreds of other smaller airports. Therefore, any disturbance to the operations of the JFK airport could have considerable impact on the overall US air network performance.

In the context of roadway networks, several studies have looked at the general problem of designing roadway networks [55, 81]. Such studies relied on optimization models to design a network or make an
optimal investment decision in order to satisfy the route choice specifications of network users, while minimizing total travel cost [55, 81]. The past few years in particular have seen a substantial effort in developing methods for measuring the robustness of roadway networks [27, 37, 47, 50, 56]. However, there has been a variety of approaches to network robustness addressing different types of traffic disturbances, resulting in disparate methods for robust network designs and their associated optimization problems. For example, Ip et al. [38] solved a resource allocation problem for choosing long-term road building projects by maximizing the robustness of a roadway network to severe traffic conditions. Koulakezian et al. [47] solved a traffic assignment problem for choosing short-term traffic routing by maximizing the robustness of a roadway network to frequent traffic demand/supply variations.

Therefore, there is a need for a systematic framework enabling the research community and transportation planners/operators to understand and apply different robust network designs, based on the context of traffic disturbances (severity and frequency) and design goals (short-term and long-term). This chapter introduces such a framework, starting with the various contexts of robustness based on the types of traffic disturbances in Section 3.2. Section 3.3 discusses the goals and requirements for developing a robust network design (short-term and long-term), with recent examples from the literature. Section 3.4 presents a sample study using traffic assignment to achieve short-term robust network design with frequent disturbances. Finally, Section 3.5 presents the conclusion.

### 3.2 Context of Robustness

In this section, we define the sensitivity of roadway networks to traffic disturbances and the possible contexts for robustness according to the severity and frequency of disturbances. We then define robustness and discuss how it can be measured.

#### 3.2.1 Sensitivity of Roadway Networks

The impact of traffic disturbances on networks is illustrated in Fig. 3.1. Traffic disturbances cause various changes in the network, which may subsequently cause changes in some performance measures of the network.

Based on these definitions, the sensitivity of a roadway network to a disturbance [64] is defined as the change in a performance measure, divided by the magnitude of the change in the network due to the traffic disturbance, as shown in Equation 3.1.
Figure 3.1: Impact of Traffic Disturbances [65]

\[
\text{Sensitivity} = \frac{\text{Change in performance measure}}{\text{Change in network}}
\]  

For instance, an incident on a highway might cause changes in the network, including taking away 30% of the highway capacity (and possibly altering the demand on the highway and on alternate roads, etc.) Therefore, the highway could be considered insensitive to incidents of this nature if this capacity drop only causes a moderate increase in travel times (a chosen performance measure of interest), and considered sensitive if it leads to a large increase in travel times.

### 3.2.2 Severity of Traffic Disturbances

The severity of traffic disturbances defines the extent of network changes. For example, disturbances can lead to:

- Normal traffic conditions, with minor variations in demand, minor incidents and changing weather conditions.

- Severe traffic conditions, including major incidents, natural disasters, and severe weather conditions.

In normal traffic conditions, the road topology is usually left intact, with certain performance degradation. However, severe traffic conditions include cases when demand tremendously exceeds capacity in parts of the network or when connectivity of the roadway network is greatly impacted, such as the case of evacuation scenarios. Severe traffic conditions typically lead to severe changes in performance measures also; thus certain special performance measures, such as the time required to evacuate a network, may be of interest in such cases.
3.2.3 Frequency of Traffic Disturbances

The frequency of traffic disturbances affecting the network is another factor to consider. For example, traffic disturbances can be classified as frequent and infrequent [65]. Traffic operators consider another associated notion of disturbances, namely their predictability of occurrence, defining them as predictable, if it is relatively easy to predict them (such as a snow storm), and unpredictable otherwise. Given these classifications of disturbances, a possible set of examples for traffic disturbances includes:

- Frequent predictable: A pedestrian at a crossing near a stadium before a sports game. This is predictable as the game is scheduled in advance.

- Frequent unpredictable: A pedestrian at the same crossing next to the stadium on a cold Sunday morning. This is not predictable as there is no game scheduled on Sunday morning and many people avoid walking at very low temperatures, yet this still occurs frequently.

- Infrequent predictable: Incidents during a storm, if the storm was predicted

- Infrequent unpredictable: An earthquake damaging roads

Therefore, considering the severity, frequency and predictability of traffic disturbances helps better understand the impacts of disturbances on network performance.

3.2.4 Robustness of Roadway Networks

The robustness of a roadway network can thus be defined as the ability to maintain acceptable performance under a set of traffic disturbances [76]. Acceptable performance refers to reasonable travel times, delays, energy consumption, etc. compared to disturbance-free or average typical traffic conditions [64]. A formal definition of roadway network robustness should consider the context of robustness based on the severity, frequency and predictability of traffic disturbances. As illustrated in Fig. 3.1, each disturbance within this context leads to a corresponding set of network changes $n_i$. Therefore, based on Equation 3.2, a roadway network is robust if it maintains acceptable performance and the maximum change in a key performance measure of interest, divided by the magnitude of the change in the network $n_i$, over all possible network changes due to disturbances within its context, is small:

$$\max_{n_i} \left| \frac{\text{Change in performance measure}}{\text{Change in network}(n_i)} \right| \text{ is small}$$

(3.2)

This definition is independent of the context of disturbances. However, the magnitude of network and performance changes varies with this context. Therefore, to study the network robustness under
emergency evacuation cases for example, the context of robustness is the set of severe, infrequent, and unpredictable traffic disturbances [38]. On the other hand, the context of traffic demand variations during a sports game is the set of frequent predictable traffic disturbances in normal traffic conditions.

Note that even after considering the context of disturbances that could affect a roadway network and choosing a performance measure such as travel time in the network, calculating Equation 3.2 for all possible network changes, caused by all possible disturbances within this context, is usually not computationally feasible. Therefore, robustness metrics have been proposed to varying degrees of success in capturing the essence of Equation 3.2. Metrics have achieved this by modeling key changes in the network and/or performance measures (shown in Fig. 3.1) due to a set of disturbances within the context assumed for a network, rather than modelling all possible network and performance changes caused by disturbances.

For example, assume that average node degree (number of neighbors a node is connected to) is a robustness metric for a network under the context of severe conditions. Based on this metric, a network is robust if the change of its average node connectivity under this possible set of disturbances is small (see Equation 3.2), i.e. below a certain threshold chosen by the road planner/operator. However, since this metric only captures a change in the network (without taking into account any changes in performance measures), it offers an incomplete or partial view of network robustness. Similarly, other metrics capture network robustness with varying degrees of accuracy.

In addition, robustness metrics can be further classified as:

- Static, if they incorporate network changes and/or static performance measures that do not change with time, e.g. free-flow travel times, in the modelling process.

- Dynamic, if they incorporate both network changes and explicitly model temporal variations, such as travel times, queuing, spread of congestion and dynamic network and performance changes in general.

Static metrics have low computational complexity compared to their dynamic counterparts, but lack in the ability to capture short-term performance changes due to traffic disturbances. The next section will discuss the various objectives of robust network design, shedding light on the roles of static and dynamic metrics in various network design problems.
3.3 Robust Network Design

Roadway network designs can have both long-term and short-term objectives. While long-term objectives typically address regional planning needs, usually performed by transportation planners, short-term design objectives address managing the day-to-day operations of a roadway network, usually performed by traffic engineers and system operators. Recently, analyzing the robustness of roadway networks to traffic disturbances has played an influential role in roadway network design, including the decision of where to invest, aiming to make roadway networks more robust. This section discusses both long-term and short-term robust network designs, highlighting the benefits of analyzing network robustness in each case and providing insights on the robustness metrics proposed for these designs.

3.3.1 Long-term Robust Network Design

Long-term network design implies designing a region-wide roadway network or updating an existing one to accommodate normal traffic disturbances, due to varying traffic demand, occurrence of incidents, and changing living and employment locations. This requires forecasting population growth, land use and traffic demand [35]. In addition, it could include planning for disaster situations with severe traffic disturbances, which could require emergency evacuation due to an earthquake, nuclear spill, etc. [9]. Defining robustness metrics is key to guide these long-term planning goals with explicit optimization of network robustness. Examples of robust long-term design goals include:

- Making policy decisions to prioritize building new roads to alleviate the performance degradation of critical roads
- Optimizing emergency evacuation plans to manage the usage of critical parts of the network
- Changing directions of existing roads or particular lanes of roads to limit the effect of seasonal traffic demand variations on critical roads

To help planners achieve long-term robust network design goals, a robustness metric should satisfy the following criteria:

- Network-wide measure: The metric should quantify the impact of traffic disturbances on the network as a whole.
- Component-specific measure: should identify the critical links/nodes for a robust system performance [65].
Table 3.1: Robustness metrics for long-term roadway network design

<table>
<thead>
<tr>
<th>Metric</th>
<th>Source</th>
<th>Solution</th>
<th>Network-wide measure</th>
<th>Component-specific measure</th>
<th>Directed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilience</td>
<td>38</td>
<td>Average of node resiliences</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Friability</td>
<td>38</td>
<td>Decrease in Network Resilience</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Diameter Change</td>
<td>63</td>
<td>Min. Net. Diameter increase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternative Paths</td>
<td>27</td>
<td>Based on No. of alternate paths</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fastest-Path Betweenness</td>
<td>30</td>
<td>Weight: No. of Restaurants</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Estimating Travel Time</td>
<td>37</td>
<td>Shortest path assignment</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Vehicle Loss Hours</td>
<td>65</td>
<td>Estimating partial link blocking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Directed: should be capable of modeling the asymmetric demand inherent in roadway networks, as they are represented by directed graphs (some existing metrics for instance assume symmetrical links between nodes, which is not the case in transportation networks).

In Table 3.1 we provide a set of proposed robustness metrics for long-term robust network design, and summarize these metrics against the requirements defined above. In this section, we summarize the robust metrics proposed in the literature (with their original metric names) by identifying the context of disturbances considered and the associated design goals, in light of the framework setup in Section 3.2 and Section 3.3.

Resilience and Friability [38] are static robustness metrics in the context of severe, infrequent and unpredictable disturbances, with the goal of guiding policy decisions on where to invest in expanding the roadway network. In a network that models cities as nodes and passageways between cities as paths/routes, the resilience of a node is defined as the weighted average of the number of passageways it has to all other nodes using city population as a node weight. Consequently, network resilience is calculated as the weighted sum of node resiliences. Moreover, friability is defined as the reduction in network resilience upon removing a node or a link from the network. Based on these robustness metrics, a robust network design problem is formulated to prioritize the investment on road expansion project(s) - among several possible projects - to maximize the resilience and minimize the friability of the network subject to budget constraints [38]. The definitions of resilience and friability metrics do not fully capture network robustness as defined in Equation 3.2. For instance, they do not consider the capacities of links, their importance to the whole network, or their impact on performance measures such as travel time or throughput. In addition, friability needs to be computed for every network node after removing it and re-computing the resiliences of its neighboring nodes, which is computationally inefficient. While the aim of prioritizing road expansion investment to improve resilience and friability aligns with transportation planner goals, an additional requirement to convince policy makers to secure funding for these roads...
requires clarifying the potential savings this additional achieved robustness provides relative to a base case of not building new roads for example. For instance, this could possibly be in terms of time required to evacuate a network.

**Diameter change** [63] is a static metric in the context of severe, infrequent unpredictable disturbances for bus traffic networks. A network is considered to be robust if the increase in the network diameter (the largest number of links that need to be traversed between any pair of nodes) upon the removal of nodes, due to disturbances, is small. This metric is dependent on few low-degree but high-load nodes in the network, a property that defines the topology of bus networks [63]. Therefore, it could be used to promote bus network designs with highly-connected stations. However, since roadway networks have more complex routing options and design parameters compared to a fixed pre-defined set of routes for bus networks, this metric does not well represent roadway network robustness.

Derrible [27] presents a similar static metric called **Alternative Paths** for metro networks. This metric is calculated as $r^T = \frac{\mu - |L^m|}{|N|}$, where $\mu$ is the number of cycles in a network with $|N|$ nodes and $|L^m|$ multiple links (between nodes, used to provide redundancy). This metric works well for the topology of metro networks, but seems implausible for urban roadway networks, which have different network topologies.

Leung et al. [50] propose the static **Node-Weighted Fastest-Path Betweenness** metric in the context of frequent disturbances within normal traffic conditions. The betweenness of a node k with respect to flows from source node s to destination node d is defined as the proportion of the shortest paths from s to d that traverse node k. The overall betweenness of node k is the sum of the betweenness values over all source-destination (s-d) pairs [70]. Node-weighted fastest-path betweenness uses the free-flow travel times of each link based on road class and length to determine the shortest paths in the betweenness calculation and also considers the importance of a node by using the number of restaurants in its vicinity as a node weight. To show the potential benefits of this metric, correlations with traffic conditions were performed in [50]. At first glance, this metric looks promising for promoting long-term designs to handle frequent disturbances; however, it uses an odd node weight and assumes free-flow travel times in its calculation. These assumptions are questionable for common traffic flow patterns in real roadway networks, as shown by the average correlation results with traffic conditions provided in [50].

Another metric proposed in the context of frequent disturbances within normal conditions is the **estimating travel time heuristic** [37]. The objective of this metric is to estimate the impact of a link failure on the total system travel time and design the network to reduce this impact. For this purpose, it does not require removing each link and re-running a traffic assignment algorithm. Instead, it requires running an assignment algorithm once to find the initial assignment, then fixing link costs based on this
assignment and calculating link failure effects on total travel time by only rerouting the affected traffic using shortest path routes. However, given the dynamics of roadway networks and the considerable effect of traffic flowing from shortest-path routes into alternate routes, this metric cannot sufficiently model the dynamic movement of vehicles in the presence of frequent disturbances in the network.

Vehicle loss hours [65] is a similar static metric for frequent incidents within normal traffic conditions. It is evaluated with travel times from a macroscopic traffic assignment tool and a marginal incident computational model based on probabilities and properties of incidents on links. It estimates the effect of drivers using alternate routes using percentages of rerouting upstream of an incident. Performance results with this metric [65] show that it is unable to model spillbacks caused by incidents. Such a robustness metric can possibly provide a modeling capability to design roadway networks with reduced impact from incidents. However, this capability is limited by its macroscopic model that simplifies driver behavior and by its inability to model spill-back effects to accurately represent the propagation of shockwaves upstream of an incident.

### 3.3.2 Short-term Robust Network Design

Traffic engineers are responsible for the day-to-day operations of the roadway network and the associated short-term network designs. In most cases, traffic engineers consider the context of frequent and infrequent disturbances under normal traffic conditions. However, severe disturbances might be of interest in the case of preparing a real-time dynamic emergency evacuation plan that can cope with short-term traffic disturbances. Short-term network design requires managing both the traffic demand and the network supply in a roadway network. Supply-side management strategies include, for instance, ramp metering [58], Variable Message Signs (VMS), Variable Speed Limits (VSL) [82], and changing road directions dynamically. Demand-side management includes Advanced Travel Information Systems (ATIS) [75] and congestion pricing [23].

Analyzing network robustness metrics helps in directing these operation goals to achieve a robust roadway network in day-day operations in the presence of disturbances. It helps to pro-actively control traffic demand and supply, considering the possible effect of disturbances that might occur, by optimizing for robustness rather than minimizing travel times during daily operations [51]. In other words, the impact of disturbances can be mitigated, leading to more stable roadway networks, by incorporating robustness metrics into the objective functions of real-time demand management and supply optimization, such as ramp metering, congestion pricing, VSL, VMS and dynamic road direction change. To guide such short-term robust network design objectives, a robustness metric should satisfy the following
Table 3.2: Robustness Metrics for Short-Term roadway network Design

<table>
<thead>
<tr>
<th>Metric</th>
<th>Source</th>
<th>Solution</th>
<th>Network-wide measure</th>
<th>Component-specific measure</th>
<th>Directed</th>
<th>Scalable</th>
<th>Dynamic</th>
<th>Measurable in Real-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability</td>
<td>[45]</td>
<td>Link flow over Available Capacity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UNPM</td>
<td>[56]</td>
<td>Mean of s-d pair throughputs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UNPM Robustness</td>
<td>[56]</td>
<td>Variation with capacity decrease</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Network Criticality</td>
<td>[47]</td>
<td>Weight over Betweenness</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

requirements (in addition to the requirements of long-term design discussed earlier):

- Scalable: should have low computational complexity relative to network size.
- Dynamic: Should capture both dynamic changes in traffic performance measures and effects of network changes as shown in Fig. 3.1.
- Measurable in Real-time: Should be easily measured using real-life traffic monitors, such as road sensors and sensors in connected vehicles.

A set of robustness metrics proposed for short-term robust network design is shown in Table 3.2. Here, we describe and compare them based on the established requirements.

**Vulnerability** [45] is a dynamic robustness metric proposed in the context of frequent incidents within normal traffic conditions. Defined as the ratio of actual flow (demand) to available capacity of links, it is calculated for network links using an initial traffic assignment and its variation is studied with a reduction of the number of lanes for links in the network. The results in [45] show that link-based metrics, including this metric, are insufficient to capture the effects of blocking a link (for example due to an incident or construction) on the whole network [45]. Using loop detectors on the roads, the flow on the links can be estimated and therefore this metric is scalable and can guide a response towards achieving robustness by possibly maximizing the weighted average for flow to available capacity ratios in network links. It is important to note that this metric assumes that when demand exceeds capacity, the network (or link) will operate at capacity. Although this is the case in many networks types such as computer networks, it is not the case in roadway networks. In roadway networks if demand exceeds capacity, throughput will degrade due to excessive turbulence in the traffic stream [49]. This could lead to measuring low flow values not because there is unused capacity but because the traffic stream is congested and not moving well. As a result, this metric is not suitable for congested networks.

The **Unified Network Performance Measure** (UNPM) [56] is a dynamic metric proposed in the context of frequent disturbances in normal traffic conditions. Given a network topology $G$ and the
equilibrium demand vector \( D \) between its source-destination (s-d) pairs, UNPM is defined as:

\[
\epsilon = \epsilon(G, D) = \frac{\sum_{s,d} D_{sd} \lambda_{sd}}{n_{sd}}
\]  

(3.3)

where \( n_{sd} \) is the number of s-d pairs in the network, and \( D_{sd} \) and \( \lambda_{sd} \) are the equilibrium demand (numbers of cars) and the equilibrium disutility for s-d pair \( sd \), respectively [56]. This is called a measure of efficiency as it computes the average demand to cost ratio for all s-d pairs, where cost is the total travel time for vehicles on an s-d pair. Therefore, this ratio is the service rate of vehicles (and the inverse of average vehicle travel time) for each s-d pair and \( \epsilon \) is an average of this ratio over all s-d pairs in the network [56]. This metric is suitable for directed networks and can guide a response maximizing the UNPM measure by rerouting traffic for s-d pairs with high \( \lambda_{sd} \) to other routes. However, the demand vector it needs cannot be calculated from real-time road and vehicle sensors, which provide traffic data in terms of flows for example, without identifying which s-d pairs they belong to. Consequently, this measure provides an offline analysis tool of robustness through a simulator, assuming full knowledge of traffic demands between s-d pairs.

Nagurney et al. [56] present another metric called UNPM Robustness based on Equation 3.3. Given a vector of link capacities \( C \), this metric \( (R^\alpha) \) is defined as the relative performance retained when the vector of link capacities for all network links is reduced to \( \alpha \cdot C \) with the scalar factor \( \alpha \in (0, 1] \):

\[
R^\alpha = \frac{\epsilon^\alpha}{\epsilon} \times 100\%
\]

where \( \epsilon^\alpha \) and \( \epsilon \) are the network performance measures calculated by Equation 3.3 with the original capacities and remaining capacities, respectively. However, this scalar reduction in link capacities should not be modelled for all network links at the same time as incidents affect specific links more than others in the network, occur at different links at different times and reduce various link capacities differently depending on the severity of the incident. Therefore, this metric is not suitable for robust short-term network design.

Lastly, network criticality [47] is a dynamic metric proposed in the context of frequent disturbances within normal traffic conditions. Assume that a random-walker starts from a source node \( s \) in the network, then chooses a neighbor at random with equal probability and goes there. It continues wandering around until it reaches a specified destination \( d \), where it stops. Thus, similar to the notion of fastest-path betweenness explained earlier, the random-walk betweenness of a node \( k \) with respect to flows from \( s \) to \( d \) is the proportion of the random walks from \( s \) to \( d \) that traverse node \( k \). The overall betweenness of node \( k \) is the sum of this quantity over all s-d pairs. Therefore, the point-to-point network criticality of node \( k \) for trajectories from \( s \) to \( d \) is [70]:
\[ \tau_{k}^{sd} = \frac{b_{sk}(d) + b_{dk}(s)}{W_k} \quad (3.4) \]

where \( b_{sk}(d) \) is the random-walk betweenness of a node \( k \) from \( s \) to \( d \), \( b_{dk}(s) \) is the random-walk betweenness of a node \( k \) from \( d \) to \( s \), the inverse of travel time is used as the link weight \( w_l \) for link \( l \) (taking into account that road links with long travel times are undesirable), node weight is defined as 
\[ W_k = \sum_{l \in A^o(k)} w_l, \]\nand \( A^o(k) \) denotes the set of outgoing links attached to node \( k \).

In generic random-walks, in which the probability of transitioning along a link is proportional to the weight of the link, \( \tau_{sd} \) is independent of node \( k \) [70]. Consequently, the average network criticality \( \tau \) (of the whole network) is defined as the mean of all point-to-point network criticalities and it can be shown to be proportional to the trace of \( L^+ = [L^+_{ij}] \), the Laplacian matrix of the graph [70]:

\[ \tau = \frac{1}{n(n-1)} \sum_{s,d} \tau_{sd} = \frac{2}{n-1} \text{Tr}(L^+) \quad (3.5) \]

This metric supports short-term robust design as it captures variations in traffic demand or supply using real-time measurements with graph weights, where optimizing for network criticality minimizes steep increases in average link travel times by guiding short-term responses for rerouting away from critical links (with low weights). Note that network criticality is inherently for undirected graphs. Therefore, to be used for directed roadway networks, an undirected symmetric matrix of the graph defined as \( W_{sym} = \frac{W + W^T}{2} \) is used, where \( W \) denotes the link weight matrix and \( W^T \) denotes the transpose of \( W \) [70]. Thus, network criticality has a limitation as it approximates directed roadway networks by undirected graphs. However, this could be resolved by developing a similar robustness metric for roadway networks by adapting the directed version of network criticality defined in [72] for usage in roadway networks.

### 3.3.3 Robust Network Design Discussion

After analyzing robustness metrics proposed for long-term and short-term network design, we can see that they capture the essence of Equation 3.2 with varying degrees of success.

Robustness metrics proposed for long-term network design have mostly been static, with some metrics using estimation techniques by modeling some aspects of traffic flow variation. For this reason, they mostly fail in modeling changes in performance measures as needed by Equation 3.2. Metrics such as...
resilience or friability can help transportation planners make policy decisions to build new roads while reducing the impact of unpredictable disturbances. However, for developing long-term robust network designs in the presence of regular disturbances, further analysis is required to clearly model the impacts of such disturbances and determine the critical components of a network. This requires using dynamic metrics showing the impact of disturbances on various traffic flow measures and conditions. In addition to long term planning, the network can benefit from short-term robust design techniques, such as, for instance, robustness-maximizing dynamic traffic assignment, presented in the next section. Operating the network with robust traffic assignment will not only enhance the daily operation but may also reduce the need for more expensive long term infrastructure expansion.

Robustness metrics for short-term network design are dynamic and primarily deal with frequent disturbances in normal traffic conditions. The most promising metrics are the UNPM and network criticality metrics as they can help detect the onset of congestion due to disturbances and guide a short-term response to re-optimize network robustness. Nevertheless, since these metrics have a high computational complexity, for short-term robust network design, novel methods should use distributed and high-performance computing strategies to reduce the computation times for these metrics, yet still achieve a high degree of calculation accuracy. Moreover, the practical usage of the UNPM measure requires the estimation of the demand vector between s-d pairs in the network based on real-time traffic measurements captured in the field.

3.4 Sample Study

This section presents a sample study for a short-term robust network design for a traffic assignment problem using robustness metrics. The goal is to show benefits of robust network design and introduce a systematic evaluation methodology for robustness metrics. We first formulate the traffic assignment problem and then present the performance results.

3.4.1 Traffic Assignment Problem Formulation

This section provides the problem formulation for a static traffic assignment [19], including with the system model used and the traffic optimization problem formulation.

System Model:

Suppose that the roadway network topology is given by a directed graph $G(N,E,W)$, where $N$, $E$, and $W$ denote the node set, link set, and link weight matrix, respectively. While a link represents a
road segment between nodes $i$ and $j$ and denoted by $l = (i, j)$ with weight $w_l$, a node represents a trip origin/destination/junction of road segments. The sets of outgoing links and incoming links of a node $k$ are denoted by $A^o(k)$ and $A^i(k)$, respectively. We assume that the analysis period of interest is taken as a peak period with relatively high demand and vehicles request, receive and follow guidance information from the network operator. From the operator’s perspective, these requests make up the triple $(s, d, \gamma_s(d))$, where $s$, $d$, and $\gamma_s(d)$ denote the traffic source, destination and the demand from $s$ to $d$, respectively, for each $s$-$d$ pair. Additionally, travel times are calculated using the volume-delay function (VDF) described earlier in Equation 2.1.

**Traffic Problem Formulation:**

The objective is to find the assignment of vehicles on each path and link during the period of interest. We can formulate the optimization problem for a network minimizing total travel time, called System Optimal (SO) and described in Section 2.4, subject to traffic flow conservation constraints [19].

$$\begin{align*}
\text{minimize} & \quad \sum_{l \in E} t(V_l) \times V_l \\
\text{subject to} & \\
\forall s, d \in N, \forall l, e \in E, \forall k \in N & \\
\sum_{l \in A^o(k)} V_{l}^{sd} - \sum_{e \in A^i(k)} V_{e}^{sd} &= \gamma_s(d)\delta(k - s) - \gamma_s(d)\delta(k - d)
\end{align*}$$

The objective function in problem (3.6) is the total travel time for all vehicles in the network and is based on both the flows $V_1$ and the individual travel times $t(V_1)$ on each link in the network. The flow conservation constraint needs to be satisfied for every node $k$ and entry $\gamma_s(d)$ of the traffic matrix, where $V_{l}^{sd}$ is the flow of link $l$ for traffic from source $s$ to destination $d$ and $\delta(x)$ is the Kronecker delta function. Under very light traffic conditions, SO would assign all traffic to the shortest paths (also called an all-or-nothing assignment). As demand levels increase relative to capacities, SO assigns traffic to routes with lower marginal travel times, minimizing the total travel time in the roadway network [49].

We also define the traffic optimization problem for the User Equilibrium (UE) Assignment, also described in Section 2.4, which seeks to maximize user welfare by minimizing individual travel times of all users in the system [49]. Therefore, the objective function in (3.6) is replaced with $\text{minimize} \quad \sum_{l} \int_0^V t_l(x) \, dx$,
where \( \int_0^{V_l} t_l(x) \, dx \) is the area under the volume-delay curve defined by Equation 2.1, given volume \( V_l \) on link \( l \) and the minimization is done over all network links [49].

The traffic optimization problem minimizing network criticality (Tau) replaces the objective function in (3.6) with: minimize \( \tau \) as defined in Equation 3.5, where link weight \( w_l = \frac{1}{t_l(V_l)} \). To calculate this metric, an undirected symmetric matrix of the graph defined as \( W_{sym} = \frac{1}{2} (W + W^T) \) is used, where \( W \) denotes the link weight matrix and \( W^T \) denotes the transpose of \( W \) [70]. This optimization distributes traffic flows such that it minimizes the use of critical links, making the network only incur minimal increases in link travel times subject to traffic disturbances (as needed by Equation 3.2). In addition, this optimization problem remains convex (since the weight defined satisfies the condition that an increase in the weight also increases the desirability of using the link), and thus can be solved efficiently using a similar solution procedure to that of SO, reaching a unique solution [70].

Lastly, we define the traffic optimization problem maximizing the UNPM measure by replacing the objective function in (3.6) with maximize UNPM, as defined in Equation 3.3. This problem maximizes the network robustness by maximizing the traffic demand-to-cost ratio for all s-d pairs in the network, in order to get the highest network efficiency possible. The convexity of this optimization problem and the uniqueness of its solution are also guaranteed given the monotone link weight function used, using a similar solution procedure to that of SO [56]. The robustness of the assignment algorithms defined here is discussed next.

### 3.4.2 Performance Results

The simulation test network includes the major highways in the metropolitan Toronto area, as shown in Fig. 3.2. Note that there are 2 links in opposite directions between every pair of nodes in the network (shown with bidirectional links for simplicity) and the link weights shown represent the number of lanes in each direction. The traffic demand includes equal traffic from 6 s-d pairs, producing various levels of congestion throughout the network. We are interested in analyzing the robustness of the solutions of the 4 static traffic assignment algorithms defined in subsection 3.4.1 (we are implementing these algorithms with a dynamic traffic assignment simulator). Therefore, we first run these traffic optimization problems and find the assignments for SO, UE, Tau and UNPM based on the initial traffic demand and supply. Next, we model frequent traffic disturbances in terms of increases in demand and decreases in number of lanes due to incidents. After applying these disturbances and without rerunning the traffic optimization again, we measure the increases in travel time given the original assignment solutions for each of SO, UE, Tau and UNPM to assess their robustness to traffic disturbances. Based
on Equation 3.2, the set of changes considered in the network include increases of up to 30% in demand and removing up to 2 lanes from a link, and the key performance measures of interest are the average travel time in the network and its increase. The assignment algorithms have no a priori knowledge of disturbances or their probability distribution. This is an important distinction from algorithms based on stochastic optimization that explicitly consider within the optimization problem: the traffic disturbance uncertainty using probability distributions [29, 78], or the probability of network links to operate below their capacities when serving different traffic patterns deviating from the average condition [22].

Figure 3.2: Greater Toronto Highway Network

**Effect of Increasing Demand:**

We study the robustness of SO, UE, Tau and UNPM by analyzing the effect of increasing demand on their solutions. This increase is the difference between the ideal case of expected demand and the actual demand occurring the roadway network. Examples for the demand increase include: special events, unexpected surge in demand in parts of the network in response to road closures nearby the network, and additional traffic demand to vehicles that do not have any wireless or vehicular communication capabilities; thus cannot receive en-route navigation updates, resorting to a shortest path route based on their prior experience. This traffic, which is randomly generated on the 6 s-d pairs with a normalized Gaussian distribution, is equivalent to 10% to 30% of the original traffic and results are averaged over
10 runs.

Fig. 3.3 shows the percentage increase in average travel times for the assignment methods while increasing demand by 10% to 30%. The % increase of the travel times is the smallest for Tau, followed by SO and UNPM with UE having the largest increase in travel times. The performance degradation is far worse in the case of a 30% increase in demand, where travel times go up by more than 90% for UE while they increase by around 80% for SO and UNPM, and by less than 60% for Tau. This indicates that meeting the equilibrium conditions for UE results in a less forgiving network and does not lead to robustness in the presence of disturbances. Note that we only increase demand by up to 30% as the network is already congested at this point and further increases will lead to the breakdown of several highways in the network.

We can also use the raw average travel times to evaluate the robustness of these methods, instead of the % increase of average travel times. These are shown in Fig. 3.4, where the first 4 columns indicate the travel times of the original assignments and the subsequent columns indicate the resulting/new calculated average travel times using the original assignments with increases in traffic demand of 10%, 20% and 30%. The travel times for Tau are the highest before any increases in demand, with the difference being around 5% compared to the SO travel times. This is the price to pay or the trade-off for gaining robustness. However, with increasing demand, Tau provides the lowest travel times starting from a 10% demand increase and performs much better than SO, UE, and UNPM for the case of a 30% increase. This explains the significant deterioration in terms of travel time increases for UE, SO and UNPM also observed in Fig. 3.3. In addition, the performance results of UNPM are almost comparable to that of SO, deteriorating slightly more than SO with increases in traffic demand.
Effect of Decreasing Supply:

Here, we study the effect of decreasing traffic supply due to an incident or weather conditions by removing 1 lane from 1 link, and measuring the new travel time experienced by the original traffic assignment. The lane removal is repeated for all links 1-by-1 and the results are averaged. The same procedure is repeated for removing 2 lanes. Fig. 3.5 shows the % increase in average travel times for the assignment methods due to removing one lane and 2 lanes. It is the smallest for Tau, especially in the 2-lane case, followed by SO, UE and UNPM, which leads to the largest travel time increase. The performance of UNPM is especially deteriorating in this case due to the high traffic it originally assigns to the link from node 1 to node 10 in Fig. 3.2, which only has 3 lanes and its capacity has been reduced by 33% and 66% in the cases of removing 1 lane and 2 lanes, respectively. In the latter case, the performance degradation is around 33% for UE and UNPM, while it is only around 23% for Tau.

The raw average travel times are shown in Fig. 3.6, with the original travel assignment times and
Chapter 3. Robust Network Design for Roadway Networks

Figure 3.6: Increase in average travel time with incidents

those with reductions in the number of lanes. The original travel times for Tau are the highest as before.
They are also high for the case of removing 1 lane, since a lane is removed from only one of the links
at a time, making it a lighter disturbance compared to adding 10% additional demand. However, Tau
provides the best performance when 2 lanes are removed, with UNPM providing the worst performance,
as explained earlier.

3.5 Conclusion

This chapter provided a systematic framework for developing a robust design for roadway networks. After
establishing the various contexts of robustness based on the types of traffic disturbances, it presented
a definition of robustness in roadway networks. Moreover, it discussed the objectives, requirements
and examples of long-term and short-term robust network design, including planning and operations.
Finally, a sample study was presented for a short-term robust network design with a design using traffic
assignment. This showed that robust assignment algorithms, especially with network criticality as a
robustness metric, lead to solutions that yield suboptimal travel times under expected traffic conditions,
but that deliver high performance over a range of unexpected traffic disturbances, even achieving better
travel times than the SO algorithm under certain traffic disturbances.

While enhancements to these robustness metrics and assignment algorithms are presented in Chapter
5 and Chapter 6 to overcome their limitations in large-scale directed roadway networks, the next Chapter
will discuss the speedup of traffic assignment simulation models to enable real-time traffic management
applications.
Chapter 4

Speedup of DTA-Based Simulation for Quasi Real-Time ITS Applications

This chapter presents methods to speed up traffic assignment algorithms through compiler optimizations and parallelism, to enable real-time ITS applications, including dynamic route guidance systems. In addition, it presents extensive experimental results to test the efficiency of these methods and their scalability with different network sizes.

4.1 DTA Simulation Requirements for ITS Applications

The requirements of a dynamic traffic assignment (DTA) model for modelling ITS applications can be categorized into two modes of use: off-line and on-line. Off-line models are typically used to quantify the performance of existing conditions or investigate the effectiveness of ITS strategies. These applications typically require multiple iterations and scenario evaluations using limited computational resources [33].

The need for on-line DTA applications arises in cases where monitoring the transportation network, estimating the network state, and forecasting the next state all are required to be done in real-time to assist operators and management centers in taking informative decision and implement mitigation strategies in real-time. For example, in urban networks, detecting the arrival patterns and presence of vehicles at a traffic light enables algorithms to estimate the network state (e.g. queue length), and therefore take appropriate action (e.g. control the green time of a traffic light) in real-time. Another example, in
rural networks, detecting a high probability of an accident (using incident detection algorithms) would trigger a set of traffic management strategies that need to be evaluated in real-time depending on the characteristics of the accidents.

The need to dynamically model the time-varying flow of vehicles has generated many contributions in developing DTA-based simulation models. The first set of models, called “micro-simulation” models, represent the behavior of each vehicle based on car-following, gap acceptance and lane choice models [33]. Models such as PARAMICS [4], AIMSUN [1], VisSim [5], have a great level of detail and computational complexity; thus, their successful use has been commonly limited to relatively small size networks [33]. The need to model larger networks with reasonable computational times has led to the development of “mesoscopic” simulation models, which provide less detail in modeling of individual vehicle movements but are less cumbersome computationally. Examples include CONTRAM [68], DYNASMART [53] and DynusT [3].

Nevertheless, even mesoscopic simulation models in their typical form do not satisfy real-time requirements when analysing large metropolises with millions of vehicles [60]. The required run-time of DTA models typically consists of 2 components: 1) initial simulation required for network loading and assigning the demand, 2) iterative run-time to reach a certain convergence criterion. The first component is typically most affected by the number of vehicles to be loaded into the simulation and assigning a shortest path for them; while the second is a function of the iterative traffic assignment algorithm and its efficiency in reaching the convergence criterion [3, 79]. Thus, when reporting on run-time of DTA simulation models, it is important to breakdown the time for these 2 components. For example, the run-time for 4 simulation hours (Morning Peak period, from 6-10 AM) of the Greater Toronto and Hamilton Area (GTHA) model, reaching equilibrium conditions after 12 iterations, on an Intel Core I7 workstation running Windows (using 8 processing cores and 12GB RAM ) is approximately 5.5 hours, while the required time for the initial simulation time and network loading is approximately 42.5 minutes. In our view, these computational times make it challenging to assess real-time traffic management strategies and adaptive route guidance systems. Moreover, in cases where only the initial simulation run (referred to as One-Shot simulation in the literature [19]) is required, e.g. to model the effects of incidents, construction and weather conditions, that run-time of 42.5min in the case of the GTHA, is way beyond what could make the simulation output useful to warn or give drivers alternate routing options.

Therefore, research attention has been devoted to speeding up DTA-based microscopic and mesoscopic simulation models using 2 sets of approaches: 1. Using Parallelism with multithreading and 2. Partitioning the traffic network into segments handled by different processors [79]. One example of using
parallelism is SEMSim traffic simulation [14], which uses multi-threading to speed-up an agent-based micro-simulation, based on the assumption that the agents in SEMSim have complete routes to follow in order to reach their destinations, which reduces the number of route calculations required. Another example is CPU/GPU-based speedup for a mesoscopic simulation based on the “Entry Time-based Supply Framework (ETSF)”, which uses the assumption that vehicles on the same lane of a segment are moving at the same speed at a time step [80]. Major issues that need to be considered with such approaches include load-balancing, inter-processor communication and synchronization between processors [79, 80]. On the other hand, an example of using partitioning is MALTA [18], where a network partitioning and recursive on-line load balancing tool is used outside of the DynusT [3] mesoscopic simulator for achieving speedup. Major issues that need to be considered for such approaches include finding good partitions, re-calculating the traffic demand matrices after partitioning, minimizing overheads, and modelling traffic dynamics across segment boundaries [79]. Due to the complexity and major issues of using partitioning, the speedup in this chapter is based on using parallelism and compiler optimization. However, key differences from [14] and [80] in this chapter include focusing on mesoscopic simulation and not imposing assumptions that reduce DTA accuracy as the ones used in [14] and [80].

This chapter discusses the speedup of DTA simulation models using compiler optimizations and parallelism, and enabling them to run on Linux, designed for High-Performance Clusters. DynusT, which stands for Dynamic Urban Systems for Transportation [3], was used as a mesoscopic simulation platform to determine the efficiency of the compiler optimization and parallelisation on a number of simulation models with different network sizes to test the scalability of the system. The experimental setup was tested on calibrated networks, that had used real data to generate the transportation demand and build the network geometry [10,41]. This makes the interpretation and generalization of the results suitable to be used in actual implementations of real-time ITS strategies.

Section 4.2 describes the User Equilibrium implementation in DTA models. Section 4.3 and Section 4.4 present the improvement options using compiler optimizations and parallelism of the DTA model, respectively. Section 4.5 discusses the context of the DynusT DTA model used and the specific optimizations made on it. Section 4.6 presents the performance results of the system on a 2 large-scale models in the GTHA. Finally, Section 4.7 presents the conclusion.

### 4.2 User Equilibrium Implementation in DynusT

DynusT [3] is a dynamic mesoscopic traffic simulator, which means that it adjusts link costs mid-simulation based on their use. In order to simulate User Equilibrium (UE) Assignment, described in
Section 2.4, DynusT iterates over the following steps (1-4), illustrated in Figure 4.1:

1. DynusT analyses which route is the shortest for each vehicle, and assigns vehicles initial paths in a distributed fashion. This is similar to an unfamiliar driver choosing a route assuming free-flow conditions.

2. DynusT performs an initial simulation, in which drivers follow the assigned routes to completion without changing course. This centrally updates the route costs based on the routes taken by each of the vehicles. This is essential as the assumed route costs in the previous step might be incorrect since the distributed algorithm cannot take into account the paths taken by other vehicles and their effect on route costs in the network.

3. These initial conditions leave some vehicles taking routes with higher cost than their alternatives. DynusT will perform dynamic assignment, re-assigning a % of the vehicles to their respective best-routes according to the calculated route costs [19].

4. It will then perform a simulation again, and updates the route costs based on this dynamic assignment.

DynusT will then repeat steps 3 and 4, reassigning traffic based on the latest simulation, and resimulating with this new assignment. It will stop when one of the following occurs:
1. The simulation iterates a maximum number of times, as assigned by the user running the simulation.

2. The simulation reaches convergence, defined as when all vehicles have no incentive to switch routes, i.e. when the gap between the current assignment solution and the ideal shortest route time, divided by the total shortest path times (a ratio called the relative gap), is below a pre-specified tolerance level [19].

Each simulation emulates a period in time, and as the simulation time progresses, new vehicles navigate through the network. Assigning the routes for these new vehicles based on the shortest paths (determined at the start of the simulation) does not represent reality simply because network conditions dynamically change over time. That is why it is important to distinguish between what is referred to as the instantaneous travel time (cost) vs experienced travel time (cost) within a transportation network. The instantaneous travel time refers to the travel time calculated when the routes are generated without considering congestion during subsequent time periods, while experienced travel times account for the times needed for traversing various links using the expected congestion state of those links during the times of entering those downstream links [19]. As illustrated in Figure 4.2, DynusT performs basic shortest path analysis and assigns routes for newly generated vehicles. These new vehicles then form a part of the simulation and continue in the same manner as the ones loaded in the previous time step. This process repeats many times per simulation (known as assignment interval), once for each new batch of vehicles entering the network.

It is important to note that due to its algorithmic structure and software implementation, DynusT is capable of performing DTA on regional-level networks over a long simulation period. This is primarily due to the Anisotropic Mesoscopic Simulation (AMS) algorithm [21] used in DynusT to perform the

![Figure 4.2: Adding vehicles mid-simulation in DynusT](image)
traffic assignment. As a mesoscopic algorithm, DynusT does not simulate car following, lane changing and gap acceptance within individual vehicles (as a microscopic simulator would), however it emulates system responses to factors affecting individual vehicles, such as queues for making left turns. AMS simulates groups of vehicles which are in close proximity using the so-called “Speed Influencing Region” (SIR), enabling it to differentiate speeds and other characteristics for vehicles on the same link but in different SIRs. This is in contrast to other mesoscopic models that imprecisely assume that traffic flow/speed/density are uniform along these links [21]. The next sections present the compiler optimizations and parallelism that can be used to speed up DTA model run-times.

4.3 Compiler Optimizations

Compiler Optimizations are modifications to how algorithms are implemented in the processors to maximize their efficiency, reducing their run-time [48]. There are many compiler optimizations that are possible for every programme code. These optimizations generally require understanding of the algorithm being implemented and exploiting that provides options for further speedup using parallelism. These options include manual optimization and automatic optimization.

4.3.1 Manual Optimization

Manual optimization is making a set of changes in the way a code is compiled, based on specific knowledge of certain bottlenecks in it [48]. One example is loop optimizations, which act on statements that execute the same operation until the condition to exit the loop is satisfied. These optimizations can lead to significant speedup as programs can spend the majority of their run-time inside loops that might be designed inefficiently [44]. Figure 4.3 shows an example of loop unrolling, where the body of the loop is unrolled into 2 independent statements that can be easily executed on 2 separate processors at the same time, giving a speedup of around 2 [44].

<table>
<thead>
<tr>
<th>Normal loop</th>
<th>After loop unrolling</th>
</tr>
</thead>
<tbody>
<tr>
<td>for (i = 0; i &lt; 100; i++) { a[i] = b[i]; }</td>
<td>for (i = 0; (i+1) &lt; 100; i += 2) { a[i] = b[i]; a[i+1] = b[i+1]; }</td>
</tr>
</tbody>
</table>

Figure 4.3: Loop unrolling example, modified from [44]
4.3.2 Automatic Optimization

Automatic optimization is allowing the compiler to look through the code automatically to try to find options for automatic parallelism [48]. There are 4 different optimization levels for codes written in C, C++ or Fortran and compiled with the Intel C++ Compiler (ICC) [7]:

1. When compiled with “O0” for “optimization level 0”, the code is completely unoptimized.
2. “O1” is optimized to create the smallest (by size rather than time) optimized code in most cases.
3. “O2” is optimized as far as it can deterministically speed up the code.
4. “O3” has aggressive optimizations applied based on heuristic approaches that are likely to boost the speed, but not guaranteed. These include automatic loop unrolling for all loops (which could be beneficial or disadvantageous based on the specific scenario).

4.4 Parallelism

Programming for a high-performance environment to achieve speedup makes use of a number of techniques to maximise efficiency. Multi-core computers have multiple, separate processors (or processor cores), which individually execute sequential commands. The code executed on each processor (core) is known as a thread, and aims to utilize powerful multi-core computers to speed up execution time. While a perfectly parallel programme can run n-times faster utilizing n threads, there are a number of issues that may limit speedup in practical applications. In the following sections we discuss speedup techniques using parallelism and their limitations.

4.4.1 Critical Sections of Code and Load Balancing

Critical sections of a programme are ones in which all threads must be at the same place in order for the critical section to run. This is usually due to data the algorithm needs from all threads, which requires that all threads finish calculating the needed data for the algorithm to continue. This is problematic because processors that finish first must wait until the slowest one finalizes its execution before continuing. To illustrate, in the case of DynusT, there are 3 sections of explicit critical code, as well as 12 places in which a parallel section ends. Since the sequential section following each ending parallel section will need data calculated in the parallel section, this serves as an implicit critical section. It forces all processors to catch up to the same point before any of them can continue into the sequential code.
Critical sections become especially important if the number of calculations is not spread evenly amongst all processors. For instance, Algorithm A is run in parallel over 4 processors in Figure 4.4, but 3 processors must wait idly while 1 finishes a longer job before the first critical section. It then parallelises again, but due to poor load balancing between the processors, over half of the processor-time is wasted. If the first critical section in Figure 4.4 was omitted, then some of the differences between processor execution times would be cancelled, improving the execution time significantly. Note that exploiting the knowledge of these critical sections enables making certain compiler optimizations to improve the load balancing between the processors and thus lead to further speedup.

Figure 4.4: Critical sections resulting in wasted resources

4.4.2 Sequential Sections

Another shortcoming of parallel code is that it does not always scale linearly with the number of processors running it. A major bottleneck here is sections of the code that must run in series - that cannot be made parallel. Regardless of how many processors we have, the same amount of time must be spent on the serial sections, thus limiting how many times faster the overall code can run. This is called the speedup ($S$) and this limit is quantified by Amdahl’s Law [12]:

$$S = \frac{1}{f + \frac{1-f}{P}}$$

where, $S$ is the speedup; i.e. how many times faster the parallel code runs on multiple processors than its serial counterpart on 1, where $P$ is the number of processors being used, and $f$ is the fraction of the code that can not be run in parallel ($0 \leq f \leq 1$). This means that increasing the number of processors does not proportionally increase the speedup. In Figure 4.5, we see an example of code that has an $f$ of
0.5 (50% parallelisable, 50% serial). This makes a theoretical speedup of:

\[ S = \frac{1}{0.5 + \frac{1 - 0.5}{P}} \]

and taking the limit of infinite processors,

\[ \lim_{P \to \infty} (S) = \frac{1}{f} = \frac{1}{0.5} = 2 \]

This is a well-known corollary of Amdahl’s law, and means that the speedup cannot exceed 2. This is because as we increase the number of processors, the parallel code’s execution time decreases, however the serial code’s execution time remains constant. As the number of processors approaches \( \infty \), the execution time approaches the serial code’s time, as seen in Figure 4.5. This however, represents the ideal case, in which the parallel section scales linearly (doubling the processors doubles the speed). This is not always achievable, due to delays at critical sections as described earlier.
4.4.3 Parallelism with Threading

Running a programme using multi-threading involves using the OpenMP compiler extension to implement its explicit parallelism. Threaded code runs across all processors in a single machine — this enables having parallel code running on at most the number of processors on the machine. For instance, a dual-core machine can only run two threads efficiently. The number of processors a programme is executed on is therefore limited to the number of cores on the machine.

4.4.4 Parallelism Across Nodes

If we wish to add more processors than the number available on one machine, we can use multiple machines. In most high-performance environments, these are called nodes, and are connected with network cables but are distinct units. A major drawback of multi-node processing is that there must now be communication between the nodes, something that takes time. Furthermore, as described in Section 4.4.2 there are diminishing returns on adding processors. For these reasons, most DTA models do not use parallelism across nodes. The next section covers the context of the DynusT DTA model used and the specific optimizations to exploit its speed up.

4.5 The Context of the DynusT DTA Model

This section discusses how compiler optimizations and parallelism were used to speed up the DynusT DTA model.

4.5.1 Compiler Optimizations on DynusT DTA Model

After analyzing the implementation of the traffic assignment algorithms in the DynusT DTA model (Section 4.2) and their performance in detail, we performed many manual optimizations within the DTA model (to enable achieving further speedup using parallelism), including the following: 1. vectorising matrix calculations (e.g. directly summing a vector of vehicle data rather than 1-by-1), 2. removing unnecessary code jumps using loop unrolling, and 3. changing the implementation to reduce memory-access, by making it use as much of the processor cache as much as possible, rather than system memory [44]. In addition, the DTA model was run using the 3 automatic optimization levels, O0, O2 and O3 (O1 was not used as it optimizes for space rather runtime).
4.5.2 Parallelism in DynusT

Analysing how parallelism can be used to speed up the DynusT DTA model, our findings showed that explicit critical sections occur only 3 times in DynusT. One of these occurrences is in the section of code that moves all vehicles mid-simulation (vehicles move along a road as time progresses). Periodically in the simulation, DynusT needs to write the information (e.g. location) for each vehicle to an array to use it later for dynamic assignment. It is important that all vehicle locations written to this array are written at the same time, thus making this code a critical section. Since this section occurs in a loop to move all vehicles, the parallel-critical transition scenario seen in Figure 4.6 happens many times (three are illustrated). For instance, if the cycle in Figure 4.6 were to occur once for each vehicle, the run-time would increase proportionally to the number of vehicles. This amplifies the delay caused by the critical section, for a large network like the Greater Toronto and Hamilton Area with 1.6 million vehicles.

Figure 4.6: Critical sections in a loop in vehicle moving code

Based on our analysis, DynusT has an empirical $f$ value of 0.163 (the serial part of the code takes 16.3% of the total time with 1 processor as explained in Section 4.4.2). Thus, its theoretical speedup is $1/f = 6.126$. This represents the ideal case, in which the parallel section scales linearly (doubling the processors doubles the speed). This is not the case in DynusT, due to delays at critical sections as described in Section 4.4.1. This results in DynusT’s execution time not reaching that asymptote, as explained in more detail in Section 4.6.3. Finally, running with OpenMP enables DynusT to run parallel operations, such as assigning vehicles to a route based on the costs for each route — item 3 in Figure 4.1. Thus, we used multiple processors to achieve speedup in DynusT (up to 8 processors based on the theoretical speedup findings). The performance analysis results with DynusT are discussed next.
4.6 Application to DynusT: Performance Analysis

In this section we apply the compiler optimization and parallelism approaches discussed above into DynusT, a mesoscopic simulation software that is widely used in many cities. DynusT is originally designed as a Windows-based programme. As many High Performance Computing (HPC) facilities [39] are rapidly emerging to be running on Linux systems, the DynusT source code was converted to run on Linux as a first step. As Windows and Linux differ in their performance running the same programme, we discuss some of these differences and show extensive testing on the performance of DynusT for Linux under various conditions.

4.6.1 Experimental Setup

In order to assess the performance of the proposed speed-up approaches, the experiments were designed to study the following factors: efficiency (with number of CPUs and optimization levels), performance in various systems (Linux vs Windows), and scalability (with different network sizes).

The simulation test networks include those of the Greater Toronto and Hamilton Area (GTHA) [10] and the Greater Toronto Area (GTA) [41], which have been extensively calibrated using real data. These networks are shown in Figure 4.7. They are different in size (shown in Table 4.1), based on these 3 criteria: 1. $n$, the number of nodes (intersections) and origin/destination points in the network, 2. $m$, the number of links (roads) in the network, and 3. $v$, the number of vehicles making a trip in the simulation. Each simulation emulates the morning traffic period from 6:00-10:00am, and a number of iterations to reach user equilibrium conditions. In the case of the 2 test networks, the number of iterations required for convergence was found to be 12. Extensive testing was performed on the performance of DynusT under various conditions, using memory use, CPU use and execution time as metrics. The number of processors used ranges from 1 (serial) to 8 on an Intel Core I7 workstation with 12GB RAM and the compiler optimization levels used are optimization level 0, 2 and 3.

<table>
<thead>
<tr>
<th></th>
<th>GTHA</th>
<th>GTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>11,713</td>
<td>14,225</td>
</tr>
<tr>
<td>$m$</td>
<td>29,184</td>
<td>26,444</td>
</tr>
<tr>
<td>$v$</td>
<td>1,981,571</td>
<td>1,602,717</td>
</tr>
</tbody>
</table>
4.6.2 CPU and Memory Usage

DynusT executes a DTA algorithm described in Section 4.2, using multi-threading with OpenMP (described in Section 4.4) for efficiency. To illustrate the required resources to run the GTHA network until convergence (using 12 iterations), a series of figures are created. In Figures 4.8 and 4.9, the CPU usage and memory usage are illustrated against the actual run-time (x-axis) for 12 iterations, respectively. Additionally, Figure 4.10 helps provide more insights into the resources allocated during one iteration (after the 0th iteration).

Figures 4.8 and 4.9 show fluctuations in CPU and memory usage with the evolution of the simulation iterations. These correspond to the steps of the DTA algorithm described in Section 4.2. On the one hand, the 0th iteration (during which network-loading and initial short paths are generated for all
vehicles as shown in Figure 4.1) exhibits a large memory peak as seen in Figure 4.9 and takes longer than subsequent iterations, as expected. On the other hand, Subsequent iterations only assign newly generated vehicles to shortest paths based on DynusT’s traffic assignment algorithm (described in Figure 4.2).

Figure 4.10 provides a clearer view of the iterative simulation (the initial section until around 8.5 min), and the dynamic assignment following it, described in Section 4.2. It also shows the variation in CPU usage, where the periods using 1 CPU (100% CPU Usage, rather than 400%) represent the serial sections of DynusT’s code execution.

As seen in Figures 4.9 and 4.10, the memory usage within a typical iteration increases overall as the simulation progresses. However, a large drop of memory (and CPU) is observed around the 8 min mark in Figure 4.10; this is due to freeing of memory and writing vehicle info into files (usually by 1 processor) to be used for reassigning some vehicle paths during the dynamic assignment step. Note that this period for writing vehicle info to files is higher (lower) when the number of vehicles in a network is higher (lower). Other drops in memory are attributed to freeing of memory between iterations and within each iteration, seen around the 1.5 and 4.5 min marks in Figure 4.10, before new vehicles are loaded into the network (described in Figure 4.2). Overall, the maximum memory usage reaches around 9GB, seen in Figure 4.9, which is below the memory capacity of the simulation workstation.

4.6.3 Evaluation with Compiler Optimizations and Parallelism

In addition to the manual optimizations implemented (as discussed in Section 4.3.1), this section discusses how the execution time of DynusT can be reduced using 2 methods:

1. Compiler optimizations, through automatic parallelism, with levels 0, 2, and 3 for optimization.
2. Explicitly parallel code, using OpenMP. This is defined by the number of cores DynusT is “allowed” to use in one of its configuration files.

Table 4.2 summarizes how the run-time of DynusT for the GTHA network changes using different compiler optimization levels and number of cores. The columns break down each section of a DynusT run into the following components: The “Initial Simulation” time, is for the 0th iteration, i.e. items 1 and 2 of Figure 4.1. The “Iterative Simulation” column lists how long the iterative simulations (item 4 in Figure 4.1) took, averaged over all iterations. Likewise, the “Dynamic Assignment” column lists the averaged execution times for assignment in-between simulations (item 4 of Figure 4.1). The total execution time simply equals initial simulation + number of iterations × (iterative simulation time + dynamic assignment time). Note that the optimization levels O0, O2, and O3 (O1 was not used as it optimizes for space and its runtime is very similar to O0) and the number of cores are included in the 2nd column next to the operating system used. The first 4 rows help compare the optimization levels O0, O2, and O3, and the remaining rows shown in Table 4.2 are all using the O3 setting.

Comparing the run-times with 4 cores between optimization levels 0, 1, and 3 (rows 3, 4, and 8), there is a drop in execution times, of almost 50% between O0 and O3. The drop is also similar for the serial case when comparing optimization levels O0 with O3 (rows 2 and 5). Although the aggressive optimizations applied with the O3 setting are not guaranteed to speedup run-time due to the heuristic methods they use, they provide better speedup than optimization level O2 here.

Comparing the total run-times with increasing number of cores (1 to 8) with O3, rows 5-12 in Table 4.2, show the expected asymptotic behaviour, reaching a minimum time with 4 cores or a speedup of 1.9x
Table 4.2: DynusT execution times on the GTHA network

<table>
<thead>
<tr>
<th>Row</th>
<th>Exec. Time (hh:mm:sec)</th>
<th>Initial Simulation</th>
<th>Iterative Simulation</th>
<th>Dynamic Assignment</th>
<th>Total (12-iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Windows O3 - 8 Cores</td>
<td>00:42:29</td>
<td>00:14:28</td>
<td>00:09:24</td>
<td>05:33:18</td>
</tr>
<tr>
<td>2</td>
<td>Linux O0 - Serial</td>
<td>01:50:18</td>
<td>00:11:39</td>
<td>00:48:41</td>
<td>13:54:26</td>
</tr>
<tr>
<td>3</td>
<td>Linux O0 - 4 Cores</td>
<td>00:36:39</td>
<td>00:07:35</td>
<td>00:21:12</td>
<td>06:21:54</td>
</tr>
<tr>
<td>4</td>
<td>Linux O2 - 4 Cores</td>
<td>00:19:25</td>
<td>00:05:30</td>
<td>00:10:12</td>
<td>03:27:51</td>
</tr>
<tr>
<td>5</td>
<td>Linux O3 - Serial</td>
<td>00:44:35</td>
<td>00:07:17</td>
<td>00:20:20</td>
<td>06:16:03</td>
</tr>
<tr>
<td>6</td>
<td>Linux O3 - 2 Cores</td>
<td>00:17:17</td>
<td>00:07:10</td>
<td>00:09:37</td>
<td>03:38:43</td>
</tr>
<tr>
<td>7</td>
<td>Linux O3 - 3 Cores</td>
<td>00:20:56</td>
<td>00:05:20</td>
<td>00:10:55</td>
<td>03:35:59</td>
</tr>
<tr>
<td>8</td>
<td>Linux O3 - 4 Cores</td>
<td>00:17:54</td>
<td>00:05:10</td>
<td>00:09:51</td>
<td>03:18:06</td>
</tr>
<tr>
<td>9</td>
<td>Linux O3 - 5 Cores</td>
<td>00:17:28</td>
<td>00:05:36</td>
<td>00:09:48</td>
<td>03:22:10</td>
</tr>
<tr>
<td>10</td>
<td>Linux O3 - 6 Cores</td>
<td>00:16:57</td>
<td>00:05:42</td>
<td>00:09:50</td>
<td>03:23:16</td>
</tr>
<tr>
<td>11</td>
<td>Linux O3 - 7 Cores</td>
<td>00:16:33</td>
<td>00:05:50</td>
<td>00:09:59</td>
<td>03:26:29</td>
</tr>
<tr>
<td>12</td>
<td>Linux O3 - 8 Cores</td>
<td>00:16:27</td>
<td>00:06:06</td>
<td>00:10:01</td>
<td>03:29:52</td>
</tr>
</tbody>
</table>

(row 8). However, as the number of cores increases beyond 4, the run-time increases. This is because beyond a certain number of cores, the communication latency between the cores cancels out and then surpasses the benefits of the multi-core speedup. Putting these two results together indicates that the optimal conditions for DynusT’s overall speedup are optimization level 3 running on 4 cores.

On another note, the execution time of the initial simulation, goes down to only 16.5 min (2.5x speedup compared to serial) using 8 cores (has less critical sections). This saving in initial simulation time is essential in cases where 10’s or 100’s of initial simulations are required. For example, determining the optimal toll structure for a regional highway network requires multiple evaluations of dynamically assigning of vehicles into the network in response to the toll structure.

Finally, the run-times on Windows improve with higher optimization levels and an increasing number of cores (not shown in Table 4.2), reaching the lowest times with 8 cores (shown in row 1). However, the total time is 1.7x longer and the initial simulation time is 2.4x longer than the times of Linux O3 - 4 Cores (row 8). This is due to:

1. The manual optimizations made on the DynusT code after converting it to run on Linux (discussed
in Section 4.3.1)

2. The removal of extra visual DynusT pop-ups that are computationally intensive in Windows

3. The higher control that Linux provides to multi-threaded programmes (rather than randomly pausing the computation of a core due to various Windows interrupts)

### 4.6.4 Variation with Network Size

In addition to analyzing the performance with compiler optimizations and parallelism on the GTHA network, we have examined DynusT’s performance on the smaller GTA network (Figure 4.7), to determine whether the compiler optimizations and parallelism can provide the same improvements in run-time as the GTHA network. The same analysis as in the previous section was conducted on the GTA network, which is a subset of the GTHA network (see Table 4.1).

<table>
<thead>
<tr>
<th>Row</th>
<th>Exec. Time (hh:mm:sec)</th>
<th>Initial Simulation</th>
<th>Iterative Simulation</th>
<th>Dynamic Assignment</th>
<th>Total (12-iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Windows O3 - 8 Cores</td>
<td>00:17:00</td>
<td>00:09:14</td>
<td>00:09:40</td>
<td>04:05:51</td>
</tr>
<tr>
<td>2</td>
<td>Linux O0 - Serial</td>
<td>01:31:31</td>
<td>00:09:05</td>
<td>00:39:35</td>
<td>11:15:23</td>
</tr>
<tr>
<td>3</td>
<td>Linux O0 - 4 Cores</td>
<td>00:31:31</td>
<td>00:06:22</td>
<td>00:20:55</td>
<td>05:58:54</td>
</tr>
<tr>
<td>4</td>
<td>Linux O2 - 4 Cores</td>
<td>00:17:39</td>
<td>00:05:21</td>
<td>00:09:18</td>
<td>03:13:27</td>
</tr>
<tr>
<td>5</td>
<td>Linux O3 - Serial</td>
<td>00:37:36</td>
<td>00:06:15</td>
<td>00:19:34</td>
<td>05:46:46</td>
</tr>
<tr>
<td>6</td>
<td>Linux O3 - 2 Cores</td>
<td>00:14:17</td>
<td>00:06:17</td>
<td>00:09:01</td>
<td>03:17:50</td>
</tr>
<tr>
<td>7</td>
<td>Linux O3 - 3 Cores</td>
<td>00:14:05</td>
<td>00:05:49</td>
<td>00:09:05</td>
<td>03:12:53</td>
</tr>
<tr>
<td>8</td>
<td>Linux O3 - 4 Cores</td>
<td>00:14:02</td>
<td>00:05:32</td>
<td>00:09:02</td>
<td>03:08:44</td>
</tr>
<tr>
<td>9</td>
<td>Linux O3 - 5 Cores</td>
<td>00:13:43</td>
<td>00:04:41</td>
<td>00:09:05</td>
<td>02:58:59</td>
</tr>
<tr>
<td>10</td>
<td>Linux O3 - 6 Cores</td>
<td>00:13:13</td>
<td>00:04:54</td>
<td>00:09:26</td>
<td>03:05:13</td>
</tr>
<tr>
<td>11</td>
<td>Linux O3 - 7 Cores</td>
<td>00:13:00</td>
<td>00:04:59</td>
<td>00:09:25</td>
<td>03:05:40</td>
</tr>
<tr>
<td>12</td>
<td>Linux O3 - 8 Cores</td>
<td>00:13:08</td>
<td>00:05:10</td>
<td>00:09:08</td>
<td>03:04:54</td>
</tr>
</tbody>
</table>

Similar to Table 4.2, Table 4.3 shows the execution times for the GTA network. We see a considerable drop in execution time, of almost 50% between O0 and O3 with 4 cores (rows 3 and 8 in Table 4.2). Moreover, we observe a speedup of 1.9x as the number of processors increases from 1 to 4, similar to
the GTHA. However, the run-time is lowest for 5 (rather than 4) processors for the GTA, reaching a speedup of 2x. This is because the amount of data being communicated between processors for the GTA network (with 23% fewer vehicles) is less than that in the GTHA network, allowing the usage of 1 more processor before the communication latency cancels out and surpasses the multi-core speedup. This was also observed with a shorter period for writing vehicle info to files in the GTA network (not shown here) between the iterative simulation and dynamic assignment for example compared to the period around the 8-min mark in Figure 4.10, due to the lower number of vehicles in the network. In addition, the execution time of the initial simulation for the GTA, goes down to only 13 min (2.9x speedup) using 8 cores. This shows that reducing the amount of information being communicated in a parallellised DTA model enables getting further speedup using more cores.

4.7 Conclusion

This chapter focused on the speedup of DTA simulation models because of the needs to have numerous evaluation runs for optimization and to enable real-time traffic management applications. The methodologies included using both compiler optimizations and parallelism. DynusT as a widely used DTA model was evaluated as a test case, while its results could be generalized because we have used real-networks and calibrated them using real data sets in the Greater Toronto and Hamilton Area (GTHA). Extensive testing was performed to evaluate various dimensions for speed-up including: network size, number of processors, various optimization levels and operating systems. The results showed that compiler optimizations and parallelism allowed the execution time for a 12-iteration simulation run and an initial simulation to be reduced using around 4 cores by around 50% and 60%, respectively. Moreover, they provided an important insight that speed-up with increasing number of cores is achievable but up to a certain point, depending on the communication latency between the cores. Therefore, reducing the amount of information being communicated in a parallellised DTA model enables getting further speedup using more cores. An added contribution of this work is that the Linux version of DynusT works on more high-performance clusters, and can thus be run using the fastest available processors. The next Chapter introduces a hybrid approach for robustness in roadway networks.
Chapter 5

Robustness in Roadway Networks: A Hybrid Approach

This Chapter introduces a hybrid approach for measuring robustness in roadway network and presents algorithms to calculate the developed robustness metrics efficiently in large-scale networks. Additionally, it presents simulation analysis with various sized networks, showing how these metrics can guide traffic planners and operators identify critical road sections to help proactively alleviate traffic congestion.

5.1 The Need for a Hybrid Approach for Robustness

The economic cost of traffic congestion in the Greater Toronto and Hamilton Area (GTHA), was estimated to be approximately $3.3 billion/year according to a 2008 study by Metrolinx, the provincial transit agency in Ontario [6]. This cost considers travel delays, impact to the environment, vehicle operating costs, and the chance of being involved in vehicle collisions [6]. Moreover, Dachis et al. [25] estimate additional costs of $1.5-$5 billion/year for the GTHA by taking into account all the losses of unrealized income from better employment, productivity and labor participation rate.

While recurring congestion is normally the focus in planning and investment decisions, almost half of traffic congestion is caused by non-recurring disturbances, primarily caused by incidents, vehicle breakdowns, road construction activities, special events, extreme weather events, etc. [24, 54]. In the Province of Ontario, 35.6% of motor vehicle collisions occur during the morning (6-9am) and afternoon peaks (4-7pm), adversely contributing to the congestion levels during these periods [8]. Toronto, the largest city in Ontario, with 1.2 Million motor vehicle registrations, contributes to 32,000 collisions yearly,
i.e. 18.6% of the Ontario total or about 88 collisions every day [8]. This makes traffic disturbances due to weather conditions and incidents part of everyday commute in a city like Toronto.

To understand the effect of traffic disturbances and reduce their impact on roadway networks, it is important to analyze traffic disturbances and measure roadway network robustness in light of these disturbances [16,30,47,56,65]. This is essential for identifying the critical sections of a roadway network, which can guide transportation planners and operators to alleviate traffic congestion, through traffic planning and dynamic control strategies, respectively. The two approaches for measuring robustness and identifying critical road sections (nodes/links) are: 1. topological analysis by adapting metrics from network science and graph theory, and 2. operational analysis of traffic measures, such as speeds, flows, densities, travel time index, etc. On the one hand, topological analysis is a data science approach based on network modelling using metrics such as the Shortest Path Betweenness Centrality (SP-BC), which measures the extent a node/link contributes to the flow of traffic in a network [46,61,69]. On the other hand, operational analysis is a data analysis approach based on statistical traffic measure variations and building robustness metrics based on these measures [66,74]. While topological metrics such as SP-BC have been used in fields such as communication networks [71], smart grids [11], and recently transportation networks [47,50], these metrics often simplify (and sometimes overlook) the traffic dynamics in roadway networks. On the other hand, solely relying on operational metrics may provide myopic results under various traffic conditions due to ignoring the importance of topology in modelling roadway network robustness [45].

This chapter develops a hybrid approach for measuring roadway network robustness by extending the scope of the SP-BC metric for application in roadway networks and augmenting it with traffic flow metrics to model dynamic traffic changes. It also presents algorithms to calculate these metrics efficiently for large-scale networks. Section 5.2 extends the SP-BC metric to develop a topological and a hybrid robustness metric for roadway networks. Section 5.3 presents algorithms to calculate the topological robustness metric for large-scale networks. Section 5.4 presents a performance evaluation of these algorithms, along with analysis of the PK-BC and HCI metric in the GTA network. Finally, Section 5.5 provides the conclusion.
5.2 Developing a Hybrid Robustness Metric for Roadway Networks

This section defines the Shortest Path Betweenness metric and illustrates how it is computed for a small roadway network. It then extends it to develop a topological robustness metric and a hybrid robustness metric for roadway networks.

5.2.1 Shortest Path Betweenness

Unlike the random walk betweenness centrality, described in Chapter 3, which was found to not perform well in large-scale directed roadway networks, the Shortest Path Betweenness Centrality (SP-BC, BC, or betweenness) of a network component (node/link) is defined as the fraction of shortest paths between all pairs of nodes in a network that pass through it [46, 61, 69]. The SP-BC measures the importance of every network component to the whole network in case of its failure or disruption. Therefore, this measure identifies critical network components.

Formally, let a roadway network be represented by a directed graph $G(N, E, W)$, where $N$, $E$, and $W$ denote the node set, link set, and link weight matrix, respectively. While a link represents a road segment between nodes $i$ and $j$ and denoted by $l = (i, j)$ with weight $w_l$, a node represents a trip origin/destination/junction of road segments or key points on a highway where the roadway physical characteristics change (such as change in number of lanes, merging/diverging points, or interchanges). Let $SP_{o,d}$ be the number of shortest paths between the origin node $o \in N$ and the destination node $d \in N$, and $SP_{o,d}(k)$ be the number of shortest paths from $o$ to $d$ that pass through network component (node/link) $k$. Thus, SP-BC of $k$ can be defined by Equation 5.1 [46, 61, 69]:

$$BC(k) = \sum_{o,d \in N} \frac{SP_{o,d}(k)}{SP_{o,d}} \quad (5.1)$$

$BC(k)$ is a summation over all network origin-destination (o-d) pairs, which considers the importance of network component $k$ to the shortest paths between these o-d pairs. To best illustrate how the SP-BC is computed, we present a small roadway network with 6 nodes and 6 links, known as the fish network a shown in Fig. 5.1, where traffic flows from node 1 to node 6. Link 1->2 is 15818 feet long and link 5->6 is 15880 feet long. Additionally, the sum of the distances of links 2->3 and 3->5 (38038 feet) is a bit shorter than that of links 2->4 and 4->5 (38039). Therefore, there is one shortest path for the flow between o-d pair 1->6, which is denoted as Path A: from node 1->2->3->5->6, while Path B: from
node 1->2->4->5->6 is slightly longer.

Table 5.1 shows the calculation of the SP-BC values for every link k in the example fish network, by summing the proportion of the shortest paths passing through each link, i.e. \(SP_{o,d}(k)/SP_{o,d}\), over all o-d pairs shown by the various columns in Table 5.1. For example, for o-d pair 1->6 (refer to column 1), links 2->4 and 4->5 are on 0 shortest paths as the shortest path (Path A) does not go through them, while the other links have a value of 1 as they belong to the only shortest path between o-d pair 1->6.

<table>
<thead>
<tr>
<th>Link</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>o-d</th>
<th>All o-d's</th>
<th>All o-d's</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-&gt;2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>19.2%</td>
</tr>
<tr>
<td>2-&gt;3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6 23.1%</td>
</tr>
<tr>
<td>2-&gt;4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2 7.7%</td>
</tr>
<tr>
<td>3-&gt;5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6 23.1%</td>
</tr>
<tr>
<td>4-&gt;5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 7.7%</td>
</tr>
<tr>
<td>5-&gt;6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5 19.2%</td>
</tr>
</tbody>
</table>

Table 5.1: SP Betweenness calculation in the fish network

We follow a similar analysis over all possible o-d pairs in Table 5.1, with the last 2 columns representing the sum and normalized sum (as a %) of link SP-BC values, respectively. As shown in Fig. 5.2, this normalized sum allows to understand link betweenness as its relative importance to the whole network. Therefore, based on this metric, the following observations can be made:
1. Links 1->2 and 5->6 are highly important to the network, which makes sense as there are no alternative links for any traffic originating in node 1 or ending in node 6.

2. Link 2->3 is found to be 3 times more important than link 2->4, while Path A (that passes through link 2->3) is slightly shorter than Path B (38038 vs 38039 feet).

3. Link 2->3 is around 20% more important relatively than link 1->2 (Fig. 5.2), although link 1->2 does not have any alternate links; while link 2->3 has the alternate of link 2->4 for any traffic not ending at node 3.

Given the above noted shortcomings of the SP-BC metric, a modified version of this metric, referred to as the PK-BC metric, is introduced in the next section.

### 5.2.2 Developing the PK-BC Metric for Roadway Networks

The first modification to the SP-BC metric includes extending the number of shortest paths used for the calculation from 1 to K. While BC considers the shortest paths between all o-d pairs that have the same path cost, it ignores paths that are slightly longer. On the other hand, to calculate the modified metric, which we refer to as the All-OD K-Shortest-Path Betweenness metric or AK-BC for short, the K shortest paths among all paths between o-d pairs are used for the calculation. The choice of K depends on the size of a network and the number of possible alternate paths, ranging from 2 to 5, given that drivers typically have a handful number of alternate routes to choose from. For K=1, the AK-BC metric is equivalent to the BC metric, and for K=∞, it will account for all possible paths for an o-d pair.

Following a similar calculation as in Table 5.1 and using K=2, the normalized AK-BC values can be calculated as shown in Fig. 5.3. By considering 2 shortest paths (K=2) for each o-d pair, this metric resolves the issues with links 2->3 and 2->4 in Fig. 5.2 as it reasonably assigns them equal importance. However, the values for links 2->3 and 2->4 are still relatively high when compared to links 1->2 and 5->6. This is because the summation over all network o-d pairs assumes that traffic can originate and end in every node in the network with equal weight. Although this is a common and reasonable assumption in many types of computer or communication networks, many nodes in roadway networks are just internal nodes to the network forming junctions between roadway segments (e.g. a change in

![Figure 5.3: AK-BC values for Fish network](image_url)
road geometry or in number of lanes); thus are not necessarily origins or destinations of traffic. Given that traffic in this fish network only flows from node 1 to 6, AK-BC myopically assigns a high importance to links 2->3 and 2->4 as they belong to shortest paths of traffic starting and ending in internal nodes 2, 3, 4, and 5.

Therefore, the AK-BC metric is modified to consider only possible o-d pairs in the network, calling it the Pre-Determined OD K-SP Betweenness metric or PK-BC for short, as the possible o-d pairs are typically known by transportation planners (subject to the location of zones/connectors/centroids, etc.).

Let $\text{K-SP}_{o,d}$ be the number of K shortest paths between an origin node $o$ and a destination node $d$ (among the set of destination nodes $D_o$ for this origin $o$), and $\text{K-SP}_{o,d}(k)$ be the number of K shortest paths from $o$ to $d$ that pass through network component $k$. Thus, PK-BC of $k$ can be defined by Equation 5.2:

$$PK-BC(k) = \sum_{o \in N} \sum_{d \in D_o} \frac{\text{K-SP}_{o,d}(k)}{\text{K-SP}_{o,d}}$$

It is noteworthy that this metric was first introduced for communication networks in [69] under the name “deterministic betweenness”, signifying o-d pairs that are already determined. In our case, the metric considers all possible paths for an o-d pair, i.e. $K=\infty$. A similar version was later used in [61] for environmentally-aware traffic monitoring in transportation networks, by modifying the original SP-BC metric (i.e. using $K=1$). On the one hand, using $K=\infty$ is computationally cumbersome for large roadway networks and practically unnecessary due to the relatively small number of routes that are typically considered by drivers in a roadway network. On the other hand, PK-BC with $K=1$ still ignores many alternate routes that are practically useful, thus using the PK-BC metric for robust routing purposes may overload few routes and underload many other possible routes in a roadway network.

Table 5.2 shows the calculation of the PK-BC Betweenness metric values for the fish network using

<table>
<thead>
<tr>
<th>Link</th>
<th># of SP’s from 1-&gt;6 through link</th>
<th>Total SP’s from 1-&gt;6</th>
<th>PK-BC</th>
<th>PK-BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-&gt;2</td>
<td>2 (Paths A &amp; B)</td>
<td>2 (A and B)</td>
<td>1</td>
<td>25.0%</td>
</tr>
<tr>
<td>2-&gt;3</td>
<td>1 (Paths A)</td>
<td>2 (A and B)</td>
<td>½</td>
<td>12.5%</td>
</tr>
<tr>
<td>2-&gt;4</td>
<td>1 (Paths B)</td>
<td>2 (A and B)</td>
<td>½</td>
<td>12.5%</td>
</tr>
<tr>
<td>3-&gt;5</td>
<td>1 (Paths A)</td>
<td>2 (A and B)</td>
<td>½</td>
<td>12.5%</td>
</tr>
<tr>
<td>4-&gt;5</td>
<td>1 (Paths B)</td>
<td>2 (A and B)</td>
<td>½</td>
<td>12.5%</td>
</tr>
<tr>
<td>5-&gt;6</td>
<td>2 (Paths A &amp; B)</td>
<td>2 (A and B)</td>
<td>1</td>
<td>25.0%</td>
</tr>
</tbody>
</table>

Table 5.2: PK-BC calculation in the fish network
K=2, by summing the ratio of the number of K shortest paths that go through each link, i.e. $SP_{o,d}(k)$ shown in the 2nd column, divided by the number of K-shortest paths for o-d pair 1->6, i.e. $SP_{1,6}$ shown in the 3rd column. The normalized betweenness sums (as a %) are shown in the last column and in Fig. 5.4. Links 2->3 and 2->4 are equally important here, similar to AK-BC. Moreover, since link 1->2 is 1 out of the 4 links on each of the 2 paths (A and B) without an alternate link, it has a 25% importance in the network. Finally, link 2->3 is 50% less important when compared to link 1->2 due to the alternative path (B) that exists from 1 to 6 not passing through link 2->3.

![Figure 5.4: PK-BC values for Fish network](image)

### 5.2.3 Analyzing the PK-BC Metric on Larger Networks

This section includes the calculation of the SP-BC, AK-BC and PK-BC metrics over two slightly larger networks called the Parking Lot (PL) [67] and Modified Parking Lot (MPL) networks, shown in Fig. 5.5 and Fig. 5.6, respectively. Traffic origin nodes are represented with square shapes and destination nodes are represented with diamond shapes in these figures.

![Figure 5.5: Parking Lot network](image)

![Figure 5.6: Modified Parking Lot network](image)
Table 5.3: Betweenness calculation in the PL network

<table>
<thead>
<tr>
<th>Link</th>
<th>SP-BC</th>
<th>AK-BC</th>
<th>PK-BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-&gt;2</td>
<td>6.8%</td>
<td>6.8%</td>
<td>8.3%</td>
</tr>
<tr>
<td>2-&gt;9</td>
<td>11.9%</td>
<td>11.9%</td>
<td>8.3%</td>
</tr>
<tr>
<td>3-&gt;9</td>
<td>5.9%</td>
<td>5.9%</td>
<td>8.3%</td>
</tr>
<tr>
<td>4-&gt;5</td>
<td>6.8%</td>
<td>6.8%</td>
<td>8.3%</td>
</tr>
<tr>
<td>6-&gt;10</td>
<td>5.1%</td>
<td>5.1%</td>
<td>8.3%</td>
</tr>
<tr>
<td>9-&gt;10</td>
<td>20.3%</td>
<td>20.3%</td>
<td>16.7%</td>
</tr>
<tr>
<td>10-&gt;7</td>
<td>5.1%</td>
<td>5.1%</td>
<td>8.3%</td>
</tr>
<tr>
<td>10-&gt;11</td>
<td>20.3%</td>
<td>20.3%</td>
<td>16.7%</td>
</tr>
<tr>
<td>11-&gt;4</td>
<td>11.9%</td>
<td>11.9%</td>
<td>8.3%</td>
</tr>
<tr>
<td>11-&gt;8</td>
<td>5.9%</td>
<td>5.9%</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

The PL network has equal traffic demand from links 1->5, 3->7, and 6->8, and its results using the 3 metrics are shown in Table 5.3. As expected, the SP-BC and AK-BC are identical since there is only 1 path between all o-d pairs. Moreover, links 2->9 and 11->4 have high importance using these metrics due to the assumption of traffic starting/ending at internal nodes, such as 2 and 4. Since the PK-BC metric only considers the three o-d pairs with traffic demand, all links are found to be of equal importance as the traffic of one o-d pair flows through all of these links, except links 9->10 and 10->11, which carry traffic flowing between two o-d pairs each (1->5 and 3->7 through link 9->10, and 1->5 and 6->8 through link 10->11), as shown by the last column in Table 5.3.

Similarly, the betweenness values for the MPL network are shown in Table 5.4 with K=3, where there is equal traffic demand from links 1->6, 3->8, and 7->9. The results of SP-BC and AK-BC are now different due to the multiple paths that exist for the two o-d pairs 3->8, and 7->9. Finally, based on the PK-BC metric, it is found that:

- Links 10->11 and 11->12 have the highest importance as they are on 1 out of 2 paths for the traffic between one o-d pair and are on the only path for the traffic between another o-d pair.

- Links 4->8, 8->9, 10->4, and 12->9 are of the lowest importance as they contribute to the traffic flowing between one o-d pair and are on 1 out of 2 paths for this o-d pair.

- The remaining links have double the importance of the latter four links as they contribute to the traffic flowing between one o-d pair and are on the only path for this o-d pair.

From the above analysis, it can be concluded that the PK-BC metric measures the topological importance of a link by calculating the % of network traffic (volume) travelling through it while considering K paths between possible network o-d pairs. Moreover, the cost used to calculate its shortest paths can be customized to include any (or combination) of the following: free flow travel times, average travel times, density, etc. Therefore, this metric can be used to identify the most important/critical links in
Table 5.4: Betweenness calculation in the MPL network

<table>
<thead>
<tr>
<th>Link</th>
<th>SP-BC</th>
<th>AK-BC</th>
<th>PK-BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-&gt;2</td>
<td>6.9%</td>
<td>6.9%</td>
<td>8.3%</td>
</tr>
<tr>
<td>2-&gt;10</td>
<td>12.3%</td>
<td>12.3%</td>
<td>8.3%</td>
</tr>
<tr>
<td>3-&gt;10</td>
<td>6.2%</td>
<td>6.2%</td>
<td>8.3%</td>
</tr>
<tr>
<td>4-&gt;8</td>
<td>1.5%</td>
<td>4.1%</td>
<td>4.2%</td>
</tr>
<tr>
<td>5-&gt;6</td>
<td>6.2%</td>
<td>6.2%</td>
<td>8.3%</td>
</tr>
<tr>
<td>7-&gt;11</td>
<td>4.6%</td>
<td>4.6%</td>
<td>8.3%</td>
</tr>
<tr>
<td>8-&gt;9</td>
<td>1.5%</td>
<td>4.4%</td>
<td>4.2%</td>
</tr>
<tr>
<td>10-&gt;4</td>
<td>3.1%</td>
<td>5.6%</td>
<td>4.2%</td>
</tr>
<tr>
<td>10-&gt;11</td>
<td>18.5%</td>
<td>15.9%</td>
<td>12.5%</td>
</tr>
<tr>
<td>11-&gt;8</td>
<td>4.6%</td>
<td>4.9%</td>
<td>8.3%</td>
</tr>
<tr>
<td>11-&gt;12</td>
<td>18.5%</td>
<td>15.6%</td>
<td>12.5%</td>
</tr>
<tr>
<td>12-&gt;5</td>
<td>10.8%</td>
<td>10.8%</td>
<td>8.3%</td>
</tr>
<tr>
<td>12-&gt;9</td>
<td>5.4%</td>
<td>2.6%</td>
<td>4.2%</td>
</tr>
</tbody>
</table>

a roadway network, hence supporting transportation planners in prioritizing investments in roads that are more critical to the performance of the network [46]. It also allows local transportation planners to identify critical links in a specific area of a road network, by first performing the PK-BC analysis for the larger road network and then zooming in to the specific area of interest, in order to take into account the traffic originating or ending outside of that area, in addition to the traffic within the specific area. However, this metric does not take into account dynamic changes in traffic flow, which could be helpful in supporting the decision making of traffic operators in their daily goals of actively controlling traffic demand and supply to alleviate the degradation of network performance due to disturbances [46]. Further extensions to this metric to model dynamic traffic changes are discussed next.

5.2.4 Extending the PK-BC Metric to a Hybrid Robustness Metric

This section discusses the extension of the PK-BC metric to a hybrid robustness metric that can help traffic operators alleviate daily congestion through dynamic traffic control schemes, such as dynamic traffic routing [47], Advanced Traveller Information Systems (ATIS) [75], dynamic congestion pricing [23], Variable Message Signs (VMS), Variable Speed Limits (VSL) [82], and dynamic ramp metering [58].

The PK-BC metric characterizes the topological load or importance of a link in a roadway network, regardless of the nature of the traffic flowing through it. We extend this metric through a dynamic traffic weight, based on the density, speed or travel time of a link, to characterize its importance to a roadway network during a dynamic traffic situation. The resulting metric, called the Hybrid Criticality Index (HCI), is defined as:
In Equation 5.3, the HCI of link $k$ at time $t$ is defined as the ratio of the PK-BC value of link $k$ and the traffic weight $w(k, t)$ of link $k$ at time $t$. As a weight for the betweenness metric, a link weight can be thought of as the inverse of a link cost, with a higher weight signifying a lower traffic load, i.e. increasing the probability of using it [50,69]. Effective examples of a link weight found in the literature for roadway networks include the speed of a link, the inverse of link density and the inverse of link travel time, all of which vary with dynamic traffic conditions. On the one hand, the higher the PK-BC link value, the more the chance of congestion. On the other hand, the link traffic weight has an inverse effect on congestion. As such, the HCI metric measures how critical a link is during the course of a morning traffic period, for example; by quantifying the traffic load of the link from both topological and dynamic traffic flow perspectives. While realizing that calculating the PK-BC and HCI metrics is the ultimate goal, this is computationally cumbersome for large-scale networks. The next section introduces algorithms to efficiently calculate the PK-BC metric (which is then used to calculate the HCI metric) for large-scale networks.

### 5.3 Developing Algorithms to Efficiently Calculate the PK-BC Metric

To calculate the PK-BC metric efficiently for large-scale networks, the focus in this section is on the most time-consuming part of the calculation, which is finding the K shortest paths for all network o-d pairs. Algorithm 1 provides the important steps to calculate the K shortest paths for a network using available codes for sparse large-scale networks in MATLAB. The analysis is done using MATLAB to maintain compatibility with developed codes from our previous research. This implementation was chosen after the implementations based on Yen’s [83] and Eppstein’s [31] K shortest path algorithms, which have the best complexity in theory, were found to be too slow and too memory-demanding for large-scale roadway networks due to their heap-memory usage. In Algorithm 1, for every origin node $o$ (line 1), the algorithm takes the set of destination nodes $D_o$ for this origin (line 2) and calculates the K shortest path for each o-d pair (line 3-8). Therefore, it becomes computationally cumbersome for large-scale networks with a large number of nodes.

K shortest path algorithms such as Algorithm 1 are an extension of shortest path algorithms such as...
Dijkstra’s algorithm [28], developed to find the shortest path between an origin \( o \) and a destination \( d \) in a network. After initializing the distances from \( o \) to all destinations to \( \infty \), a core intermediate step of Dijkstra’s algorithm is to go through network nodes setting them as the current node and to examine all unvisited neighbors of the current node, trying to find shorter paths through these nodes from origins to the destinations in the network. Therefore, while a shortest path (or K shortest path) algorithm can stop iterating through network nodes as soon it finds the shortest path (or K shortest paths) to the desired destination \( d \), it could also continue to calculate the shortest path (or K shortest paths) to all possible remaining destinations in the network through a small number of additional iterations.

Based on this premise, the K-SP algorithm in Algorithm 2 was customized for sparse large-scale roadway networks by re-implementing an Origin-Based (O-Based) version in MATLAB, shown in algorithm 2. For every origin node \( o \) (line 1), this algorithm calculates the K shortest paths to all network destinations concurrently (lines 2-7). It is noteworthy that each inner step in line 5 takes longer than that of Algorithm 1 because it iterates: 1. until there is no more improvement on the K-1 shortest paths found previously, and 2. to ensure that the new paths found are different from all previous shortest paths for all destinations. This was one of the challenges of the implementation, in addition to using a mix of sparse and full matrices to allow the computation to be fast and not run out of memory at the same time. The resulting complexity of Algorithm 2 is \( O(|K| \cdot |N|^2(|E| + |N|\log N)) \) compared to \( O(|K|^2 \cdot |E| \cdot |N|(|E| + |N|\log N)) \) for Algorithm 1, assuming a network of \( |N| \) nodes and \( |E| \) links. The performance of these algorithms is presented in the next section.

```
1: for all \( o \in O_i \) do
2:   for all \( d \in D_o \) do
3:     Find 1 path from \( o \) to \( d \)
4:     \( k \leftarrow 1 \)
5:     while \( k < K \) do
6:       Find new path \( k \) from \( o \) to \( d \)
7:       \( k \leftarrow k + 1 \)
8:     end while
9:   end for
10: end for
```

**Algorithm 1:** O-D Based K-SP Algorithm

```
1: for all \( o \in O_i \) do
2:     Find 1 path from \( o \) to \( d \)
3:     \( k \leftarrow 1 \)
4:     while \( k < K \) do
5:       Find new path \( k \) from \( o \) to \( d \)
6:       \( k \leftarrow k + 1 \)
7:     end while
8: end for
```

**Algorithm 2:** O-Based K-SP Algorithm
5.4 Performance Evaluation

This section presents a performance evaluation of the algorithms to calculate the PK-BC, along with analysis of the PK-BC and HCI metrics in a large-scale network in the GTA.

5.4.1 Performance of Algorithms Calculating the PK-BC Metric

This section analyzes the performance of the K-SP algorithms in calculating the PK-BC metric, described in Section 5.3. The test network used for analysis is the Greater Toronto Area (GTA) shown in Fig. 5.7, which has been calibrated using real data [10, 41]. This network has 14,225 nodes, 26,444 links and 1,602,717 vehicles during a 4-hour simulation of the morning peak period, from 6-10 AM. After running the network demand through a large-scale traffic simulator called DynusT [3], the set of possible o-d pairs for traffic in this network are represented by an \( n \times n \), i.e. 14,225\( \times 14,225 \) matrix, where \( n \) is the number of nodes. Due to the size of the GTA network, each full traffic matrix consumes around 1 GB of computer memory; thus the analysis often used sparse matrices for storing matrices and only used full matrices for fast computation. The following analysis is based on a workstation with a 4.0 GHz Intel core i7 8-core processor and 24 GB of 2133 MHz DDR4 RAM.

With Algorithm 1, it takes an average of around 0.172 sec to calculate the K shortest paths for 1 o-d pair with K=2 using the 8 cores in parallel. This makes the computation time for the PK-BC metric considering the 622,820 o-d pairs in the GTA network equivalent to 29.75 hours. On the other hand, with Algorithm 2, it takes an average of around 1.59 sec to calculate the K shortest paths for 1 origin with
Chapter 5. Robustness in Roadway Networks: A Hybrid Approach

K=2 using the 8 cores in parallel. This makes the computation time for the PK-BC metric considering the 12,279 origins in the GTA network equivalent to 5.42 hours, i.e. more than 5 times faster than the OD-Based K-SP algorithm, as shown in Fig. 5.8.

The computation time increases further with Algorithm 1 for higher values of K, especially because the step on line 6 of Algorithm 1 requires finding a possible new shortest path for every o-d pair by examining paths through various intermediate nodes and then checking whether this path is different from all previous found paths 1 to K-1. This makes the O-Based algorithm relatively more efficient with higher values of K; more than 10 times faster for K=4 and K=6, as seen in Fig. 5.8. Therefore, this O-Based K-SP algorithm can be used to efficiently compute the PK-BC metric or any future tools that require a fast K-SP calculation in large-scale networks.

![Figure 5.8: PK-BC metric calculation times using OD- and O-Based SP Algorithms](image)

5.4.2 Analyzing the PK-BC Metric in the GTA Network

After developing algorithms to calculate the PK-BC metric in large-scale roadway networks, it is calculated with K=4 for the morning peak period from 6-10 AM for the GTA network representing the geographic area in Fig. 5.9.

The analysis involves 2 cases, typical traffic demand calibrated with real data sets in the GTA using DynusT, referred to as the base case; and another case with the same traffic demand but modelling an incident on the highway at location 4 (Gardiner Expressway) in Fig. 5.9. This incident location was chosen after a thorough analysis of 3 factors: 1. the frequency of incidents on a road section in the past few years, 2. the number of alternate routes to consider in case of full closure, and 3. the amount of traffic travelling through the roadway sections, based on public traffic incident reports and analytics reports from the CVST platform [2, 73]. The chosen incident location was among the top 10 incident-prone road sections identified by this analysis and was modelled from 7:30am to 9:00am, blocking all 3
lanes of the highway, which is typical for major collisions due to snow storms or with a fatal collision in Toronto.

The results, shown in Fig. 5.10, include 2 sets of space-time graphs for the speeds from 6:30-9:00am for consecutive freeway links over the 9 major network locations highlighted in Fig. 5.9 on the left (going from upstream to downstream of traffic flow), along with the PK-BC values of the same links (normalized over the 26,444 network links) on the right. All results in this chapter are color-coded with a linear scale between the lowest and highest value in the network in each case.

Analyzing the space-time graphs, the differences in speeds between the base and incident cases can be explained for some of the locations in Fig. 5.9 as follows:

- **Location 4** (the incident location) suffers from very low speeds in the incident case, and the downstream links (which have metered traffic due to the incident upstream) have higher speeds shown in yellow compared to the orange and red in the base case (the Gardiner Expressway typically carries a significant traffic volume destined to downtown Toronto).

- **Locations 5 and 6** are relatively more congested in the incident case as more traffic from the western parts of the GTA heads to Toronto through highway 401 (diverted away from the Gardiner Expressway).
Figure 5.10: Comparison between speeds for the base and incident cases and PK-BC values in the GTA network - each location is represented by a series of upstream and downstream links covering nearby interchanges

- Location 7 (highway 401), gets more congested in the incident case, after around 110min (7:50am), as more traffic diverts there away from the Gardiner Expressway.

Analyzing the PK-BC metric, it is clear that it has one value for each link regardless of the operating conditions of the network (day-day normal operations, or incident case), as it involves a priori calculation without incorporating the changes in traffic patterns. Moreover, the PK-BC metric does not replicate traffic speeds, but rather indicates the critical areas in the network, where congestion typically occurs (i.e., links with high PK-BC values are shown in dark orange and red, and low speeds are shown in dark orange and red).

Therefore, the value of the PK-BC metric is to allow transportation planners to identify a priority list of roadway sections that are critical to the performance of a roadway network, thus require urgent attention to maintain robust networks in the long term. It also allows local transportation planners to identify critical links in a small area of a roadway network, by first performing the PK-BC analysis for the larger roadway network and then zooming in to the specific area of interest, to take into account the traffic originating or ending outside of that small area. The next section presents the analysis with the HCI metric, which could guide traffic operators in alleviating daily traffic congestion through dynamic traffic control schemes.

5.4.3 Analyzing the HCI Metric in the GTA network

This section analyzes the performance of the HCI metric presented in subsection 5.2.4 using the incident case presented in the previous section and through a smaller incident in the GTA network. Although
traffic weights based on link speed, density and travel time provided relatively similar results, link speed was chosen as a traffic weight for HCI here to compare with the previous speed-based analysis in subsection 5.4.2, and due to the ease of real-time speed data collection through loop detectors, road sensors, or GPS.

**Full-closure incident case in the GTA:**

The HCI results for the same freeway links analyzed before are shown in Fig. 5.11 for the incident case presented in subsection 5.4.2. Consider a traffic operator perceiving in real-time, the space-time graph of speeds (middle Figure in Fig. 5.10) versus the space-time graph of HCIs (shown in Fig. 5.11) when an incident occurs at 7:30am during the morning peak, as modelled here. Unlike the speed data, the HCI data clearly indicates a few critical locations in the roadway network (4, 6 and 8) during certain times that a traffic operator can focus on, in order to alleviate the effects of the incident and traffic congestion in the roadway network. This is because HCI is a robustness cost that dynamically measures the impact of traffic disturbances a link would have on the performance of the overall roadway network based on its: 1) topological importance to the network (measured by PK-BC), and 2) current traffic condition (measured by speed).

![Figure 5.11: HCI for the incident case in the GTA network](image)

Analyzing the critical locations in Fig. 5.11, which correspond to the geographic locations in Fig. 5.9, we note the following:

- Location 4 is the incident location, as discussed previously.
• Location 6 starts with a 5-lane link on highway 401 going eastbound, just before it splits into collector and express lanes. This link starts getting relatively critical around the 45-min mark as traffic builds up, and then gets very critical as more traffic from the west diverts to the 401 away from the Gardiner, due to the incident.

• The red section at location 8 is a critical road section on a 3-lane freeway called the Don Valley Parkway (DVP), due to the number of off-ramps in that area and lack of alternatives, and is congested during most morning and afternoon commuting times to and from Toronto.

Therefore, given this dynamic and real-time HCI data, a traffic operator can use various traffic control tools, such as ATIS systems, dynamic congestion pricing or ramp metering to help alleviate the congestion in the network. In our view, HCI can be used as a link robustness cost, along with link travel cost, to dynamically re-route traffic through a dynamic robust traffic assignment algorithm. Such an analysis is performed in Chapter 6. The next section includes analysis using the HCI metric in the GTA network, during a smaller incident in Toronto.

Partial-closure incident case in the GTA:

This section shows the value of the HCI metric for traffic operators even during an incident leading only to a partial lane closure, modelled after an actual incident in Toronto. The incident occurred on the morning of July 9, 2014 on the westbound express lanes of highway 401, at location 7 in Fig. 5.9 (in the opposite direction). It led to the closure of 1 of 4 express lanes from 7:45am to 8:15am, as shown on the right of the camera image looking westbound in Fig. 5.12.

Figure 5.12: Camera snapshot of 401 incident

Fig. 5.13 shows the space-time graph of speeds during the incident using highway loop detector
data provided by the Ministry of Transportation of Ontario (MTO) for links over a 5km range, with the incident location marked with an X. The impact of the shockwave propagation is visible as speeds decrease upstream of the incident location (upwards in the figure). The 3 most upstream links are impacted further as many vehicles slow down to change lanes and exit the express lanes through an off-ramp in that section, to avoid the incident location.

![Figure 5.13: Space-time graph of speed for 401 incident case](image)

5.5 **Conclusion**

This Chapter introduces a hybrid approach for measuring robustness in roadway networks. It extends the shortest-path betweenness metric to a topological robustness metric for roadway networks, called the PK-BC metric, and presents algorithms that calculate it 5-10X faster than existing ones for large-scale networks. Moreover, it further extends this metric with link weights based on traffic flow metrics, leading
Figure 5.14: Space-time graph of Speeds and HCI for 401 partial-closure incident case to the Hybrid Criticality Index (HCI), to model the dynamic variations of roadway network traffic. In addition, it presents simulation analysis with various sized networks, including a large-scale network calibrated with real data sets in the Greater Toronto Area (GTA). The performance results indicated how the PK-BC and HCI metrics can identify critical road sections to help proactively alleviating traffic congestion, for planning and real-time dynamic control, respectively. The next Chapter analyzes how the HCI metric can be used as a robustness cost within a robust dynamic traffic assignment algorithm to dynamically re-route traffic, in order to maintain robust roadway network performance in the presence of disturbances.
Chapter 6

Implementation of Robust Dynamic Traffic Assignment for Roadway Networks

This Chapter presents the implementation of the hybrid robustness metric developed in Chapter 5 for robust routing in roadway networks, using the learnings of robust network design in Chapter 3, within a large-scale Dynamic Traffic Assignment (DTA) simulator model and using the speedup features developed in Chapter 4.

6.1 The Need for a Large-scale Robust DTA Implementation

Dynamic Traffic Assignment algorithms play a key role in real-time ITS strategies, including ATIS systems (Section 2.1). As discussed in Section 2.4, classical traffic assignment algorithms, such as System Optimal (SO) and User Equilibrium (UE), achieve optimal travel times under expected conditions, assuming that 1. drivers have full knowledge of traffic states in the network and 2. they follow the routes based on the optimal resulting assignment. However, the resulting solutions are very sensitive to unexpected traffic disturbances, such as incidents or variations in demand or in traveller route choices (which make the assumptions invalid), leading to escalating travel times and deteriorating system performance, as seen in Chapter 3. In contrast, robust traffic assignment algorithms aim to provide solutions that can deliver high performance over a range of unexpected traffic disturbances, achieving better travel times than the optimal algorithms under certain traffic disturbances. Therefore, a robust traffic assignment
method mitigates the requirement for the system to have full knowledge of traffic states and traveller route choices (required for SO and UE) by leading to an assignment, which is relatively robust (insensitive) to variations in these, given the partial knowledge that it has. In this effort, research attention has been devoted to developing robust traffic assignment algorithms, as an alternative to SO and UE, to enhance the performance of various dynamic ITS strategies, in mitigating the effect of traffic disturbances on a roadway network [17,42,47,52].

One approach is to look at this problem from the viewpoint of assigning traffic based on demand uncertainty [52]. The authors in [52] propose a robust optimization approach to minimize total travel time under the worst-case scenario, such as evacuation for disasters. They use a technique called Robust Optimization to compute feasible solutions for the whole possible set of uncertain demands and a variant of the gradient projection method to solve a reduced version of the problem. While experiments on a small grid and a medium-sized network show fast convergence for the solution, this is a worst-case scenario approach and the algorithm provides high-quality solutions only with specific types of a demand uncertainty set [52].

Another approach is to find the bottleneck links in the network, such as in [42], [17] and [47]. Boris et al. [42] propose a static traffic assignment problem minimizing the probability of congestion, by maximizing the probability that traffic breakdown does not occur at any bottleneck network links, calculated as the product of probabilities of not having traffic breakdown at each of these links. However, this is limited due to 1. assumptions of free flow conditions in the analysis and results being based on the achieved critical traffic flow instead of travel time, 2. the unclear sensitivity analysis method used for traffic demand and supply, and 3. the three-phase traffic flow model, developed by the same authors [42] used in this paper. On the other hand, Cappanera et al. [17] propose a game-theoretic approach for allocating “protection resources” among network components to achieve roadway network robustness to external disturbances. They assume that travel costs and delays are only affected by link status and are not affected by link flow parameters such as congestion. The idea is to apply attacker-defender game-theoretic models to identify the critical components to be hardened (“secured” by increasing its robustness). The proposed solution algorithm is shown to be computationally effective for networks with more than 200 nodes and 1000 links. Nevertheless, the oversimplification of the analysis in [17], which includes neglecting the effects of traffic congestion and generating results only on small grid networks, etc., weakens the contribution and the validity of the proposed approach. Similarly, the robust traffic assignment approach minimizing network criticality, presented in [47] and discussed in Section 3.4, is limited in terms of the size of the network analyzed, the static traffic flow model used and the modelling of a directed network with undirected network weights for calculating network criticality.
Therefore, there is a need for a robust traffic assignment method within a DTA framework and testing on large transportation networks, in order to promote robust DTA methods as credible alternatives for classical ones, such as SO and UE. This chapter introduces such a method, by implementing a robust DTA algorithm based on a hybrid robustness metric within the DynusT DTA-based mesoscopic simulator. Simulation analysis was performed on a large-scale calibrated network, that had used real data to generate the transportation demand and build the network geometry in the Greater Toronto Area (GTA), highlighting the superior performance of robust DTA methods in the presence of traffic disturbances.

Section 6.2 describes the DTA implementation within DynusT. Section 6.3 presents the implementation of the robust DTA method within DynusT. Section 6.4 presents the performance results on a small and large-scale network, using various incidents as disturbances. Finally, Section 6.5 presents the conclusion.

6.2 DTA Model in DynusT

This section describes the DTA modules of the DynusT traffic simulator and details of the traffic assignment process in DynusT.

6.2.1 Components of the DynusT DTA model

As described briefly in Section 4.2, DynusT [3] is a traffic simulator designed to perform simulation-based DTA and associated analysis for roadway networks. DynusT consists of two main modules, namely traffic simulation and traffic assignment, as seen in Fig. 6.1.

Vehicles are loaded into the simulation based on the traffic Origin/Destination (O/D) matrix for a roadway network. The mesoscopic simulation of vehicles omits inter-vehicle car-following details while maintaining realistic macroscopic traffic properties based on the Anisotropic Mesoscopic Simulation (AMS) [21], described in Section 4.2. The traffic assignment module is made of two algorithmic components: a time-dependent shortest-path (TDSP) algorithm and a time-dependent traffic assignment algorithm, or routing, as shown in Fig. 4.1. The TDSP algorithm finds the time-dependent shortest path for vehicles going from each origin to their destinations at a certain departure time defined by the O/D matrix, while the traffic assignment component selects a route for each vehicle following some heuristic rules that are proven to lead to approximate user equilibrium, in which each vehicle selects its least cost path available, as described in Section 2.4. After all the vehicles are simulated, DynusT uses the time gap between a vehicle’s simulated travel time and its available shortest path time to assess the
level of convergence. If the average time gap for all the vehicles in the simulation is small enough, DynusT terminates, otherwise it continues iterating between its two models until convergence is achieved [20].

Based on these DTA modules, DynusT can be run in 2 simulation modes, namely one-shot and iterative simulation modes. On the one hand, the one-shot simulation, which is equivalent to “Iteration 0 Traffic Simulation” in Fig. 6.1, includes the initial TDSP run, assignment of initial vehicles paths and the simulation of vehicle movement through the network. This can be used in certain cases to model the effects of immediate driver reaction to incidents, construction and weather conditions [19]. On the other hand, the iterative simulation, represented by the whole process in Fig. 6.1, re-assigns a % of vehicles after each iteration to their respective best-routes according to the calculated route costs, until convergence is achieved, as described in Section 4.2.
6.2.2 DTA Traffic Assignment Process in DynusT

It is important to look deeper into the iterative assignment process in DynusT, which uses a temporal decomposition scheme called Method of Isochronal Vehicle Assignment (MIVA), shown in Fig. 6.1 and expanded in Fig. 6.2. The design concept is to divide the entire analysis period during the re-routing process into Epochs (evenly or based on computing loads), where vehicles generated in each Epoch are assigned in a parallel (multi-threaded) fashion, as seen in Fig. 6.2. For vehicles departing within a single Epoch, arrival times are used to estimate the Projection Period, in which the domain for the TDSP algorithm is defined for updating path sets for vehicles within the current Epoch. At the end of one Epoch, all TDSP and assignment-related memory is de-allocated, and the process is repeated until completing all Epochs. Therefore, this scheme provides high scalability due to its memory efficiency by confining the peak run-time memory requirement as the maximum memory usage for individual Epoch regardless of the total analysis period, and due to its run-time efficiency enabled by the parallel processing of vehicle assignment during each epoch. This enables DynusT to perform mega scale spatial- and temporal-scale DTA, overcoming the computational challenge of both modelling large-scale roadway networks and maintaining solution algorithm quality [20].

Figure 6.2: Method of Isochronal Vehicle Assignment in DynusT [57]
6.3 Robust DTA Implementation Within DynusT

This section describes the requirements for implementing a robust DTA for roadway networks and presents the details of the implementation.

6.3.1 Requirements for Robust DTA Implementation in Roadway Networks

After developing the Hybrid Criticality Index (HCI) in Chapter 5, the objective here is to develop a robust routing algorithm in roadway networks based on this metric, in order to show its benefits for traffic operators to help proactively alleviate traffic congestion due to disturbances, through real-time ITS strategies based on robust routing.

Given that HCI is a link robustness cost, defined for links in a roadway network at a certain unit of time, one could consider defining a dynamic centralized traffic optimization problem minimizing a global robustness value for the whole roadway network, such as [42] and [47], where this global value is based on the HCIs of all links at every unit of time. However, in order to show the benefits of robust routing and promote it as a credible alternative to UE and SO in real-time ITS applications, it is essential to implement a distributed optimization within a simulation-based DTA, which accurately models traffic flow, congestion, and the dynamic flow of vehicles in real-life through a distributed algorithm.

Therefore, DynusT was chosen for implementing a robust DTA algorithm based on HCI within its algorithmic structure as it meets these requirements, allows network modelling for large-scale networks and can be run efficiently using compiler optimizations and parallelism, as discussed in Chapter 4.

6.3.2 HCI-based Robust DTA Implementation in DynusT

The distributed optimization in DynusT is based on simulating vehicle movement through various routes in the network using route costs, updating these route costs throughout the simulation, and then re-optimizing the routes taken by vehicles to minimize their individual costs, reaching User Equilibrium (UE). Therefore, the HCI-based robust DTA implementation in DynusT includes modifications to the DTA modules in DynusT to incorporate a robustness cost based on HCI in addition to travel time cost. These changes are numbered in Fig. 6.3 and described at a high-level here:

1. The DynusT network data structures were modified to include a PK-BC value for each network link, as defined in Section 5.2 and required for calculating the HCI metric. DynusT’s Input/Output modules were modified to read a vector of PK-BC values for each roadway network, along with other inputs, such as network geometry data, O/D traffic matrix, etc.
Figure 6.3: HCI-Based Robust DTA Implementation in DynusT

2. Several DynusT simulation modules were modified to calculate the HCI value of each link based on link speed and travel time (experiments were conducted based on both). This is repeated every assignment interval (shown in Fig. 6.2) for memory efficiency, i.e. every time new vehicles are added into the simulation, as shown in Figure 4.2.

3. The Dynust TDSP module was modified to incorporate a generalized cost for links, including link HCI, in addition to travel time and link toll (where available). This ensures a robust HCI-based initial TDSP calculation and route assignment (every assignment interval) within a one-shot simulation. Following the guidelines of modelling congestion pricing using a link toll in DynusT [20], the HCI values for a network are normalized to within 5% of free flow link travel times, ensuring that travel times are the major determining factor used by drivers in assessing the cost of a link.

4. The TDSP module was modified to calculate a generalized cost including link HCI every Epoch during the vehicle re-routing process in DynusT. This is possible through mapping between Epochs and Assignment Intervals, as seen in Fig. 6.2.
5. The Gap Function-based reassignment module was modified so that each vehicle selects its least cost path available based on a generalized cost including HCI, thereby optimizing for a new equilibrium assignment based on this generalized cost. This ensures that after convergence is achieved, all vehicles going from the same origin and destination during the same departure time through various routes end up with the same total generalized cost including HCI.

The next section analyzes the performance of this HCI-based robust traffic assignment algorithm in the presence of traffic disturbances.

### 6.4 Performance Results

This section presents a performance evaluation of the developed HCI-based robust DTA implementation in DynusT, compared to the original non-robust DTA implementation based on travel times. The test networks include the Modified Parking Lot (MPL) network, introduced in subsection 5.2.3, and the Greater Toronto Area (GTA) network, introduced in Section 4.6.

#### 6.4.1 MPL Network Analysis

The MPL network, introduced in subsection 5.2.3 is shown here in Fig. 6.4. This network has 12 nodes and 13 3-lane highway links, and there is equal traffic demand from links 1->6, 3->8, and 7->9. Traffic origin nodes are represented with square shapes and destination nodes are represented with diamond shapes. Its PK-BC values calculated with K=3 are shown in Table 6.1.

![Figure 6.4: Modified Parking Lot network](image)

<table>
<thead>
<tr>
<th>Link</th>
<th>1-&gt;2</th>
<th>2-&gt;10</th>
<th>3-&gt;10</th>
<th>4-&gt;8</th>
<th>5-&gt;6</th>
<th>7-&gt;11</th>
<th>8-&gt;9</th>
<th>10-&gt;4</th>
<th>10-&gt;11</th>
<th>11-&gt;8</th>
<th>11-&gt;12</th>
<th>12-&gt;5</th>
<th>12-&gt;9</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK-BC</td>
<td>8.30%</td>
<td>8.30%</td>
<td>8.30%</td>
<td>4.20%</td>
<td>8.30%</td>
<td>8.30%</td>
<td>4.20%</td>
<td>4.20%</td>
<td>12.50%</td>
<td>8.30%</td>
<td>12.50%</td>
<td>8.30%</td>
<td>4.20%</td>
</tr>
</tbody>
</table>

Table 6.1: PK-BC values in the MPL network
The analysis compares the performance of the original non-robust (UE) assignment in DynusT based on minimizing individual travel times and the HCI-based robust assignment in the presence of incidents. For the HCI-based assignment, the PKBC values in Table 6.1 are used and experiments were conducted with both speed and the inverse of travel time as dynamic traffic weights. The network model simulates 200 minutes with typical traffic demand and models an incident from 30-120 minutes using 6 different incident scenarios, shown in Table 6.2. The scenarios include 3 incident locations (links) having different topological importance, i.e. corresponding to the 3 PK-BC values in the network, along with 2 incident severity levels (affecting 1 and 2 lanes out of 3) for each location.

<table>
<thead>
<tr>
<th>Incident</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
<th>Inc4</th>
<th>Inc5</th>
<th>Inc6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>10-&gt;4</td>
<td>10-&gt;4</td>
<td>11-&gt;8</td>
<td>11-&gt;8</td>
<td>11-&gt;12</td>
<td>11-&gt;12</td>
</tr>
<tr>
<td>PK-BC</td>
<td>4.2%</td>
<td>4.2%</td>
<td>8.3%</td>
<td>8.3%</td>
<td>12.5%</td>
<td>12.5%</td>
</tr>
<tr>
<td>Severity</td>
<td>1/3 lanes</td>
<td>2/3 lanes</td>
<td>1/3 lanes</td>
<td>2/3 lanes</td>
<td>1/3 lanes</td>
<td>2/3 lanes</td>
</tr>
</tbody>
</table>

Table 6.2: MPL Incident Scenarios

As discussed in subsection 3.4.2, to systematically compare the performance of each assignment in the presence of incidents, both the UE assignment and HCI-based assignment are first run without an incident in iterative simulation mode with a maximum of 20 iterations, reaching convergence. This models the optimal routes drivers take based on the UE or HCI-based assignment when there are no traffic disturbances. Next, a one-shot simulation is run with one of the incident scenarios in Table 6.2, using the optimal UE assignment as input and this is repeated using the HCI-based assignment as input. This accurately models driver behavior in the presence of an unexpected disturbance with drivers starting with their routes from the UE or HCI-based assignment and some of them diverting from those routes after the incident occurs [19]. The performance of the UE and HCI-based assignment (using the inverse of travel time as link weight) with the 6 incident scenarios is shown in Table 6.3. Total travel times are shown for the base case (the iterative simulation without an incident) and for each incident scenario with a one-shot run, after the iterative runs without an incident for each assignment. The last row in Table 6.3 represents the relative improvement (reduction) of travel times in the HCI-based assignment compared to the UE one in each incident scenario.

Comparing Inc1 and Inc2 scenarios in Table 6.3, it is visible that total travel times increase by around 50% because of Inc2 versus Inc1 for both the UE and HCI-based assignments, due to the increase of the incident severity (affecting 2 lanes out of 3, instead of 1 lane). This is similarly the case for both assignments when comparing Inc3 and Inc4 scenarios affecting link 11->8, and Inc5 and Inc6 scenarios affecting link 11->12, as expected. In addition, the impact on total travel times is much higher in Inc5 and Inc6 scenarios, compared to the other incidents with corresponding incident severity, as the incident
Table 6.3: Total Travel Times (hrs) in MPL network in 6 incident scenarios for UE assignment and HCI-based assignment using inverse of link travel time as weight

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Base Case</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
<th>Inc4</th>
<th>Inc5</th>
<th>Inc6</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE</td>
<td>8528.1</td>
<td>9423.6</td>
<td>15371.4</td>
<td>10830.2</td>
<td>14973.0</td>
<td>13113.6</td>
<td>20661.7</td>
</tr>
<tr>
<td>HCI-based</td>
<td>7764.8</td>
<td>8606.9</td>
<td>14394.1</td>
<td>9994.3</td>
<td>15165.4</td>
<td>11812.6</td>
<td>18910.2</td>
</tr>
<tr>
<td>HCl vs UE</td>
<td>8.95%</td>
<td>8.67%</td>
<td>6.36%</td>
<td>7.72%</td>
<td>-1.29%</td>
<td>10.04%</td>
<td>8.48%</td>
</tr>
</tbody>
</table>

Table 6.3: Total Travel Times (hrs) in MPL network in 6 incident scenarios for UE assignment and HCI-based assignment using inverse of link travel time as weight.

on link 11->12 affects 1 out of 2 paths for the traffic going from node 7->9 and affects the only path going from node 1->6, making link 11->12 among the 2 most important links in the network based on PK-BC values, as shown in Table 6.1.

Looking at the last row in Table 6.3, the robust HCI-based assignment performs better than the UE assignment in the base case. This result might seem contradictory to the robust assignment in subsection 3.4.2, which was found to perform worse than the SO and UE assignments in the base case scenario with no disturbances. However, unlike the results in subsection 3.4.2, which were based on a simplified static traffic flow model, achieving the robust base case here requires thorough optimization until reaching a robust HCI-based equilibrium based on a dynamic traffic flow model. Therefore, based on our findings, we note that it is possible for a robust assignment to perform better than a non-robust one, even in the base case without traffic disturbances, given enough optimization to reach equilibrium conditions based on a dynamic traffic model. Thus, the assignment solutions in subsection 3.4.2 have not actually reached equilibrium conditions based on a dynamic traffic model, and can be thought of as the solutions of running iterative mode with a maximum of 3 iterations for example (instead of 20).

Moreover, the HCI-based assignment performs better than the UE assignment in all but one of the incident scenarios. The HCI-based assignment provides a robust routing mechanism using dynamic costs based on both PK-BC values and dynamic link travel time costs in the network, thus load balancing the traffic between possible alternate paths to reduce peak HCI values. Looking at the HCI values for both assignments, we notice the most critical link in the network is link 10->11 and the HCI-based assignment routes more traffic away from this link compared to the UE assignment in order to maintain a relatively lower HCI value for this critical link. This is observed in Fig. 6.5, which shows the HCI value of link 10->11 throughout the simulation for both the HCI-based and UE assignments during the Inc1 scenario. This link plays a big role in the routing of traffic from nodes 1->6 and 3->8, thus the load balancing of the HCI-based assignment (by reducing the traffic assigned to this link) helps it to perform better than the UE assignment in most incident scenarios, as seen in Table 6.3. The UE assignment performs better than the HCI-based assignment only in the Inc4 scenario, which impacts 2 lanes of a relatively critical link in the network. In this scenario, the HCI-based assignment fails to load
balance the traffic enough in the network to mitigate the impact of the incident by assigning too much traffic to this link, due to the lack of overall alternate paths in the network. This issue is resolved when dealing with large-scale real-world roadway networks with many alternate paths, as discussed in the next section. It is important to note that the UE and HCI-based assignments are equilibrium assignments as no user can reduce their travel time or generalized travel cost, respectively, by choosing another route in these assignments, thus they can be practically implemented in real-life.

Finally, the analysis in Table 6.3 is repeated with the HCI-based assignment using link speed as weight and the performance results compared to the UE assignment with the 6 incident scenarios are shown in Table 6.4. Similar to before, the HCI-based assignment performs better than the UE assignment in the base case and all but one of the incident scenarios, by load balancing the traffic between alternate paths to reduce peak HCI values based on link speed in this case. This HCI-based assignment would be useful in dynamic real-time ITS applications, where measuring link speeds is more accurate or faster than measuring link travel times, thus providing better performance in the presence of traffic disturbances. The next section presents performance results of the HCI-based assignment in the GTA network.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Base Case</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
<th>Inc4</th>
<th>Inc5</th>
<th>Inc6</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE</td>
<td>8528.1</td>
<td>9423.6</td>
<td>15371.4</td>
<td>10830.2</td>
<td>14973.0</td>
<td>13131.6</td>
<td>20661.7</td>
</tr>
<tr>
<td>HCI-based</td>
<td>8291.6</td>
<td>9161.6</td>
<td>15052.4</td>
<td>10490.4</td>
<td>15121.5</td>
<td>12723.0</td>
<td>19859.2</td>
</tr>
<tr>
<td>HCl vs UE</td>
<td>2.77%</td>
<td>2.78%</td>
<td>2.08%</td>
<td>3.14%</td>
<td>-0.99%</td>
<td>3.11%</td>
<td>3.88%</td>
</tr>
</tbody>
</table>

Table 6.4: Total Travel Times (hrs) in MPL network in 6 incident scenarios for UE assignment and HCI-based assignment using link speed as weight.
6.4.2 GTA Network Analysis

The GTA network, first introduced in Section 4.6 is shown here in Fig. 6.6. This large-scale network has 14,225 nodes and 26,444 links, and has been calibrated using real data to generate the transportation demand and build the network geometry [10, 41]. After calculating the PK-BC values for this network with K=4 as described in subsection 5.4.2, the performance of the HCI-based assignment in this network is compared to that of the UE assignment in the presence of incidents. As before, experiments were conducted with both speed and the inverse of travel time as dynamic traffic weights for the HCI-based assignment. The network model simulates the morning peak from 6-10 AM and models 3 incident scenarios of decreasing impact, described in Table 6.5, at locations shown in Fig. 6.6.

The scenarios include 3 incident locations (links) having decreasing topological importance, highlighted by their PK-BC rank among 26446 links in the network, along with different incident severity levels (affecting 1, 2, or 3 lanes) and durations. As described in subsection 5.4.2, the Incident 1 location on the Gardiner was among the top 10 incident-prone road sections in Toronto and it was modelled from 7:30am to 9:00am, blocking all 3 lanes of the highway, which is typical for major collisions due to snow.
storms or with a fatal collision in Toronto. Incident 2 on Highway 401 was modelled after an actual incident in Toronto on the morning of July 9, 2014 as described in Section 5.4.3, and Incident 3 is on one of the busy arterials in Toronto, feeding traffic to Allen Road (which goes north to Highway 401).

As in the MPL network analysis, a one-shot simulation is run with both the UE and HCI-based assignments with one of the incident scenarios in Table 6.5, after iterative runs (base case) without an incident reaching convergence for each assignment. The performance of the UE assignment and HCI-based assignment (using the inverse of travel time as link weight) in the base case and with the 3 incident scenarios is shown in Table 6.6, and the performance of the UE assignment and HCI-based assignment (using speed as link weight) is shown in Table 6.7.

Table 6.5: GTA Incident Scenarios

<table>
<thead>
<tr>
<th>Incident</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Gardiner Expressway</td>
<td>Highway 401</td>
<td>Eglinton</td>
</tr>
<tr>
<td>Direction</td>
<td>Eastbound</td>
<td>Westbound</td>
<td>Westbound</td>
</tr>
<tr>
<td>PK-BC rank</td>
<td>88/26446</td>
<td>181/26446</td>
<td>1223/26446</td>
</tr>
<tr>
<td>Severity</td>
<td>3/3 lanes</td>
<td>1/4 lanes</td>
<td>2/3 lanes</td>
</tr>
<tr>
<td>Duration</td>
<td>7:30-9:00am</td>
<td>7:45-8:15am</td>
<td>7:45-8:15am</td>
</tr>
</tbody>
</table>

As expected, the impact of increasing total travel times is highest in the Inc1 scenario compared to the Base Case in both Table 6.6 and Table 6.7 due to the topological importance of the affected link, and the severity and duration of the incident. Additionally, this impact is relatively higher for the Inc2 scenario compared to the Inc3 scenario due to the higher topological importance of the Highway 401 link affected in Inc2, even though the severity of Inc3 is higher. As seen in Table 6.6 and Table 6.7, the HCI-based assignment, using both the inverse of link travel time and link speed as weight, performs...
better by around 2-3% compared to the UE assignment after reaching convergence without an incident (the base case) and in all incident scenarios. This amounts to a reduced commute of around 7,500 to 16,000 hours every morning in the GTA, which provides savings of around $13.5M to $29M per year assuming only gas costs and not considering the impact of the 88 incidents that occur on average every day in Toronto alone [8]. Note that this assumes a conservative average of $5 in gas costs per hour of driving, instead of overall average value of time that could be considered at $15/hour.

As explained earlier, the HCI-based assignment achieves better performance by load balancing the traffic between alternate paths to reduce peak HCI values in the network and pushing congestion away from critical links. To see this, we compare the UE assignment and HCI-based assignment (using speed as link weight) in the GTA network with a one-shot simulation and modelling incident 1. Fig. 6.7 and Fig. 6.8 show the resulting space-time graphs for the speeds from 6:30-9:00am, with the UE assignment and HCI-based assignment respectively, for consecutive freeway links over the 5 major network locations highlighted in Fig. 6.9 (going from upstream to downstream of traffic flow). Analyzing the space-time graphs, the differences between the 2 cases can be explained for the 5 locations in Fig. 6.9 as follows:

- Location 1 is very similar in both cases with very low speeds at the incident location and the downstream links have metered traffic due to the incident upstream (the Gardiner Expressway typically carries a significant traffic volume destined to downtown Toronto).

- Location 2 starts getting congested due to the incident after the 90min mark (7:30am) in the UE assignment case (Fig. 6.7), as more traffic from the western parts of the GTA heads to Toronto through Highway 401 (diverted away from the Gardiner). However, the congestion at location 2 starts 21 minutes later with the HCI-based assignment as less traffic is routed through this critical location, as highlighted with the dashed line in Fig. 6.8.

- At location 3, the HCI-based assignment reduces the spread of the congestion upstream, compared to the UE assignment as more traffic heads to Toronto through Highway 401. Note that this is equivalent to Location 6 in Fig. 5.9 and Fig. 5.11.

- Location 4 is relatively similar in both cases as congestion builds up on Highway 401 with more traffic coming from Highway 400, although the HCI-based assignment has better speeds after the 120min mark due to more vehicles being routed away from Highway 401 as congestion builds up.

- Finally, the congestion build up at location 5 on Highway 401 (east of Allen Road) is delayed (pushed right) by around 16 min in the HCI-based assignment due to more vehicles being routed south from Highway 401 through Allen Road compared to the UE assignment.
Figure 6.7: Space-time graph of speed with UE Assignment with Incident 1

Figure 6.8: Space-time graph of speed with HCI-based Assignment with Incident 1
Fig. 6.7 and Fig. 6.8 help show how the load balancing of traffic between possible alternate paths in the HCI-assignment helps push away congestion from and reduce its spread at many critical links, improving overall travel times in the network compared to the UE-based assignment, both without and with incidents, as seen in Table 6.6 and Table 6.7.

Another way to compare the solutions of the UE and HCI-based assignments is by looking at the difference between travel times on network links between these assignments in the GTA network with a one-shot simulation and modelling incident 1. Fig. 6.10 and Fig. 6.11 show this color-coded travel time difference (travel times of UE assignment - travel times of HCI-based assignment) at 7:40am (10min after the incident starts) for the entire GTA network and the Toronto area, respectively.

While a few links have higher travel times with the HCI-based assignment, shown in light blue and highlighted with a square in Fig. 6.10, there are many links that have higher travel times with the UE assignment, shown in yellow and orange and highlighted with circles in Fig. 6.10. The higher travel times for the UE assignment are more visible when zooming into Toronto, where the incident has the highest impact, shown by the yellow and orange and highlighted with circles in Fig. 6.11. The situation is similar with even higher travel time differences for the UE assignment after 20, 30 and 40 minutes from the
Chapter 6. Implementation of Robust Dynamic Traffic Assignment for Roadway Networks

Figure 6.10: Difference of link travel times between UE and HCI-based assignments in Inc1 case at t=100 min for the GTA network

Figure 6.11: Difference of link travel times in the Toronto area between UE and HCI-based assignments in Inc1 case at t=100 min for the GTA network
start of the incident (not shown here). The examples in Fig. 6.10 and Fig. 6.11 give an indication of the impact the HCI-based assignment has in reducing travel times around the incident location compared to the UE assignment. Future work could look into further details to measure the difference of total travel times throughout the simulation period within the corridor/area affected by the incident, between the 2 assignments.

Overall, the HCI-based assignment provides a more robust alternative to UE by using dynamic costs based on both PK-BC values and dynamic link costs in the network, regardless of the incident location, or more generally the type of traffic disturbance in the network. Therefore, it can be used by traffic operators in various dynamic real-time ITS applications, either by using link travel times or link speeds as weights (depending on the availability and accuracy of this data), to help proactively alleviate traffic congestion due to disturbances.

Finally, the analysis in Tables 6.6 and 6.7 is repeated using the inverse of travel time and link speed as weight, but using the Shortest Path Betweenness Centrality (SP-BC) metric discussed in subsection 5.2.1 instead of the PK-BC metric, and the performance results are shown in Tables 6.8 and 6.9.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Base Case</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE</td>
<td>605363.6</td>
<td>624493.4</td>
<td>606053.3</td>
<td>605460.9</td>
</tr>
<tr>
<td>HCI with SP-BC</td>
<td>599080.9</td>
<td>611711.3</td>
<td>599024.9</td>
<td>599320.6</td>
</tr>
<tr>
<td>HCI vs UE</td>
<td>1.04%</td>
<td>2.05%</td>
<td>1.16%</td>
<td>1.01%</td>
</tr>
</tbody>
</table>

Table 6.8: Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and HCI-based assignment using SP-BC and inverse of link travel time as weight

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Base Case</th>
<th>Inc1</th>
<th>Inc2</th>
<th>Inc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE</td>
<td>605363.6</td>
<td>624493.4</td>
<td>606053.3</td>
<td>605460.9</td>
</tr>
<tr>
<td>HCI with SP-BC</td>
<td>591156.1</td>
<td>609548.7</td>
<td>591600.4</td>
<td>591457.4</td>
</tr>
<tr>
<td>HCI vs UE</td>
<td>2.35%</td>
<td>2.39%</td>
<td>2.38%</td>
<td>2.31%</td>
</tr>
</tbody>
</table>

Table 6.9: Total Travel Times (hrs) in GTA network in 3 incident scenarios for UE assignment and HCI-based assignment using SP-BC and link speed as weight

This modified HCI-based assignment with the SP-BC metric, using both the inverse of link travel time and link speed as weight, performs better than the UE assignment in the base case and in all incident scenarios. This is because the HCI-based assignment, even when using a preliminary topological robustness metric, such as the SP-BC metric, still considers both the topological importance of links and dynamic traffic conditions in the network, allowing it to load balance traffic to reduce congestion more than UE. However, the performance of this HCI-based assignment does not surpass the performance of the HCI-based assignment with the PK-BC metric, shown in Tables 6.6 and 6.7, as the SP-BC metric considers only 1 shortest path for travelling between any o-d pair and assumes that traffic can originate...
and end in all nodes in the network, as discussed in Section 5.2.

6.5 Conclusion

This Chapter presents an implementation of a robust Dynamic Traffic Assignment (DTA) algorithm for roadway networks. It builds on the algorithmic components of the large-scale mesoscopic DynusT simulator by incorporating the Hybrid Criticality Index (HCI) as a link robustness cost along with travel time in DynusT and modifying its assignment and routing modules to route vehicles based on a generalized cost including HCI. This HCI-based assignment was then evaluated compared to the User Equilibrium (UE) assignment on a small network and a calibrated large-scale network in the Greater Toronto Area, using various incidents as disturbances. Performance results showed that the HCI-based assignment, based on both the topological importance of links and dynamic links costs, load balances the traffic in a roadway network and pushes congestion away from critical links, thus improving overall travel times compared to the UE-based assignment, with and without the presence of a range of traffic incidents. This makes a strong case for traffic operators to use robust DTA methods based on HCI within actual implementations of real-time ITS strategies to help proactively alleviate traffic congestion due to disturbances.
Chapter 7

Conclusions and Future Work

The economic cost of traffic congestion is growing massively in metropolitan cities with continuing population growth and global trends of urbanization. At the same time, building more transportation infrastructure to support this growth or even fixing aging infrastructure is becoming more expensive and difficult to implement. These factors have made it very challenging to manage traffic congestion by increasing traffic supply through building new roads or dynamically operating the daily flow of traffic. Hence, the research community and the world have been examining new ways to reduce traffic congestion in our cities.

While the growth in car sharing, autonomous vehicles and improving transit will help, they are not enough to reduce the variations in traffic demand and traffic supply due to weather conditions, road closures and the occurrence of incidents, etc., which act as traffic disturbances in roadway networks and cause congestion. In response to this, we have developed robust dynamic route guidance algorithms to reduce traffic congestion in roadway networks due to these disturbances. In this chapter, we review the contributions of this thesis and present future research directions.

7.1 Contribution

This thesis focused on the problem of robustness in roadway networks. It has resolved the issue of the many disparate robustness methods for road networks developed in the literature by introducing a systematic framework for developing a robust design for roadway networks. It has also presented new metrics and algorithms to quantify the robustness of roadway networks and reduce the impact of traffic disturbances. While other research works ignore the importance of topology in modelling robustness or simplify the traffic dynamics in roadway networks, this work extended the Shortest-Path
Betweenness metric, from the field of network science, to apply it in roadway networks and demonstrated its applicability within a dynamic large-scale simulator that accurately models traffic congestion and the dynamic flow of vehicles.

The contributions of this thesis can be organized into four domains: 1) Robust network design for roadway networks, 2) speedup of dynamic traffic assignment simulation for real-time ITS applications, 3) a hybrid approach for robustness in roadway networks, and 4) implementation of robust traffic assignment algorithms for roadway networks. In this section, we provide a brief overview of these contributions.

7.1.1 Robust Network Design for Roadway Networks

First, we examined the notion of robustness and what it meant for roadway networks in Chapter 3. Looking at the many disparate robustness methods proposed in the literature, we defined the various contexts of robustness based on the severity, frequency and predictability of traffic disturbances and presented a definition of robustness in roadway networks.

By examining the various goals of robust network design, including long-term transportation planning and short-term traffic operations, we introduced a systematic framework for developing a robust design for roadway networks based on the contexts of robustness and network design goals, and identified criteria for assessing robustness metrics in each case. We also provided examples of robustness metrics for long-term and short-term robust network design from the literature and assessed them based on the established criteria.

Finally, we presented a sample study for a short-term robust network design using traffic assignment and tested it on a highway network of Toronto. Here, we formulated a traffic assignment problem based on the network criticality metric, from network science, and studied the effect of increasing traffic demand and decreasing supply on the performance of the roadway network, with various traffic assignment algorithms. Simulation results showed the potential of robust assignment algorithms, to deliver high performance over a range of unexpected traffic disturbances, even achieving better travel times than classical traffic assignment algorithms under certain traffic disturbances.

7.1.2 Speedup of Dynamic Traffic Assignment Simulation for Real-Time ITS Applications

The next area of research we explored was speeding up traffic assignment algorithms to support real-time ITS applications, including robust dynamic route guidance systems, in Chapter 4. In this effort, we studied the components of traffic assignment within traffic simulators and presented methods to speed up
traffic assignment algorithms through compiler optimizations and parallelism. In addition, we presented
the theoretical limits of reducing algorithm execution times by increasing the number of processors used
indefinitely.

We conducted extensive experiments to study the efficiency (with number of CPUs and optimization
levels), performance in various systems (Linux vs Windows), and scalability (with different network
sizes) of the proposed speed-up approaches. The results showed that the execution time for a 12-
iteration simulation run and a one-shot simulation (to model incidents) can be reduced using compiler
optimizations and 4 cores by around 50% and 60%, respectively. We also showed that increasing the
number of processors beyond a certain point actually increases execution times due to the communication
latency between processors.

Finally, we converted the open-source code of the large-scale multithreaded mesoscopic traffic simu-
lator, namely DynusT, to run on Linux, enabling it to be used on more high-performance clusters, and
can thus be run using the fastest available processors to enable real-time ITS applications.

7.1.3 A Hybrid Approach for Robustness in Roadway Networks

In Chapter 5, we explored the development of robustness metrics for roadway networks. While topological
metrics from the field of network science, such as the Shortest-Path Betweenness metric, have been used in
various fields such as communication networks, smart grids and recently transportation networks, these
metrics often simplify and sometimes overlook the traffic dynamics in roadway networks. Therefore,
we first developed a topological robustness metric specifically customized for roadway networks, by
extending the Shortest-Path Betweenness metric. Simulation results showed that this metric helps
transportation planners identify a priority list of important road sections in a road network, guiding
their planning initiatives and some of their budget allocation.

Next, we looked at the requirements for robust roadway network design for short-term traffic opera-
tions, established in Chapter 3, and noticed that operational traffic metrics, proposed in the literature,
may provide myopic results under various traffic conditions due to ignoring the importance of topology
in modelling roadway network robustness. Thus, we further augmented the topological robustness met-
ric with dynamic graph weights based on traffic flow metrics, introducing a hybrid robustness metric
called HCI for roadway networks. Simulations with various traffic incidents showed how this metric
helps dynamically identify critical locations in a road network in real-time, guiding traffic operators to
apply dynamic traffic control schemes to alleviate the impact of traffic disturbances.

In our path to calculate these metrics efficiently for large-scale networks, we developed fast K shortest-
path calculation algorithms customized for large-scale networks that can be used for various types of networks and to calculate any future K shortest-path based metrics. While existing shortest path algorithms were found to be too slow and too memory-demanding for large-scale roadway networks, the developed algorithms, which balanced the use of sparse and full matrices to allow fast computation and maintain memory efficiency, were shown to be 5-10X faster than current K shortest-path algorithms for large-scale networks.

7.1.4 Implementation of Robust Traffic Assignment Algorithms for Roadway Networks

In Chapter 6, we implemented a robust HCI-based dynamic traffic assignment algorithm within the algorithmic components of DynusT. We evaluated the performance of this assignment compared to existing travel assignment methods using a small Parking Lot network (used for robustness analysis of networks) and a calibrated large-scale model of the Greater Toronto Area (GTA), in the presence of incidents as traffic disturbances.

Extensive simulation results showed the robust assignment algorithm improves vehicle travel times with and without the presence of incidents, by using dynamic costs based on both the topological importance of road sections and dynamic link costs in the network. The reduction in travel times would provide savings of around $13.5M to $29M per year in the GTA, even when assuming only gas costs.

Additionally, the fact that the calibrated GTA network used for the analysis was based on real data to generate the transportation demand and build the network geometry, makes the interpretation and generalization of the results sensible to be used in actual implementations of real-time ITS strategies. Furthermore, it supports the case for operators to use such robust traffic assignment methods, similar to the one developed in this Thesis, within ITS strategies, such as dynamic route guidance systems, to proactively alleviate traffic congestion due to disturbances.

7.2 Future Work

There are several interesting topics in the area of robust roadway networks that remain for future investigation. In this section, we provide an overview of some of these topics.
7.2.1 Studying the Effect of Increasing Demand

Although we were able to examine the impact of decreasing traffic supply through incidents on the performance of the HCI-based assignment, to show the robustness of the HCI-based assignment with various types of traffic disturbances, further effort is required to study the impact of sudden increases in traffic demand on the HCI-based assignment compared to existing assignment methods. Given the algorithmic structure of DynusT, it would take a significant implementation effort to increase the traffic demand in the network after running a base case scenario with regular demand. In addition, further research is required along with detailed analysis of traffic demand variations in order to appropriately model the additional demand to accurately represent real-life variations, e.g. traffic demand increase due to travellers heading to a sports game or sudden traffic demand increase due to an emergency evacuation.

7.2.2 Mobile Application Providing Travel Time Reliability Based on HCI

Although we were able to show the benefits of the HCI-based assignment in the presence of traffic disturbances, a remaining question left for future study due to time limitations is how to incentivize travellers to follow routes resulting from the robust HCI-based assignment in order to provide a practical implementation of an HCI-based assignment in real-life dynamic ITS systems through the mass public. One possible route is to develop a mobile route guidance application providing travel time reliability based on HCI. In this regard, additional research effort is required to quantify the travel time reliability of routes in a route guidance application based on the HCI robustness values of links on each route. In terms of the mobile application model, instead of providing 2-3 routes such as the ones in Google Maps with no %’s for the reliability of each route, the envisioned application would provide 2-3 routes with specific reliability values, e.g. Route 1: 20 min, reliability: 75%, and Route 2: 24 min, reliability 95% (based on lower HCI values on the links of this route and its alternate routes). While many drivers will choose to take Route 1 due to its lowest expected travel time, some would choose to take Route 2 due to the higher reliability of that Route, e.g. while trying to make it to an important meeting on time. In addition, this reliability measure serves as a non-monetary incentive (or lack of penalty in terms of a toll with congestion pricing) to route some people away from Route 1, and pave the way to achieving an HCI-based equilibrium, and reducing overall travel times in the network. This also provides a fast way, visible at this point, for commercializing this work within a mobile route guidance application. Alternatively, if using a non-monetary incentive of reliability is not enough to incentivize some people away from Route 1 to achieve an HCI-based equilibrium, a monetary disincentive through congestion pricing is a possible alternative (but would require traffic agency involvement).
7.2.3 City-Based Robust ATIS system

Another question left for future study is how to promote a practical implementation of an HCI-based assignment in real-life dynamic ITS systems for a traffic agency. One possible route is through implementing a pilot ATIS system based on the HCI-based assignment for a traffic agency in a major city. In this regard, additional research effort is required for implementing the System Optimum (SO) assignment, which reduces overall network travel times, within DynusT. This would allow extensive comparisons for the performance of the HCI-based assignment to that of SO, with and without the presence of various traffic disturbances in a large-scale traffic model, in addition to the preliminary SO-based implementation on a smaller scale metropolitan highway network in Toronto using a static traffic assignment model in Chapter 3. Although SO is more of a theoretical assignment, it would be critical to show the robustness of the HCI-based assignment compared to the SO assignment to promote the implementation of an HCI-based assignment within real-time dynamic ITS strategies to traffic agencies and pave the way to developing a pilot of a city-based robust ATIS system.
Bibliography


